Games on Graphs

Nathanaël Fijalkow, Nathalie Bertrand, Patricia Bouyer, Romain Brenguier, Arnaud Carayol, John Fearnley, Hugo Gimbert, Florian Horn, Rasmus Ibsen-Jensen, Nicolas Markey, et al.

Games on Graphs

(version 1)

Nathalie Bertrand Patricia Bouyer-Decitre
Romain Brenguier Arnaud Carayol John Fearnley
Nathanaël Fijalkow Hugo Gimbert Florian Horn
Rasmus Ibsen-Jensen Nicolas Markey Benjamin Monmege
Petr Novotný Mickael Randour Ocan Sankur
Sylvain Schmitz Olivier Serre Mateusz Skomra

Coordinated by Nathanaël Fijalkow

GAMES ON GRAPHS

Preface

What is this book about?

The objective of this book is to present the state of the art on games on graphs, which is part of a larger research topic called game theory. Games on graphs is the field concerned with games whose rules and evolution are represented by a graph. We mostly focus on infinite duration games, but their study is deeply interleaved with finite duration games. They form a prominent model in two related subjects: the first is automata theory and logic, and the second is verification and synthesis, both of which have been very active for decades. Some of the models were introduced and studied in neighbouring research communities such as optimisation, reinforcement learning, model theory, and set theory.

This book does not claim to give a full account of all existing results or models in the litterature, which is close to impossible for two reasons: the wealth of existing results and the constant flow of new ones.

The primary objective in this book is algorithmic: constructing efficient algorithms for analysing different types of games. Yet the goal is not to describe their implementation in full details but rather to explain their theoretical foundations. In this endeavour we often need to set the stage by proving properties of the corresponding games and most prominently of their winning strategies. So the language of this book is mathematics.

This book owes a lot to two reference textbooks on games: Automata, Logics, and Infinite Games: A Guide to Current Research, edited by Erich Grädel, Wolfgang Thomas, and Thomas Wilke [GTW02], and Lectures in Game Theory for Computer Scientists, edited by Krzysztof R. Apt and Erich Grädel [AG11].

How to read

All the material presented in this book is accessible to an advanced master student or a PhD student with a background in computer science or mathematics. The goal is at the same time to present all the basic and fundamental results commonly assumed by

the research community working on games on graphs, and most of the latest prominent advances. We assume familiarity with complexity theory and the notions of graphs and automata but as much as possible do not rely on advanced results in these fields.

The book is divided in five parts each including two or three chapters. At the end of each chapter is a section dedicated to bibliographic references. Chapter 1 introduces some notations and notions used throughout the book. After that and to some extent each part is independent. As much as possible we avoid back references but some chapters naturally build on the previous ones in which case we clearly indicate this.

How to cite

To cite the whole book, here is a bib item.

```
@book{gamesbook,
  title     = {Games on Graphs},
  author    = {Nathanaël Fijalkow and
  Nathalie Bertrand and
  Patricia Bouyer-Decitre and
  Romain Brenguier and
  Arnaud Carayol and
  John Fearnley and
  Hugo Gimbert and
  Florian Horn and
  Rasmus Ibsen-Jensen and
  Nicolas Markey and
  Benjamin Monmege and
  Petr Novotný and
  Mickael Randour and
  Ocan Sankur and
  Sylvain Schmitz and
  Olivier Serre and
  Mateusz Skomra},
  editor    = {Nathanaël Fijalkow},
  publisher = {Online},
  date      = {2023},
}
```

If you wish to only cite one chapter, here is an example.

```
@InCollection{timedgameschapter,
  title     = {Timed Games},
  author    = {Nicolas Markey, Ocan Sankur},
  booktitle = {Games on Graphs},
  editor    = {Nathanaël Fijalkow},
  year      = {2023}
}
```

Acknowledgements

The following people have contributed to early versions of the book in different ways, we thank them warmly for their comments, suggestions, discussions, bug fixes and reports: Antonio Casares, Hugo Francon, Pierre Gaillard, Théo Matricon, Rémi Morvan, Damian Niwiński, Pierre Ohlmann.

Please send any comments (including typos) you may have to

nathanael.fijalkow@gmail.com,

or directly to the relevant authors.

Versions

A full version of the book is available on ArXiv. The current document is version 1. A printed published version will be considered at a later point. An online HTML version is under preparation.

Illustrations

The illustrations (cover and each chapter) were realised by Podpunkt:

http://podpunkt.pl/

Contents

INTRODUCTION

Chapter 1

Introduction

NATHANAËL FIJALKOW

Section 1.1 defines a first model of games, which is the common denominator of (almost all) models studied in this book. We introduce the computational models that we use in Section 1.2 and briefly discuss linear programming. In Section 1.3 we list the main objectives appearing in all chapters. A few notions will be useful throughout the book: they are developed in this chapter. We start with the notion of automata, discussed in Section 1.4, and then memory for strategies in Section 1.5. We then show how automata and memory structures can be used to construct reductions between games in Section 1.6. We introduce in Section 1.7 the notions of subgames and traps.

The notion of fixed point algorithms is central to the study of games. We first recall the main two methods for proving the existence and computing fixed points in Section 1.8. We then give an overview of two prominent families of fixed point algorithms for games: value iteration algorithms in Section 1.9 and strategy improvement algorithms in Section 1.10.

Usual notations

We write $[i, j]$ for the interval $\{i, i+1, \dots, j-1, j\}$, and use parentheses to exclude extremal values, so $[i, j)$ is $\{i, i+1, \dots, j-1\}$.

An alphabet Σ is a finite set. We let Σ^* denote the set of finite sequences of Σ (also called finite words), Σ^+ the subset of non-empty sequences, and Σ^ω the set of infinite sequences of Σ (also called infinite words). For a (finite or infinite) sequence $u = u_0 u_1 \cdots$, we let u_i denote the i^{th} element of u and $u_{<i}$ the prefix of u of length i, *i.e.* the finite sequence $u_0 u_1 \cdots u_{i-1}$. Similarly $u_{\leq i} = u_0 u_1 \cdots u_i$. The length of u is $|u|$.

1.1 A first model of games

The first model we define is the common denominator of most models studied in this book:

- 2-player,
- zero sum,
- turn based,
- deterministic,
- perfect information

game.

Players

The term 2-player means that there are two players. Many, many different names have been used: Eve and Adam, Player 0 and Player 1, Player I and Player II as in descriptive complexity, Éloïse and Abélard, Circle and Square, corresponding to the graphical representation, Even and Odd, mostly for parity objectives, Player and Opponent, Pathfinder and Verifier in the context of automata, Min and Max, which makes sense for quantitative objectives, and this is only a very partial list of names they have been given. In the names Eve and Adam, the first letters refer to $\exists$ and $\forall$ suggesting a duality between them. We will use the pronouns she/her for Eve and he/him for Adam, so we can speak of her or his strategy. In this book, we will use Eve and Adam in qualitative contexts (win / lose), and Min and Max in quantitative contexts.

We speak of 1-player games when there is only one player. In the context of stochastic games, we refer to random as a third player, and more precisely as half a player. Hence a $2\frac{1}{2}$-player game is a stochastic game with two players, and a $1\frac{1}{2}$-player game is a stochastic game with one player.

The situation where there are more than two players is called multiplayer games.

Graphs

A (directed) graph is given by a set V of vertices and a set E of edges given by the functions $\mathrm{In}, \mathrm{Out} : E \to V$: for an edge e we write $\mathrm{In}(e)$ for the incoming vertex and $\mathrm{Out}(e)$ for the outgoing vertex. We say that e is an outgoing edge of $\mathrm{In}(e)$ and an incoming edge to $\mathrm{Out}(e)$. To introduce an edge, it is convenient to write $e = v \to v'$ to express that $v = \mathrm{In}(e)$ and $v' = \mathrm{Out}(e)$. A path π is a finite or infinite sequence:

$$\pi = v_0 \to v_1 \to v_2 \cdots$$

We let $\mathrm{first}(\pi)$ denote the first vertex occurring in π and $\mathrm{last}(\pi)$ the last one if π is finite. We say that π starts from $\mathrm{first}(\pi)$ and if π is finite that π ends in $\mathrm{last}(\pi)$. We write $\pi_{\leq i}$ for the finite path $v_0 \to v_1 \to \cdots \to v_i$. We sometimes talk of a path and let the context determine whether it is finite or infinite.

We let $\mathrm{Paths}(G)$ denote the set of finite paths in the graph G, sometimes omitting G when clear from the context. To restrict to paths starting from v we write $\mathrm{Paths}(G,v)$. The set of infinite paths is $\mathrm{Paths}_\omega(G)$, and $\mathrm{Paths}_\omega(G,v)$ for those starting from v. We naturally see sets of infinite paths as subsets of E^ω.

We use the standard terminology of graphs: for instance a vertex v' is a successor of v if there exists $e \in E$ such that $\mathrm{In}(e) = v$ and $\mathrm{Out}(e) = v'$, and then v is a predecessor of v', a vertex v' is reachable from v if there exists a path starting from v and ending in v', the outdegree of a vertex is its number of outgoing edges, the indegree is its number of incoming edges, a simple path is a path with no repetitions of vertices, a cycle is a path whose first and last vertex coincide, it is a simple cycle if it does not strictly contain another cycle, a self loop is an edge from a vertex to itself, and a sink is a vertex with only self loops as outgoing edges.

Remark 1 (Edges as subsets of pairs of vertices)**.** *Our definition with $In, Out : E \to V$ includes graphs where there can be multiple edges between a single pair of vertices. Although this can be useful in some cases, we will also sometimes simply define the set of edges as $E \subseteq V \times V$, which implies that an edge is fully determined by incoming and outgoing vertices.*

Arenas

The *arena* is the place where the game is played, they have also been called game structures or game graphs.

In the turn based setting we define here, the set of vertices is divided into vertices controlled by each player. Since we are for now interested in 2-*player* games, we have $V = V_{\mathrm{Eve}} \uplus V_{\mathrm{Adam}}$, where V_{Eve} is the set of vertices controlled by Eve and V_{Adam} the set of vertices controlled by Adam. We represent vertices in V_{Eve} by circles, and vertices in V_{Adam} by squares, and also say that $v \in V_{\mathrm{Eve}}$ belongs to Eve, or that Eve owns or controls v, and similarly for Adam. An arena is given by a graph and the sets V_{Eve} and V_{Adam}. In the context of games, vertices are often referred to as positions.

We adapt the notations to $V = V_{\mathrm{Min}} \uplus V_{\mathrm{Max}}$ if the players are Min and Max. Circles are controlled by Max and squares by Min.

The adjective *finite* means that the arena is finite, *i.e.* there are finitely many vertices and edges. We oppose *deterministic* to *stochastic*: in the first definition we are giving here, there is no stochastic aspect in the game. An important assumption, called *perfect information*, says that the players see everything about how the game is played out, in particular they see the other player's moves.

We assume that all vertices have an outgoing edge. This is for technical convenience, as it implies that we do not need to explain what happens when a play cannot be prolonged.

Playing

The interaction between the two players consists in moving a token on the vertices of the arena. The token is initially on some vertex. When the token is in some vertex v,

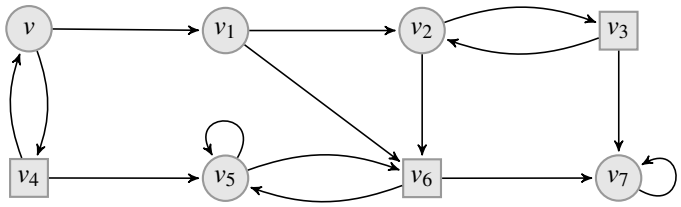


Figure 1.1: An example of an arena. In a qualitative context, circles are controlled by Eve and squares by Adam, and in a quantitative context circles are controlled by Max and squares by Min.

the player who controls the vertex chooses an edge $e = v \to v'$ and pushes the token along this edge to the next vertex v'. The outcome of this interaction is the sequence of vertices and edges traversed by the token: it is a path. In the context of games a path is also called a *play* and as for paths usually written π. We note that plays can be finite (but non empty) or infinite.

Strategies

The most important notion in this book is that of *strategies* (sometimes called policies). A strategy for a player is a full description of his or her moves in all situations. Formally, a *strategy* is a function mapping finite plays to edges:

$$\sigma : \text{Paths} \to E.$$

Convention 1. *We use σ for strategies of Eve and Max, and τ for strategies of Adam and Min, so when considering a strategy σ it is implicitly for Eve or Max, and similarly τ is implicitly a strategy for Adam or Min.*

We say that a play $\pi = v_0 \to v_1 \to \dots$ is consistent with a strategy σ of Eve if for all i such that $v_i \in V_{\text{Eve}}$ we have $\sigma(\pi_{\leq i}) = v_i \to v_{i+1}$. The definition is easily adapted for strategies of Adam.

Once an initial vertex v and two strategies σ and τ have been fixed, there exists a unique infinite play starting from v and consistent with both strategies, it is written $\pi^v_{\sigma,\tau}$. Note that the fact that it is infinite follows from our assumption that all vertices have an outgoing edge.

Conditions

The last ingredient to wrap up the definitions is (winning) *conditions*, which is what Eve wants to achieve. There are two types of conditions: the *qualitative*, or Boolean ones, and the *quantitative* ones.

A *qualitative condition* is $W \subseteq \text{Paths}_\omega$: it separates winning from losing plays, in other words a play which belongs to W is winning and otherwise it is losing. We also say that the play satisfies W.

A *quantitative condition* is $f : \text{Paths}_\omega \to \mathsf{R} \cup \{\pm\infty\}$: it assigns a real value (or plus or minus infinity) to a play, which can be thought of as a payoff or a score. More generally, we sometimes consider quantitative conditions of the form $f : \text{Paths}_\omega \to Y$ with Y a totally ordered set.

Often we define W as a subset of E^ω and f as $f : E^\omega \to \mathsf{R} \cup \{\pm\infty\}$, since Paths_ω is included in E^ω.

Objectives

To reason about classes of games with the same conditions, we introduce the notions of *objectives* and colouring functions. An objective and a colouring function together induce a condition. The main point is that *objectives are independent of the arenas*, so we can speak of the class of conditions induced by a given objective, and by extension a class of games induced by a given objective, for instance parity games.

We fix a set C of colours. A *qualitative objective* is $\Omega \subseteq C^\omega$, and a *quantitative objective* is a function $\Phi : C^\omega \to \mathsf{R} \cup \{\pm\infty\}$.

Colouring functions

The link between an arena and an objective is given by a *colouring function* $\mathfrak{c} : E \to C$ labelling edges of the graph by colours. We extend $\mathfrak{c}$ componentwise to induce $\mathfrak{c} : \text{Paths}_\omega \to C^\omega$ mapping plays to sequences of colours:

$$\mathfrak{c}(e_0 e_1 \dots) = \mathfrak{c}(e_0)\ \mathfrak{c}(e_1) \dots$$

A qualitative objective Ω and a colouring function $\mathfrak{c}$ induce a qualitative condition $\Omega(\mathfrak{c})$ defined by:

$$\Omega(\mathfrak{c}) = \{\pi \in \text{Paths}_\omega : \mathfrak{c}(\pi) \in \Omega\}.$$

When $\mathfrak{c}$ is clear from the context we sometimes say that a play π satisfies Ω but the intended meaning is that π satisfies $\Omega(\mathfrak{c})$, equivalently that $\mathfrak{c}(\pi) \in \Omega$.

Similarly, a quantitative objective $\Phi : C^\omega \to \mathsf{R} \cup \{\pm\infty\}$ and a colouring function $\mathfrak{c}$ induce a quantitative condition $\Phi(\mathfrak{c}) : \text{Paths}_\omega \to \mathsf{R} \cup \{\pm\infty\}$ defined by:

$$\Phi(\mathfrak{c})(\pi) = \Phi(\mathfrak{c}(\pi)).$$

We extend the notation $v \to v'$ in the presence of a colouring function: $e = v \xrightarrow{c} v'$ further indicates that $\mathfrak{c}(e) = c$. As we will see, a convenient abuse of notations consists in identifying a colour c and the set of edges of that colour $\mathfrak{c}^{-1}(c)$.

Games

We can now give the following definitions.

Definition 1 (Games)**.**

- *A graph is a tuple $G = (V, E, In, Out)$ where V is a set of vertices, E is a set of edges, and $In, Out : E \to V$ define the incoming and outgoing vertices of edges.*

- *An arena is a tuple* $\mathscr{A} = (G, V_{\text{Eve}}, V_{\text{Adam}})$ *where* G *is a graph over the set of vertices* V *and* $V = V_{\text{Eve}} \uplus V_{\text{Adam}}$. *In a quantitative context we use* $V = V_{\text{Min}} \uplus V_{\text{Max}}$.
- *A colouring function is a function* $\mathfrak{c} : E \to C$ *where* C *is a set of colours.*
- *A qualitative condition is* $W \subseteq Paths_\omega$.
- *A qualitative objective is a subset* $\Omega \subseteq C^\omega$. *A colouring function* $\mathfrak{c}$ *and a qualitative objective* Ω *induce a qualitative condition* $\Omega(\mathfrak{c})$.
- *A* qualitative game $\mathscr{G}$ *is a tuple* $(\mathscr{A}, W)$ *where* $\mathscr{A}$ *is an arena and* W *a qualitative condition.*
- *A quantitative condition is* $f : Paths_\omega \to \mathsf{R} \cup \{\pm\infty\}$.
- *A quantitative objective* Φ *is a function* $\Phi : C^\omega \to \mathsf{R} \cup \{\pm\infty\}$. *A colouring function* $\mathfrak{c}$ *and a quantitative objective* Φ *induce a quantitative condition* $\Phi(\mathfrak{c})$.
- *A* quantitative game $\mathscr{G}$ *is a tuple* $(\mathscr{A}, f)$ *where* $\mathscr{A}$ *is an arena and* f *a quantitative condition.*

To be specific, the definition above is for 2-player zero sum turn based deterministic perfect information games. As a convention we use the condition to qualify games, so for instance parity games are games equipped with a parity condition. This extends to graphs: we speak of a graph with condition W for a graph equipped with a condition W, and for instance a mean payoff graph if W is a mean payoff condition.

We often introduce notations implicitly: for instance when we introduce a qualitative game $\mathscr{G}$ without specifying the arena and the condition, it is understood that the arena is $\mathscr{A}$ and the condition W.

We always implicitly take the point of view of Eve. Since we consider zero sum games we can easily reverse the point of view by considering the qualitative game $(\mathscr{A}, \text{Paths}_\omega \setminus W)$ and the quantitative game $(\mathscr{A}, -f)$. Indeed for the latter minimising f is equivalent to maximising $-f$. The term zero sum comes from this: the total outcome for the two players is $f + (-f)$, meaning zero.

We sometimes speak of a coloured graphs or a coloured arena when attaching a colouring function to a graph or an arena.

Remark 2 (Finite graphs)**.** *Unless otherwise stated we assume that graphs are finite, meaning that there are finitely many vertices and finitely many edges. We equivalently say that the arena or the game is finite. Part IV will study games over infinite graphs.*

Labelling edges or vertices

In our definition the colouring function labels edges: $\mathfrak{c} : C \to E$, and infinite paths are infinite sets of edges. It is sometimes convenient to label vertices instead: $\mathfrak{c} : C \to V$, and in some cases it may feel more natural: for instance for reachability conditions (to be defined in Section 1.3), the target is naturally a set of vertices rather than a set of edges. But labelling edges is often technically easier, and more succinct, so it is the

preferred convention in this book. There is a third option: labelling vertices with partial colouring functions, *i.e.* allowing some vertices to remain uncoloured. To keep things well defined, in that case we make the assumption that every path contains infinitely many coloured vertices. There are two consequences to the convention of labelling edges or vertices: the complexity of algorithms, and the memory requirements, both (often) mildly affected by the choice.

Let us state simple and general reduction results between the three conventions. Clearly,

$$\text{vertex labelling} \subseteq \text{partial vertex labelling} \subseteq \text{edge labelling}.$$

Indeed, to go from (partial) vertex labelling to edge labelling it is enough to define $\mathfrak{c}'(e) = \mathfrak{c}(\mathrm{In}(e))$: the colour of an edge is the colour of its incoming vertex. A path induces the same sequence of colours for both $\mathfrak{c}$ and $\mathfrak{c}'$, implying a strong equivalence between the two games.

Conversely, we can reduce from edge labelling to partial vertex labelling:

Lemma 1 (From labelling edges to labelling vertices)**.** *Let G a graph with n vertices and m edges, $\mathfrak{c} : E \to C$ an edge colouring function, we define a graph G' with $n+m$ vertices and $2m$ edges and a* partial *vertex colouring function $\mathfrak{c}' : V' \to C$ such that there is a one to one correspondence between infinite paths in G coloured by $\mathfrak{c}$ and infinite paths in G' coloured by $\mathfrak{c}'$.*

As a consequence, an algorithm using edge colouring functions of complexity $T(n,m)$ induces an algorithm for partial vertex colouring functions of complexity $T(n+m, 2m)$.

Proof. We add edges as intermediate vertices. Both the sets of vertices and of edges of G' are (disjoint copies of) $V \cup E$. For each edge $e = v \to v'$ of G, we add two edges in G': one from v to e, and one from e to v'. The colouring function $\mathfrak{c}'$ is defined only on E by $\mathfrak{c}'(e) = \mathfrak{c}(e)$. The construction is illustrated in Figure 1.2. Infinite paths in G and in G' are in one to one correspondence, and the sequences of colours are the same, since colours appear exactly every second position in every infinite path of G'. □

Figure 1.2: Reduction from edge colouring to vertex colouring.

There are no easy reductions from partial vertex labelling to vertex labelling, but in most cases an ad-hoc simple trick proves the two conventions to be (essentially) equivalently. For instance, of the objective contains a neutral letter, then the construction above can be replicated: for an objective $\Omega \subseteq C^\omega$, we say that $\varepsilon \in C$ is a neutral letter if for all $\rho \in C^\omega$, let ρ_ε the sequence obtained from ρ by removing all ε, then $\rho \in \Omega$ if and only if ρ_ε is finite or in Ω.

Winning in qualitative games

Now that we have the definitions of a game we can ask the main question: given a game $\mathcal{G}$ and a vertex v, who wins $\mathcal{G}$ from v?

Let $\mathcal{G}$ be a qualitative game and v a vertex. A strategy σ for Eve is called *winning* from v if every play starting from v consistent with σ is winning, *i.e.* satisfies W. Another common terminology is that σ ensures W. In that case we say that Eve has a winning strategy from v in $\mathcal{G}$, or equivalently that Eve wins $\mathcal{G}$ from v. This vocabulary also applies to Adam: for instance a strategy τ for Adam is called *winning* from v if every play starting from v consistent with τ is losing, *i.e.* does not satisfy W.

We let $W_{\text{Eve}}(\mathcal{G})$ denote the set of vertices v such that Eve wins $\mathcal{G}$ from v, it is called *winning region*, or sometimes winning set. A vertex in $W_{\text{Eve}}(\mathcal{G})$ is said winning for Eve. The analogous notation for Adam is $W_{\text{Adam}}(\mathcal{G})$.

We say that a strategy is *optimal* if it is winning from all vertices in $W_{\text{Eve}}(\mathcal{G})$.

Fact 1 (Winning regions are disjoint)**.** *For all qualitative games $\mathcal{G}$ we have $W_{\text{Eve}}(\mathcal{G}) \cap W_{\text{Adam}}(\mathcal{G}) = \emptyset$.*

Proof. Assume for the sake of contradiction that both players have a winning strategy from v, then $\pi^v_{\sigma,\tau}$ would both satisfy W and not satisfy W, a contradiction. □

It is however not clear that for every vertex v, *some* player has a winning strategy from v, which symbolically reads $W_{\text{Eve}}(\mathcal{G}) \cup W_{\text{Adam}}(\mathcal{G}) = V$. One might imagine that if Eve picks a strategy, then Adam can produce a counter strategy beating Eve's strategy, and vice versa, if Adam picks a strategy, then Eve can come up with a strategy winning against Adam's strategy. A typical example would be rock-paper-scissors (note that this is a concurrent game, meaning the two players play simultaneously, hence it does not match the definitions given so far), where neither player has a winning strategy.

Whenever $W_{\text{Eve}}(\mathcal{G}) \cup W_{\text{Adam}}(\mathcal{G}) = V$, we say that the game is *determined*. Being determined can be understood as follows: the outcome can be determined before playing assuming both players play optimally, since one of them can ensure to win whatever is the strategy of the opponent.

Theorem 1 (Borel determinacy)**.** *Qualitative games with Borel conditions are determined.*

The definition of Borel sets goes beyond the scope of this book. Suffice to say that all conditions studied in this book are (very simple) examples of Borel sets, implying that our qualitative games are all determined (as long as we consider perfect information and turn based games, the situation will change with more general models of games).

Computational problems for qualitative games

We identify three computational problems. The first is that of *solving a game*, which is the simplest one and since it induces a decision problem, allows us to make complexity theoretic statements.

Problem 1 (Solving the game)**.**

INPUT: *A qualitative game $\mathcal{G}$ and a vertex v*

OUTPUT: *Does Eve win $\mathcal{G}$ from v?*

The second problem extends the previous one: most algorithms solve games for all vertices at once instead of only for the given initial vertex. This is called *computing the winning regions*.

Problem 2 (Computing the winning regions)**.**

INPUT: *A qualitative game $\mathcal{G}$*

OUTPUT: *Compute $W_{\text{Eve}}(\mathcal{G})$ and $W_{\text{Adam}}(\mathcal{G})$*

We did not specify how the winning regions or the winning strategies are represented, this will depend on the types of games we consider. The third problem is *constructing a winning strategy*.

Problem 3 (Constructing a winning strategy)**.**

INPUT: *A qualitative game $\mathcal{G}$ and a vertex v*

OUTPUT: *Construct a winning strategy for Eve from v*

Values in quantitative games

Let $\mathcal{G}$ be a quantitative game and v a vertex. Given $x \in \mathsf{R}$ called a threshold, we say that a strategy σ for Max *ensures* x from v if every play π starting from v consistent with σ has value at least x under f, *i.e.* $f(\pi) \geq x$. In that case we say that Max has a strategy ensuring x in $\mathcal{G}$ from v.

Note that by doing so we are actually considering a qualitative game in disguise, where the qualitative condition is the set of plays having value at least x under f. Formally, a quantitative condition f and a threshold x induce a qualitative condition

$$f_{\geq x} = \{\pi \in \text{Paths}_\omega \mid f(\pi) \geq x\}.$$

Analogously, we say that a strategy τ for Min *ensures* x from v if every play π starting from v consistent with τ has value at most x under f, *i.e.* $f(\pi) \leq x$.

We let $\text{val}^{\mathcal{G}}_{\text{Max}}(v)$ denote the quantity

$$\sup_{\sigma} \inf_{\tau} f(\pi^v_{\sigma,\tau}),$$

where σ ranges over all strategies of Max and τ over all strategies of Min. We also write $\text{val}^{\sigma}_{\text{Max}}(v) = \inf_\tau f(\pi^v_{\sigma,\tau})$ so that $\text{val}^{\mathcal{G}}_{\text{Max}}(v) = \sup_\sigma \text{val}^{\sigma}_{\text{Max}}(v)$. This is called the value of Max in the game $\mathcal{G}$ from v, and represents the best outcome that she can ensure against any strategy of Min. Note that $\text{val}^{\mathcal{G}}_{\text{Max}}(v)$ is either a real number, ∞, or $-\infty$.

A strategy σ such that $\text{val}^{\sigma}_{\text{Max}}(v) = \text{val}^{\mathcal{G}}_{\text{Max}}(v)$ is called *optimal from* v, and it is simply *optimal* if the equality holds for all vertices. Equivalently, σ is optimal from v if for every play π consistent with σ starting from v we have $f(\pi) \geq \text{val}^{\mathcal{G}}_{\text{Max}}(v)$.

There may not exist optimal strategies which is why we introduce the following notion. For $\varepsilon > 0$, a strategy σ such that $\mathrm{val}^{\sigma}_{\mathrm{Max}}(v) \geq \mathrm{val}^{\mathscr{G}}_{\mathrm{Max}}(v) - \varepsilon$ is called *ε-optimal*. If $\mathrm{val}^{\mathscr{G}}_{\mathrm{Max}}(v)$ is finite there exist ε-optimal strategies for any $\varepsilon > 0$.

Symmetrically, we let $\mathrm{val}^{\mathscr{G}}_{\mathrm{Min}}(v)$ denote

$$\inf_{\tau} \sup_{\sigma} f(\pi^{v}_{\sigma,\tau}).$$

Fact 2 (Comparison of values for Min and Max)**.** *For all quantitative games $\mathscr{G}$ and vertex v we have $val^{\mathscr{G}}_{\mathrm{Max}}(v) \leq val^{\mathscr{G}}_{\mathrm{Min}}(v)$.*

Proof. For any function $F : X \times Y \to \mathsf{R} \cup \{\pm\infty\}$, we have

$$\sup_{x \in X} \inf_{y \in Y} F(x,y) \leq \inf_{y \in Y} \sup_{x \in X} F(x,y).$$

□

If this inequality is an equality, meaning

$$\mathrm{val}^{\mathscr{G}}_{\mathrm{Max}}(v) = \mathrm{val}^{\mathscr{G}}_{\mathrm{Min}}(v),$$

we say that the game $\mathscr{G}$ is *determined* in v, and let $\mathrm{val}^{\mathscr{G}}(v)$ denote the value in the game $\mathscr{G}$ from v. Similarly as for the qualitative case, being determined can be understood as follows: the outcome can be determined before playing assuming both players play optimally, and in that case the outcome is the value.

We say that a quantitative objective $f : C^{\omega} \to \mathsf{R} \cup \{\pm\infty\}$ is Borel if for all $x \in \mathsf{R}$, the qualitative objective $f_{\geq x} \subseteq C^{\omega}$ is a Borel set.

Corollary 1 (Borel determinacy for quantitative games)**.** *Quantitative games with Borel conditions are determined, meaning that for all quantitative games $\mathscr{G}$ we have $val^{\mathscr{G}}_{\mathrm{Max}} = val^{\mathscr{G}}_{\mathrm{Min}}$.*

Proof. If $\mathrm{val}^{\mathscr{G}}_{\mathrm{Max}}(v) = \infty$ then thanks to the inequality above $\mathrm{val}^{\mathscr{G}}_{\mathrm{Min}}(v) = \infty$ and the equality holds. Assume $\mathrm{val}^{\mathscr{G}}_{\mathrm{Max}}(v) = -\infty$ and let r be a real number. (The argument is actually the same as for the finite case but for the sake of clarity we treat them independently.) We consider $f_{\geq r}$. By definition, this a qualitative Borel condition, so Theorem 1 implies that it is determined. Since Max cannot have a winning strategy for $f_{\geq r}$, as this would contradict the definition of $\mathrm{val}^{\mathscr{G}}_{\mathrm{Max}}(v)$, this implies that Min has a winning strategy for $f_{\geq r}$, meaning a strategy τ such that every play π starting from v consistent with τ satisfy $f(\pi) < r$. In other words, $\mathrm{val}^{\tau}_{\mathrm{Min}}(v) = \sup_{\sigma} f(\pi^{v}_{\sigma,\tau}) \leq r$, which implies that $\mathrm{val}^{\mathscr{G}}_{\mathrm{Min}}(v) \leq r$. Since this is true for any real number r, this implies $\mathrm{val}^{\mathscr{G}}_{\mathrm{Min}}(v) = -\infty$.

Let us now assume that $x = \mathrm{val}^{\mathscr{G}}_{\mathrm{Max}}(v)$ is finite and let $\varepsilon > 0$. We consider $f_{\geq x+\varepsilon}$. By definition, this a qualitative Borel condition, so Theorem 1 implies that it is determined. Since Max cannot have a winning strategy for $f_{\geq x+\varepsilon}$, as this would contradict the definition of $\mathrm{val}^{\mathscr{G}}_{\mathrm{Max}}(v)$, this implies that Min has a winning strategy for $f_{\geq x+\varepsilon}$, meaning a strategy τ such that every play π starting from v consistent with τ satisfy $f(\pi) < x+\varepsilon$. In other words, $\mathrm{val}^{\tau}_{\mathrm{Min}}(v) = \sup_{\sigma} f(\pi^{v}_{\sigma,\tau}) \leq x+\varepsilon$, which implies that $\mathrm{val}^{\mathscr{G}}_{\mathrm{Min}}(v) \leq x+\varepsilon$. Since this is true for any $\varepsilon > 0$, this implies $\mathrm{val}^{\mathscr{G}}_{\mathrm{Min}}(v) \leq \mathrm{val}^{\mathscr{G}}_{\mathrm{Max}}(v)$. As we have seen the converse inequality holds, implying the equality. □

Note that this determinacy result does not imply the existence of optimal strategies.

Computational problems for quantitative games

As for qualitative games, we identify different computational problems. The first is solving the game.

Problem 4 (Solving the game)**.**
INPUT: *A quantitative game $\mathcal{G}$, a vertex v, and a threshold $x \in \mathbb{Q} \cup \{\pm\infty\}$*

OUTPUT: *Does Max have a strategy ensuring x from v?*

A very close problem is the *value problem.*

Problem 5 (Solving the value problem)**.**
INPUT: *A quantitative game $\mathcal{G}$, a vertex v, and a threshold $x \in \mathbb{Q} \cup \{\pm\infty\}$*

OUTPUT: *Is it true that $val^{\mathcal{G}}(v) \geq x$?*

The two problems of solving a game and the value problem are not quite equivalent: they become equivalent if we assume the existence of optimal strategies.

The value problem is directly related to *computing the value*.

Problem 6 (Computing the value)**.**
INPUT: *A quantitative game $\mathcal{G}$ and a vertex v*

OUTPUT: *Compute $val^{\mathcal{G}}(v)$*

What computing the value means may become unclear if the value is not a rational number, making its representation complicated. Especially in this case, it may be enough to approximate the value, which is indeed what the value problem gives us: by repeatedly applying an algorithm solving the value problem one can approximate the value to any given precision, using a binary search.

Lemma 2 (Binary search for computing the value)**.** *If there exists an algorithm A for solving the value problem of a class of games, then there exists an algorithm for approximating the value of games in this class within precision ε using $\log(\frac{1}{\varepsilon})$ calls to the algorithm A.*

The following problem is global, in the same way as computing the winning regions.

Problem 7 (Computing the value function)**.**
INPUT: *A quantitative game $\mathcal{G}$*

OUTPUT: *Compute the value function $val^{\mathcal{G}} : V \to \mathsf{R} \cup \{\pm\infty\}$*

Finally, we are sometimes interested in constructing optimal strategies provided they exist.

Problem 8 (Constructing an optimal strategy)**.**
INPUT: *A quantitative game $\mathcal{G}$ and a vertex v*

OUTPUT: *Compute an optimal strategy from v*

A close variant is to construct ε-optimal strategies, usually with ε given as input.

Prefix independent objectives

A qualitative objective Ω is:

- *closed under adding prefixes* if for every finite sequence ρ and for every infinite sequence ρ', if $\rho' \in \Omega$ then $\rho\rho' \in \Omega$;
- *closed under removing prefixes* if for every finite sequence ρ and for every infinite sequence ρ', if $\rho\rho' \in \Omega$ then $\rho' \in \Omega$;
- *prefix independent* if it is closed under both adding and removing prefixes; in other words whether a sequence satisfies Ω does not depend upon finite prefixes.

Let π be a finite play consistent with σ, we write $\sigma_{|\pi}$ for the strategy defined by

$$\sigma_{|\pi}(\pi') = \sigma(\pi\pi').$$

Fact 3 (Winning for objectives closed under removing prefixes)**.** *Let $\mathcal{G}$ be a qualitative game with objective Ω closed under removing prefixes, σ a winning strategy from v, and π a finite play consistent with σ starting from v. Then $\sigma_{|\pi}$ is winning from $v' = last(\pi)$.*

Proof. Let π' be an infinite play consistent with $\sigma_{|\pi}$ from v', then $\pi\pi'$ is an infinite play consistent with σ starting from v, implying that it is winning, and since Ω is closed under removing prefixes the play π' is winning. Thus $\sigma_{|\pi}$ is winning from v'. □

Corollary 2 (Reachable vertices of a winning strategy for objectives closed under removing prefixes)**.** *Let $\mathcal{G}$ be a qualitative game with objective Ω closed under removing prefixes and σ a winning strategy from v. Then all vertices reachable from v by a play consistent with σ are winning.*

In other words, when playing a winning strategy the play does not leave the winning region.

Similarly, a quantitative objective Φ is:

- *monotonic under adding prefixes* if for every finite sequence ρ and for every infinite sequence ρ' we have $\Phi(\rho') \leq \Phi(\rho\rho')$;
- *monotonic under removing prefixes* if for every finite sequence ρ and for every infinite sequence ρ' we have $\Phi(\rho') \geq \Phi(\rho\rho')$;
- *prefix independent* if it is monotonic under both adding and removing prefixes.

The fact above extends to quantitative objectives with the same proof.

Fact 4 (Winning for objectives monotonic under removing prefixes)**.** *Let $\mathcal{G}$ be a quantitative game with objective Φ monotonic under removing prefixes, σ a strategy ensuring x from v, and π a finite play consistent with σ starting from v. Then $\sigma_{|\pi}$ ensures x from $v' = last(\pi)$.*

Proof. Let π' be an infinite play consistent with $\sigma_{|\pi}$ from v', then $\pi\pi'$ is an infinite play consistent with σ starting from v, implying that $\Phi(\pi\pi') \geq x$, and since Φ is monotonic under removing prefixes this implies that $\Phi(\pi') \geq x$. Thus $\sigma_{|\pi}$ ensures x from v'. □

Corollary 3 (Comparison of values along a play)**.** *Let $\mathscr{G}$ be a quantitative game with objective Φ monotonic under removing prefixes and σ an optimal strategy from v. Then for all vertices v' reachable from v by a play consistent with σ we have $val^{\mathscr{G}}(v) \leq val^{\mathscr{G}}(v')$.*

In other words, when playing an optimal strategy the value is non-decreasing along the play.

1.2 Computational models

The Random Access Machine model of computation

For complexity statements we consider the classical Turing model of computation. However for algorithmic results the Turing model is a bit painful and unnatural hence it is customary to use the Random Access Machine (RAM) model instead. Intuitively this corresponds to using a standard imperative programming language on a usual computer which can create, access, and update variables. There are variants of the RAM model; to be specific the one we use and describe here is called '*word RAM*'. The main reason to use the RAM model is to make our life easier by hiding some small computational costs which are inessential for our purposes.

The memory is arranged in *machine words* whose size is a parameter w to be fixed depending on the problem. A machine word is a register which stores some information as a binary word of length w. The first key assumption of the RAM model is that *memory can be accessed in constant time*. In other words, machine words are registers with a unique address and can be accessed either directly or indirectly. A concrete implication is that checking whether an element belongs to a set is a single operation (if each element of the set can be stored in a single machine word).

We consider an (often implicit) set of basic operations operating on a constant number of machine words; addition, multiplication, subtraction, division, and comparison of integers are typical examples. The second key assumption is the '*unit cost model*', it says that the time complexity (also called cost) of basic operations is constant. This convention implies that we can manipulate counters for small numbers with no additional complexity, this will be useful in many situations.

We note that this is unrealistic as it means that for instance we can compute the number 2^{2^n} by repeatedly squaring 2: the complexity is $O(n)$ but this number uses $O(2^n)$ bits hence cannot be generated in polynomial time using a Turing machine. We will not make use of such weaknesses in our algorithms.

The size of an input is the number of machine words required to store it. The most common choice for the machine word size is $w = \log(s)$ where s is the size of the input

as we want to at least be able to store an integer x of order s in one machine word. However in situations in which there are numerical inputs, it is reasonable to assume that each input number fits into one machine word, leading to a potentially larger w. Note that an algorithm in the word RAM model with machine word size w is allowed to use numbers that are larger than 2^w, but such numbers should be split among several machine words.

The time complexity is the number of steps performed by the machine, as a function of the input size, and the space complexity is the maximal number of machine words used throughout the computation.

Games representations in the RAM model

The important parameters for algorithms on games are n the number of vertices and m the number of edges. Note that our assumption that every vertex has an outgoing edge implies that $n \leq m$.

The machine word size is always at least $w = \log(m)$, so that both a vertex and an edge can be stored in one machine word. A graph is given by the list of vertices and the list of edges, which implies that its size is $O(n+m) = O(m)$. An algorithm can go through all vertices, all edges, or all successors or predecessors of a given vertex, with no additional cost for the space complexity.

An arena additionally specifies for each vertex which player controls the vertex, which is a boolean value also stored in one machine word. The representation of conditions and colouring functions is different for each and is discussed when introducing them.

Polynomial versus strongly polynomial time algorithms

Let us consider a computational problem in which the input consists of a sequence of N integers plus a number n of other input bits. We write L for the total number of bits needed to encode the input integer numbers. We say that an algorithm runs in *strongly polynomial time* if:

- the number of arithmetic operations is bounded by a polynomial in the number of integers N in the input instance; and
- the space used by the algorithm is bounded by a polynomial in the size $L+n$ of the input.

An equivalent definition using the unit cost word RAM model is that the algorithm uses machine word size $w = L + \log(n)$ and runs in polynomial time.

Linear programming

We give here only the very essential definitions and results related to linear programming, and refer to [BT97] for a reference book on the topic.

A *linear program* (in canonical form) uses a set of real variables organised in a vector x and is defined by

$$\begin{array}{ll} \text{maximise} & c^T x \\ \text{subject to} & Ax \leq b, \end{array}$$

where c and b are rational vectors (with c^T the transpose of c) and A is a rational matrix. More explicitly:

$$\begin{array}{ll} \text{maximize} & c_1 x_1 + \ldots + c_n x_n \\ \text{subject to} & a_{11} x_1 + \ldots + a_{1n} x_n \leq b_1 \\ & \vdots \\ & a_{m1} x_1 + \ldots + a_{mn} x_n \leq b_m. \end{array}$$

Solving a linear program is finding an optimal assignment x^* of the variables.

Theorem 2 (Linear programming)**.** *There exists a polynomial time algorithm for solving linear programs.*

Note that we define here linear programs with a maximising objective, but the same problem with minimising $c^T x$ can be easily shown to be equivalent.

The statement above says that there exists a weakly polynomial time algorithm; whether there exists a strongly polynomial algorithm for linear programming is a long standing open question.

1.3 Objectives

We present in this section the main objectives and their representations. An objective may depend upon a set of parameters which are sometimes omitted when clear from the context.

Let us recall how we define classes of conditions: we first define an objective, for instance $\texttt{Safe} \subseteq \{\text{Win}, \text{Lose}\}^\omega$. For an arena and a colouring function $\mathtt{c} : E \to \{\text{Win}, \text{Lose}\}$ defined over this arena this induces the safety condition $\texttt{Safe}(\mathtt{c})$. Given an arena and a condition W over this arena we say that W is *a* safety condition if there exists $\mathtt{c}$ such that $W = \texttt{Safe}(\mathtt{c})$. The same terminology is used for all other objectives.

Prefix dependent qualitative objectives: safety and reachability

The *safety objective* is the simplest qualitative objective: the set of colours is $\{\text{Win}, \text{Lose}\}$, and safety requires that the colour Lose is never seen. Formally:

$$\texttt{Safe} = \left\{ \rho \in \{\text{Win}, \text{Lose}\}^\omega : \forall i, \rho_i \neq \text{Lose} \right\}.$$

In the example represented in Figure 1.3, a play is winning if if it never visits v_7. Eve wins from the four vertices on the left and loses from all the others, which is represented by the two dotted areas.

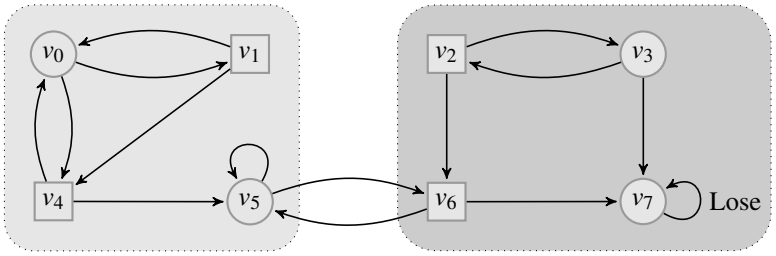


Figure 1.3: An example of a safety game. The unlabelled edges are implicitly labelled by Win. The condition of Eve is to avoid the colour Lose, hence equivalently here to avoid vertex v_7. The two dotted areas represent the winning regions of each player.

The dual of the safety objective is the *reachability objective*: the set of colours is $\{\text{Win}, \text{Lose}\}$, and reachability requires that the colour Win is seen at least once. Formally:

$$\texttt{Reach} = \left\{\rho \in \{\text{Win}, \text{Lose}\}^\omega : \exists i, \rho_i = \text{Win}\right\}.$$

Safety and reachability conditions are dual:

$$\text{Paths}_\omega \setminus \texttt{Safe}(\mathfrak{c}) = \texttt{Reach}(\bar{\mathfrak{c}}) \quad ; \quad \text{Paths}_\omega \setminus \texttt{Reach}(\mathfrak{c}) = \texttt{Safe}(\bar{\mathfrak{c}}).$$

where $\bar{\mathfrak{c}}$ swaps Win and Lose in $\mathfrak{c}$. Consequently, if the condition for Eve is a safety condition, then the condition for Adam is a reachability condition, and conversely.

Remark 3 (Labelling vertices for safety and reachability conditions)**.** *Safety and reachability conditions are most of the time defined on vertices rather than edges, as indeed it is more natural to set as target a vertex or a set of vertices than a set of edges. However in this book we use the more general convention of labelling edges. As illustrated in Lemma 1, labelling vertices is a special case of labelling edges. In Figure 1.3, the condition of Eve is to avoid the colour Lose, and since this is v_7 is a self loop with colour Lose, the condition can be equivalently stated as avoiding vertex v_7.*

Prefix independent qualitative objectives: Büchi and Parity

Safety and reachability objectives specify which colours occur or not, hence are prefix dependent. We now introduce objectives specifying which colours occur infinitely many times, which will naturally be prefix independent.

The *Büchi objective* is over the set of colours $\{1,2\}$, it requires that the colour 2 is seen infinitely many times. Formally:

$$\texttt{Buchi} = \left\{\rho \in \{1,2\}^\omega : \forall j, \exists i \geq j, \rho_i = 2\right\}.$$

The dual of the Büchi objective is the *CoBüchi objective*: the set of colours is $\{2,3\}$, it requires that the colour 3 is seen finitely many times. Formally:

$$\texttt{CoBuchi} = \left\{\rho \in \{2,3\}^\omega : \exists j, \forall i \geq j, \rho_i \neq 3\right\}.$$

Büchi and CoBüchi conditions are dual:

$$\mathrm{Paths}_\omega \setminus \texttt{Buchi}(\mathfrak{c}) = \texttt{CoBuchi}(\mathfrak{c}+1) \quad ; \quad \mathrm{Paths}_\omega \setminus \texttt{CoBuchi}(\mathfrak{c}) = \texttt{Buchi}(\mathfrak{c}-1).$$

where $\mathfrak{c}+1$ adds one to $\mathfrak{c}$ (since $\mathfrak{c}$ maps vertices to $\{1,2\}$, $\mathfrak{c}+1$ maps vertices to $\{2,3\}$), and similarly for $\mathfrak{c}-1$. Consequently, if the condition for Eve is a Büchi condition, then the condition for Adam is a CoBüchi condition, and conversely.

We now define the parity objectives. Let $[i,j]$ be an interval with $i,j \in \mathbb{N}$ used as a parameter defining the range of priorities. The *parity objective* with parameters i,j uses the set of colours $[i,j]$, which are referred to as priorities, and is defined by

$$\texttt{Parity}([i,j]) = \left\{ \rho \in [i,j]^\omega \;\middle|\; \begin{array}{c} \text{the largest priority appearing} \\ \text{infinitely many times in } \rho \text{ is even} \end{array} \right\}.$$

We made the dependence in $[i,j]$ explicit by writing $\texttt{Parity}([i,j])$, but we will most of the time write Parity and assume that the range of priorities is $[1,d]$, so d is the number of priorities. There are two possible conventions for defining parity objectives: considering the largest priority appearing infinitely many times, or the smallest one. They are strictly equivalent (through the transformation $p \mapsto 2d-p$) but depending on the situation one can be technically more convenient than the other.

We illustrate the definition on two examples.

$$1\,2\,4\,7\,5\,7\,5\,3\,6\,3\,6\,3\,6\,3\,6\,\cdots \in \texttt{Parity},$$

because the two priorities which appear infinitely often are 3 and 6, and the largest one is 6, which is even.

$$2\,2\,2\,4\,1\,7\,5\,3\,3\,3\,3\,3\,3\,3\,3\,\cdots \notin \texttt{Parity},$$

because the only priority which appears infinitely often is 3 and it is odd.

Two remarks are in order. First, Büchi and CoBüchi objectives are parity objectives for the set of colours $[1,2]$ and $[2,3]$, respectively. Second, the parity conditions are self dual:

$$\mathrm{Paths}_\omega \setminus \texttt{Parity}([i,j])(\mathfrak{c}) = \texttt{Parity}([i+1,j+1])(\mathfrak{c}+1),$$

where $\mathfrak{c}+1$ adds one to $\mathfrak{c}$. Hence if the condition for Eve is a parity condition, then the condition for Adam is also a parity condition.

Figure 1.4 presents an example of a parity game. The priority of an edge is given by its label.

Conventions

Given an objective $\Omega \subseteq C^\omega$ we use a colouring function $\mathfrak{c} : E \to C$ to induce the condition $\Omega(\mathfrak{c})$. It is convenient to extend this notation to sets of colours, edges, and vertices as follows.

- A set of colours $F \subseteq C$ induces the colouring function $\mathfrak{c}_F : E \to \{\mathrm{Win}, \mathrm{Lose}\}$ defined by $\mathfrak{c}_F(X) = \mathrm{Win}$ if $c \in F$ and Lose otherwise.

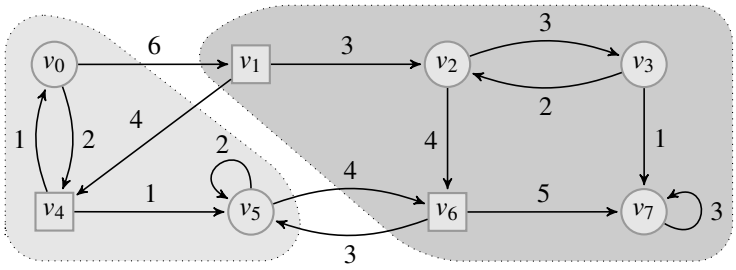


Figure 1.4: An example of a parity game. The two dotted areas represent the winning regions of each player.

- A set of edges $F \subseteq E$ induces the colouring function $\mathfrak{c}_F : E \to \{\text{Win}, \text{Lose}\}$ defined by $\mathfrak{c}_F(e) = \text{Win}$ if $e \in F$ and Lose otherwise.
- A set of vertices $F \subseteq V$ induces the colouring function $\mathfrak{c}_F : E \to \{\text{Win}, \text{Lose}\}$ defined by $\mathfrak{c}_F(e) = \text{Win}$ if $\text{In}(e) \in F$ and Lose otherwise.

As a convention, we write $\texttt{Reach}(F), \texttt{Safe}(F), \texttt{Buchi}(F)$, and $\texttt{CoBuchi}(F)$ as a shorthand for $\texttt{Reach}(\mathfrak{c}_F), \texttt{Safe}(\mathfrak{c}_F), \texttt{Buchi}(\mathfrak{c}_F)$, and $\texttt{CoBuchi}(\mathfrak{c}_F)$, respectively, for all three cases above.

Representations for qualitative objectives

For reachability, safety, Büchi, and CoBüchi conditions we need one bit per vertex to specify its colour, hence the machine word size $w = \log(m)$ implies that one machine word can store either an edge or a vertex together with this one bit of information.

The situation changes for parity conditions, where each vertex has a priority in $[1, d]$ hence requiring $\log(d)$ bits. We then choose the machine word size $w = \log(m) + \log(d)$ in such a way that one machine word can store either an edge or a vertex together with its priority.

Quantitative objectives

We define five quantitative objectives, which are variations of each other: shortest path, mean payoff, total payoff, energy, and discounted payoff. For all these objectives, the set of colours is (essentially) $C = \mathbb{Z}$, the set of integers. A colour is called a *weight*, interpreted as a cost. The cost of a finite path is the sum over the weights, which Max wants to maximise and Min to minimise. The problem is that mathematically, the cost of infinite paths may not be defined: the four definitions below can be seen as four different ways to tackle this mathematical obstacle.

Shortest path

The following definition resolves the problem above by adding a reachability objective, ensuring that paths of interest are finite. The *shortest path* quantitative objective is

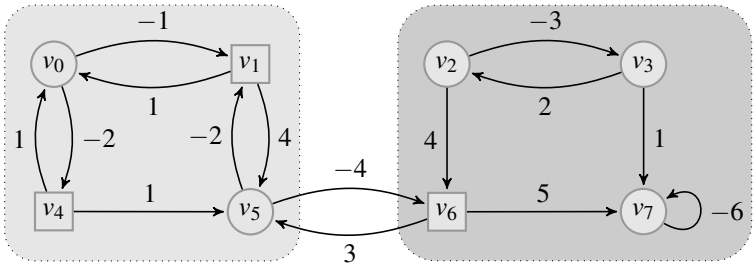


Figure 1.5: An example of a mean payoff game with threshold 0.

defined over the set of colours $C = \mathbb{Z} \cup \{\text{Win}\}$ by

$$\texttt{ShortestPath}(\rho) = \begin{cases} \sum_{i=0}^{k-1} \rho_i & \text{for } k \text{ the first index such that } \rho_k = \text{Win}, \\ \infty & \text{if } \rho_k \neq \text{Win for all } k. \end{cases}$$

Note that we are looking for a path of minimal cost, hence not necessarily the shortest in number of edges.

Mean payoff

The *mean payoff* quantitative objective weights paths by their length. It comes in two different flavours: using the supremum limit

$$\texttt{MeanPayoff}^{+}(\rho) = \limsup_k \frac{1}{k} \sum_{i=0}^{k-1} \rho_i.$$

or the infimum limit

$$\texttt{MeanPayoff}^{-}(\rho) = \liminf_k \frac{1}{k} \sum_{i=0}^{k-1} \rho_i.$$

As we shall see, in most settings the two objectives will be equivalent. For this reason, we often use `MeanPayoff` to denote $\texttt{MeanPayoff}^{-}$.

Figure 1.5 presents an example of a mean payoff game. The weight of an edge is given by its label. In this example the threshold is 0, *i.e.* the induced qualitative condition is $\texttt{MeanPayoff}_{\geq 0} = \{\rho \in C^\omega : \texttt{MeanPayoff}(\rho) \geq 0\}$.

Total payoff

The following definition simply replaces limit by supremum limit. We define the *total payoff* quantitative objective over the set of colours $C = \mathbb{Z}$ by

$$\texttt{TotalPayoff}(\rho) = \limsup_n \sum_{i=0}^{n-1} \rho_i$$

Total payoff is closely related to mean payoff, in the sense that it refines it. Indeed, the total payoff of a play is finite if and only if the mean payoff of this play is null. Hence, if a vertex in a game has value 0 for mean payoff, computing its value for total payoff gives an additional insight: it quantifies how the partial sums are fluctuating around the mean payoff. For instance, this allows one to distinguish situation $1,-1,1,-1,1,\ldots$ where the total payoff is 1 from a close situation $-1,1,-1,1,\ldots$ with total payoff 0.

Energy

The *energy* quantitative objective is defined over the set of colours $C = \mathbb{Z}$:

$$\texttt{Energy}(\rho) = \inf\left\{\ell \in \mathbb{N} : \forall k \in \mathbb{N},\ \ell + \sum_{i=0}^{k-1} \rho_i \geq 0\right\}.$$

The interpretation is the following: weights are energy consumptions (negative values) and recharges (positive values), and $\texttt{Energy}(\rho)$ is the smallest initial budget ℓ such that all partial sums remain non-negative. In a counter-intuitive way, the objective of Min is to collect higher (positive) weights.

We can observe that $\texttt{Energy}(\rho) = \sup\left\{\sum_{i=0}^{k-1}(-\rho_i) : k \in \mathbb{N}\right\}$, which motivates defining:

$$\texttt{Energy}^{+}(\rho) = \sup\left\{\sum_{i=0}^{k-1} \rho_i : k \in \mathbb{N}\right\}.$$

The two objectives are clearly equivalent since one can reduce from one to the other by switching the signs of every weight. The benefit of this definition is that it is intuitively simpler: Max aims at positive weights.

Discounted payoff

The *discounted payoff* quantitative objective is parameterised by a discount factor $\lambda \in (0,1)$. It is defined by:

$$\texttt{DiscountedPayoff}_\lambda(\rho) = \lim_k \sum_{i=0}^{k-1} \lambda^i \rho_i.$$

Expanding the definition: $\texttt{DiscountedPayoff}_\lambda(\rho) = \rho_0 + \lambda\rho_1 + \lambda^2\rho_2 + \ldots$. Intuitively, the importance of the weights decrease over time: the weight ρ_i is multiplied by λ^i which goes to 0 as i goes to infinity. The discount factor ensures that the limit exists for sequences with bounded weights, which holds for all plays since a (finite) game contains finitely many different weights.

Representations for quantitative objectives

Let us consider a game $\mathscr{G}$ with one of the payoff conditions defined above. Let W denote the largest weight appearing in $\mathscr{G}$ in absolute value.

Choosing the machine word size $w = \log(m) + \log(W)$ implies that either an edge or a vertex together with its weight can be stored in one machine word and that we can perform arithmetic operations on weights. For all payoff games except discounted games, this means that the input size is $O(m)$.

For discounted payoff games we additionally need to represent the discount factor λ, which we assume is a rational number $\lambda = \frac{a}{b}$. Since we want to perform arithmetic operations on λ it is convenient to store it on one machine word, hence the choice for the machine word size $w = \log(m) + \log(W) + \log(b)$.

1.4 Automata

The study of games is deeply intertwined with automata over infinite words and trees. We will not elaborate much on that aspect in this book, but in a few places we will use automata. We define here (non-deterministic) automata over infinite words, and refer to [Tho97] for a survey on automata theory over infinite objects (words and trees) and logic, and to [Pin21] for the most recent and complete textbook on automata theory.

Definition 2 (*Automata*)**.** *Let Σ be an alphabet. An automaton over the alphabet Σ is a tuple $\mathbf{A} = (Q, q_0, \Delta, A)$ where:*

- *Q is a finite set of states,*
- *$q_0 \in Q$ is the initial state,*
- *$\Delta \subseteq Q \times \Sigma \times Q$ is the transition relation,*
- *$A \subseteq \Delta^\omega$ is the acceptance condition.*

The size of $\mathbf{A}$ is its number of states, written $|\mathbf{A}|$. We assume that automata are complete: from any state q and letter a, there exists a transition $(q, a, q') \in \Delta$. This mirrors the convention for games that every vertex has an outgoing edge.

We use *transition based acceptance conditions* instead of state based acceptance conditions, for the same reasons as we use edge labelling colouring functions rather than vertex labelling. This more succinct definition of automata naturally composes with games see Section 1.6.

For a (finite or infinite) word $w = w_0 w_1 \dots$, a run $\rho = (q_0, w_0, q_1)(q_1, w_1, q_2)\dots$ over w is a sequence of consecutive transitions starting from the initial state q_0. An infinite run is accepting if it belongs to A, in which case we also say that it satisfies A. A word w is accepted if there exists an accepting run over w. We let $L(\mathbf{A})$ denote the set of accepted words and call it the language defined by $\mathbf{A}$, or sometimes recognised by $\mathbf{A}$.

An automaton is *deterministic* if for all states $q \in Q$ and letter $a \in \Sigma$, there exists at most one transition $(q, a, q') \in \Delta$. In that case the transition relation becomes a transition function $\delta : Q \times \Sigma \to Q$. The key property of deterministic automata is that for every word there exists exactly one run over it.

We use the same approach as for games for defining classes of automata with the same conditions: an objective $\Omega \subseteq C^\omega$ and a colouring function $\mathfrak{c} : \Delta \to C$ induce an acceptance condition $\Omega[\mathfrak{c}] \subseteq \Delta^\omega$. For deterministic automata the colouring function becomes $\mathfrak{c} : Q \times \Sigma \to C$. As for games the objective qualifies the automaton, so we speak of a parity automaton if it uses a parity objective.

Theorem 3 (Omega-regular languages)**.** *Non-deterministic Büchi, CoBüchi, parity, and deterministic parity automata define the same class of languages called* ω-regular languages.

1.5 Memory

A strategy can be a very complicated object, in particular it is infinite. Indeed, it is a function $\sigma : \text{Paths} \to E$, which means that in order to choose the next move the strategy considers everything played so far: the strategy depends upon the whole play.

An important part of the study of games is to prove that simple strategies suffice for many purposes, and one aspect that makes strategies simple is that they use little memory. For understanding a certain class of games a great insight is often to prove the existence of *simple* winning strategies, as for instance positional or using finite memory.

Positional strategies

Positional strategies carry no memory about the play constructed so far and in choosing an edge only look at the current vertex. The word memoryless is sometimes used in lieu of positional. Formally, a *positional strategy* for Eve is a function

$$\sigma : V_{\text{Eve}} \to E.$$

A positional strategy induces a strategy by $\widehat{\sigma}(\pi) = \sigma(\text{last}(\pi))$.

For reasoning about positional strategies it is useful to define the following notion. Let $\mathcal{G}$ be a game and σ a positional strategy, we define $\mathcal{G}[\sigma]$ the *graph with condition* W induced by σ on $\mathcal{G}$. The set of vertices is V and the set of edges is

$$E[\sigma] = \left\{ e = v \to v' \in E : v \in V_{\text{Adam}} \text{ or } (v \in V_{\text{Eve}} \text{ and } \sigma(v) = e) \right\}.$$

It is equipped with the condition W inherited from $\mathcal{G}$.

Fact 5 (Game induced by a positional strategy)**.** *Let $\mathcal{G}$ be a game with condition W, σ a positional strategy, and v a vertex. Then the strategy σ is winning from v if and only if all infinite paths in $\mathcal{G}[\sigma]$ from v satisfy W.*

We say that a qualitative objective Ω is *positionally determined* if for every game $\mathcal{G}$ with objective Ω and every vertex v, either of the two players has a positional winning strategy from v. We sometimes simply say that Ω is positional, and talk about a positional determinacy result.

We also say that a qualitative objective Ω is *half-positionally determined* if for every game $\mathcal{G}$ with objective Ω and every vertex v, if Eve has a winning strategy from

v, then she has a positional winning strategy from v. Similarly, we sometimes shorten to 'Ω is half-positional', and speak of a half-positional determinacy result.

Remark 4. *Another popular terminology replaces 'half-positionally determined' and 'positionally determined' by 'positionally determined' and 'bi-positionally determined'. The best argument for the terminology we adopt in this book is that in general determinacy is about both players, hence it naturally extends to positional determinacy.*

As we discussed earlier, the task of solving a game does not include constructing winning strategies. We present a general binary search technique for doing so assuming half-positional determinacy.

Lemma 3 (Binary search for constructing positional strategies)**.** *Let Ω be a half-positionally determined qualitative objective. If there exists an algorithm A for solving games with objective Ω, then there exists an algorithm for constructing winning strategies for Eve for games in this class using $n \cdot \log(\frac{m}{n})$ calls to the algorithm A.*

Proof. Let Ω be a half-positionally determined objective and $\mathscr{G}$ a qualitative game with objective Ω. We first determine $W_{\mathrm{Eve}}(\mathscr{G})$, which requires one call to a solving algorithm for each vertex. We fix a vertex $v \in W_{\mathrm{Eve}}(\mathscr{G})$ and determine a winning positional move from v. We let $d(v)$ denote the outdegree of v. We choose a subset of $\lfloor \frac{d(v)}{2} \rfloor$ outgoing edges of v, construct the game where we remove these edges, and solve it using v as initial vertex. If Eve wins this game from v, then there is a positional winning strategy that picks one of the remaining outgoing edges of v, otherwise we need to choose one of the removed edges. This binary search algorithm requires $O(\log(d(v)))$ calls to a solving algorithm for finding a winning positional move from v. Doing so for all vertices requires

$$O\left(\sum_{v \in V} \log d(v)\right) \leq n \cdot \log\left(\frac{m}{n}\right)$$

calls to a solving algorithm. □

Positional determinacy may hold only for some class of arenas (such that finite arenas or arenas with finite outdegree), in which case this is made explicit: *positionally determined over finite arenas*.

Parity objectives are positionally determined; this will be proved in Chapter 2. We illustrate it on Figure 1.6 by annotating Figure 1.4 with the positional winning strategies for both players.

We say that a quantitative objective Φ is *positionally determined* if for every game $\mathscr{G}$ with objective Φ and every vertex v, there exists a pair of positional optimal strategies from v. Let us state the quantitative counterpart of Lemma 3. The proof is the same.

Similarly, a quantitative objective Φ is *half-positionally determined* if for every game $\mathscr{G}$ with objective Φ and every vertex v, there exists a positional optimal strategy for Max from v.

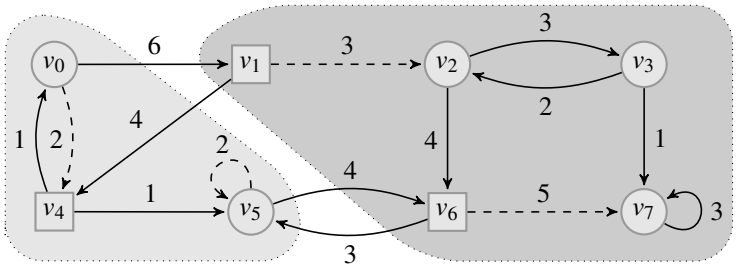


Figure 1.6: The example of a parity game given in Figure 1.4 with additional positional winning strategies for both players (corresponding to dashed edges).

Lemma 4 (Binary search for constructive winning strategies, quantitative case)**.** *Let Ω be a half-positionally determined quantitative objective. If there exists an algorithm A for computing the value of games with objective Ω, then there exists an algorithm for constructing optimal positional strategies for Max for games in this class using $n \cdot \log(\frac{m}{n})$ calls to A.*

Uniformity

A qualitative objective Ω is *uniformly positionally determined* if for every game $\mathscr{G}$ with objective Ω, there exists a pair (σ, τ) of positional optimal strategies: σ is winning from $W_{\text{Eve}}(\mathscr{G})$, meaning from every vertex in $W_{\text{Eve}}(\mathscr{G})$, and τ is winning from $W_{\text{Adam}}(\mathscr{G})$. Similarly, a quantitative objective Φ is *uniformly positionally determined* if for every game with objective Φ, there exists a pair (σ, τ) of positional strategies which are optimal from every vertex.

Being uniformly (half-)positionally determined is a stronger property than being (half-)positionally determined, but in most cases an objective satisfies either both or none, as for example if the objective is prefix independent.

Lemma 5 (From (half-)positional to uniformly (half-)positional prefix independent objectives)**.** *If an objective is (half-)positionally determined and prefix independent then it is uniformly (half-)positionally determined.*

Proof. Let us consider a game $\mathscr{G}$ with qualitative objective Ω and assume that Ω is half-positionally determined. (The argument is exactly the same for positionally determined, and for quantitative objectives, so we will not repeat it.) For each $v \in W_{\text{Eve}}(\mathscr{G})$ let σ_v be a positional winning strategy. Thanks to Fact 3 the strategy σ_v is winning from all vertices reachable by a play consistent with σ_v starting from v. Without loss of generality let us assume that σ_v is only defined on these vertices.

We fix $\leq$ a total order on the set of vertices[1]. We let σ be the positional strategy defined by $\sigma(u)$ is $\sigma_v(u)$ where v is the least vertex (with respect to $\leq$) such that σ_v is defined on u. We say that σ uses σ_v at u.

[1]The argument we give in this proof extends to infinite games whose set of vertices can be well ordered. A well-order is a total order such that every non-empty subset has a least element, which is exactly the property we need in this proof.

We argue that σ is winning from $W_{\text{Eve}}(\mathscr{G})$. Consider a play consistent with σ starting from some vertex in $W_{\text{Eve}}(\mathscr{G})$ and look at the sequence of strategies it uses. By definition this sequence is non-increasing (with respect to $\leq$), hence it is stationary. In other words the play is eventually consistent with some strategy σ_v, implying that this suffix satisfies Ω. Since Ω is prefix independent this means that the play itself satisfies Ω, so σ is indeed winning. □

Finite memory strategies

Memoryless strategies are sometimes not enough. A more powerful class of strategies is *finite memory* strategies. Intuitively, a finite memory strategy uses a finite state machine called a memory structure to store the relevant pieces of information about the play constructed so far.

To define finite memory strategies formally, we fix a graph G. A memory structure is $\mathscr{M} = (M, m_0, \delta)$: the set M is a set of (memory) states, $m_0 \in M$ is the initial state and $\delta : M \times E \to M$ is the update function. The update function is extended to $\delta^* : E^* \to M$ by $\delta^*(\varepsilon) = m_0$ and $\delta^*(\rho e) = \delta(\delta^*(\rho), e)$. The size of a memory structure is its number of states. Note that a memory structure is a deterministic automaton over the alphabet E but without specifying the acceptance condition.

We define a strategy using $\mathscr{M}$ as a function

$$\sigma : V_{\text{Eve}} \times M \to E.$$

It induces a strategy $\widehat{\sigma}$ via $\widehat{\sigma}(\pi) = \sigma(\text{last}(\pi), \delta^*(\pi))$. A common abuse of notations is to write σ for $\widehat{\sigma}$.

We note that positional strategies correspond to strategies using the trivial memory structure consisting of only one state.

We say that a qualitative objective Ω is *determined with finite memory strategies* if for every game $\mathscr{G}$ and every vertex v, either of the two players has a finite-memory winning strategy. There are several variants of this definition covering cases where the memory is constant or bounded, and uniformly over all vertices or not.

We give in Figure 1.7 an example of a game where Eve has a winning strategy using two memory states but no positional winning strategy. The condition is $\texttt{Buchi}(\text{Win}_1) \wedge \texttt{Buchi}(\text{Win}_2)$, meaning that a play is winning if both Win_1 and Win_2 are visited infinitely many times. A positional strategy would either always choose the edge labelled Win_1 or the one labelled Win_2, hence does not satisfy the condition. Some memory is required to switch between the two.

Formally, let e_1 be the edge labelled by Win_1 and e_2 be the one labelled by Win_2. We let $\mathscr{M} = (\{m_1, m_2\}, m_1, \delta)$ defined by $\delta(m_1, e_1) = m_2$ and $\delta(m_2, e_2) = m_1$. In words, m_1 is the memory state saying that the last chosen edge was e_1 and m_2 correspondingly for e_2. We switch from m_1 to m_2 when traversing e_1 and conversely with e_2. Then we define $\sigma(v_0, m_1) = e_1$ and $\sigma(v_0, m_2) = e_2$ inducing the strategy $\widehat{\sigma}$ using $\mathscr{M}$.

Let us note that to transform this example into one where colours label vertices, we would need three states instead of one. Memory requirements for edge-labelled games and vertex-labelled games are not equivalent.

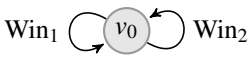


Figure 1.7: An example of a game where Eve has a winning strategy for $\texttt{Buchi}(\text{Win}_1) \wedge \texttt{Buchi}(\text{Win}_2)$ using two memory states but no positional winning strategy.

1.6 Reductions

Automata and memory structures can be used to construct reductions between games. Automata operate at the level of objectives, independently of the colouring function and the arena, while memory structures work at the level of conditions, hence depend on the graph.

Reductions between objectives using automata

Let Ω a qualitative objective over the set of colours C, and Ω' a second qualitative objective. We say that Ω *reduces to* Ω' if there exists a *deterministic* automaton $\mathbf{A}$ over the alphabet C with acceptance objective Ω' defining Ω, *i.e.* such that $L(\mathbf{A}) = \Omega$.

This implies that we can transform a game $\mathscr{G}$ with objective Ω into an equivalent one $\mathscr{G} \times \mathbf{A}$ with objective Ω' by composing $\mathscr{G}$ with $\mathbf{A}$: the automaton reads the sequence of colours from C induced by the play and produces a new sequence of colours which is accepted if its satisfies Ω'.

Formally, let $\mathscr{A}$ an arena and

$$\mathbf{A} = (Q, q_0, \delta : Q \times C \to Q, \Omega'(\mathfrak{c}_{\mathbf{A}}))$$

a deterministic automaton with $\mathfrak{c}_{\mathbf{A}} : Q \times C \to C'$. We construct the arena $\mathscr{A} \times \mathbf{A}$ as follows. We first define the graph $G \times Q$ whose set of vertices is $V \times Q$ and set of edges is defined as follows: for every edge $e = v \xrightarrow{c} v' \in E$ and state $q \in Q$ there is an edge $e[q] = (v,q) \xrightarrow{\mathfrak{c}_{\mathbf{A}}(q,c)} (v', \delta(q,c))$: the second component computes the run of $\mathbf{A}$ on the sequence of colours induced by the play. The arena is $\mathscr{A} \times \mathbf{A} = (G \times Q, V_{\text{Eve}} \times Q, V_{\text{Adam}} \times Q)$. The game is $\mathscr{G} \times \mathbf{A} = (\mathscr{A} \times \mathbf{A}, \Omega'(\mathfrak{c}'))$.

The following lemma states two consequences to the fact that Ω reduces to Ω'.

Lemma 6 (Automata reductions)**.** *If Ω reduces to Ω' through the automaton $\mathbf{A}$, then Eve has a winning strategy in $\mathscr{G}$ from v_0 if and only if she has a winning strategy in $\mathscr{G} \times \mathbf{A}$ from (v_0, q_0).*

Consequently, the following properties hold:

- *Assume that there exists an algorithm for solving games with objective Ω' with complexity $T(n,m)$. Then there exists an algorithm for solving games with objectives Ω of complexity $T(n \cdot |\mathbf{A}|, m \cdot |\mathbf{A}|)$.*
- *If Ω' is determined with finite memory strategies of size m, then Ω is determined with finite memory strategies of size $m \cdot |\mathbf{A}|$.*

Since the next type of reduction extends this one and the two proofs are very similar we will prove this lemma as a corollary of the next one.

Reductions between objectives using automata are very general: they operate at the level of objectives and therefore completely ignore the arena.

Reductions between conditions using memory structures

Reductions between conditions using memory structures extend the previous ones, the main difference being that the memory structure reads the sequences of edges and produces a sequence of memory states. The edges contain more information than the sequence of colours (which is what the automaton reads), and this information is dependent on the graph.

Formally, let $\mathscr{A}$ an arena and $\mathscr{M}$ a memory structure. We construct the arena $\mathscr{A} \times \mathscr{M}$ as follows. We first define the graph $G \times M$ whose set of vertices is $V \times M$ and set of edges E_M is defined as follows: for every edge $e = v \to v' \in E$ and state $m \in M$ there is an edge $e[m] = (v,m) \to (v', \delta(m,e))$. The arena is $\mathscr{A} \times \mathscr{M} = (G \times M, V_{\text{Eve}} \times M, V_{\text{Adam}} \times M)$.

Fact 6 (Strategies with memory)**.** *There is a one to one correspondence between plays in $\mathscr{A}$ from v_0 and in $\mathscr{A} \times \mathscr{M}$ from (v_0, m_0).*

To the play

$$\pi = e_0 e_1 \dots \text{ where } e_i = v_i \to v_{i+1}$$

we associate the play

$$\pi' = e_0[m_0]\ e_1[m_1] \dots,$$

where for all i we define inductively $m_{i+1} = \delta(m_i, e_i[m_i])$.

Let W be a condition on $\mathscr{A}$ and W' a condition on $\mathscr{A} \times \mathscr{M}$. We say that W *reduces to* W' (through $\mathscr{M}$) if for all plays π in $\mathscr{A}$, we have

$$\pi \in W \iff \pi' \in W'.$$

Let $\mathscr{M}$ and $\mathscr{M}'$ two memory structure over the same graph, we let $\mathscr{M} \times \mathscr{M}'$ denote the memory structure obtained by direct product.

Lemma 7 (Memory structure reductions)**.** *If W reduces to W' through the memory structure $\mathscr{M}$, then Eve has a winning strategy in $\mathscr{G} = (\mathscr{A}, W)$ from v_0 if and only if she has a winning strategy in $\mathscr{G} \times \mathscr{M} = (\mathscr{A} \times \mathscr{M}, W')$ from (v_0, m_0).*

More specifically, if Eve has a winning strategy in $\mathscr{G} \times \mathscr{M}$ from (v, m_0) using $\mathscr{M}'$ as memory structure, then she has a winning strategy in $\mathscr{G}$ from v using $\mathscr{M} \times \mathscr{M}'$ as memory structure. In particular if the strategy in $\mathscr{G} \times \mathscr{M}$ is memoryless, then the strategy in $\mathscr{G}$ uses $\mathscr{M}$ as memory structure.

Proof. A winning strategy in $\mathscr{G}$ directly induces a winning strategy in $\mathscr{G} \times \mathscr{M}$ simply by ignoring the additional information and thanks to the equivalence above because W reduces to W'. For the converse implication, let σ be a winning strategy in $\mathscr{G} \times \mathscr{M}$ using $\mathscr{M}'$ as memory structure. Recall that σ is defined through the function $\sigma : (V_{\text{Eve}} \times$

$M) \times M' \to E_M$. Let $p : E_M \to E$ mapping the edge e_m to e. We construct a strategy σ' in $\mathcal{G}$ using $\mathcal{M} \times \mathcal{M}'$ as memory structure by

$$\sigma'(v,(m,m')) = p(\sigma((v,m),m')).$$

The correspondence between plays in $\mathcal{A}$ and $\mathcal{A} \times \mathcal{M}$ maps plays consistent with σ to plays consistent with σ', which together with the fact that W reduces to W' implies that σ' is a winning strategy in $\mathcal{G}$ from v. □

To obtain Lemma 6 as a corollary of Lemma 7 we observe that a reduction between objectives using an automaton induces a reduction between the induced conditions using a memory structure. Formally, let us assume that Ω reduces to Ω', and let $\mathbf{A} = (Q, q_0, \delta, \Omega'(\mathfrak{c}_\mathbf{A}))$ such that $L(\mathbf{A}) = \Omega$. Let $\mathcal{G} = (\mathcal{A}, \Omega(\mathfrak{c}))$ a game.

We first define the memory structure $\mathcal{M} = (Q, q_0, \delta')$. The transition function is $\delta' : Q \times E \to Q$, it is defined by $\delta'(q,e) = \delta(q, \mathfrak{c}(e))$. We consider the arena $\mathcal{A} \times \mathcal{M}$, and define the colouring function $\mathfrak{c}'(e_q) = \mathfrak{c}_\mathbf{A}(q, \mathfrak{c}(e))$.

We note that $\Omega(\mathfrak{c})$ reduces to $\Omega'(\mathfrak{c}')$: for all plays π in $\mathcal{A}$, we have $\pi \in \Omega(\mathfrak{c})$ if and only if $\pi' \in \Omega'(\mathfrak{c}')$: this is a reformulation of the fact that $L(\mathbf{A}) = \Omega$.

We construct the game $\mathcal{G}' = (\mathcal{A} \times \mathcal{M}, \Omega'(\mathfrak{c}'))$. Thanks to Lemma 7 the two games have the same winner and a strategy in the latter induce a strategy in the former by composing with the memory structure $\mathcal{M}$, implying Lemma 6.

1.7 Traps and subgames

Let us consider a game $\mathcal{G}$ and a set X of vertices. Assume that for every $v \in X$ there exists $v \to v' \in E$ such that $v' \in X$, then we can define the game $\mathcal{G}[X]$ by restricting $\mathcal{G}$ to the vertices in X and say that $\mathcal{G}[X]$ is the *subgame* of $\mathcal{G}$ induced by X. Formally, the arena is $\mathcal{A}[X]$ with X the set of vertices and $E[X]$ is the set of edges such that both incoming and outgoing vertices are in X. The assumption on X ensures that every vertex of $\mathcal{G}[X]$ has an outgoing edge. Both the colouring function and the conditions are naturally induced from $\mathcal{G}$ to $\mathcal{G}[X]$.

We say that X is a *trap* for Adam if

- for every $v \in X \cap V_{\text{Eve}}$, there exists $v \to v' \in E$ with $v' \in X$, and
- for every $v \in X \cap V_{\text{Adam}}$, for all $v \to v' \in E$, we have $v' \in X$.

Intuitively, a trap for Adam is a subset of vertices which Eve can decide to stay in while Adam cannot force to leave. The same notion can be defined for Eve. Traps satisfy the property above so if X is a trap then the game $\mathcal{G}[X]$ described above is well defined, meaning that every vertex has an outgoing edge.

Fact 7 (Traps induce subgames)**.** *Let $\mathcal{G}$ be a game, X a trap for Adam, and σ a winning strategy for Eve in the subgame $\mathcal{G}[X]$. Then σ induces a winning strategy in $\mathcal{G}$.*

Proof. Any play consistent with σ in $\mathcal{G}$ stays forever in X because X is a trap for Adam, hence is winning. □

The notion of traps is very useful when decomposing games. We present some simple facts about traps, here stated for Adam but easily transposed for Eve.

Fact 8 (Traps)**.** *Let $\mathcal{G}$ a game.*

- *Let P, Q two traps for Adam in the game $\mathcal{G}$, then P is a trap for Adam in the subgame of $\mathcal{G}$ induced by Q (but $P \cap Q$ may not be a trap in $\mathcal{G}$).*
- *Let P a trap for Eve in the game $\mathcal{G}$ and Q a trap for Adam in the game $\mathcal{G}$, then $P \cap Q$ is a trap for Eve in the subgame of $\mathcal{G}$ induced by Q.*
- *Let P a trap for Adam in the game $\mathcal{G}$ and Q a trap for Adam in the subgame of $\mathcal{G}$ induced by X, then Q is a trap for Adam in $\mathcal{G}$.*

1.8 Generic fixed point algorithms

Let X be a set and $\mathbb{O} : X \to X$ a function that we call an *operator*, we say that $x \in X$ is a *fixed point* of $\mathbb{O}$ if $\mathbb{O}(x) = x$. Fixed points will appear extensively in this book. We describe here two general approaches for computing them: Kleene and Banach fixed point theorems.

Kleene fixed point theorem

Let us consider a *lattice* $(X, \leq)$: the binary relation $\leq$ is a partial order, and every pair of elements has a least upper bound and a greatest lower bound. It is a *complete lattice* if every set has a least upper bound and a greatest lower bound. A lattice is finite if the set X is finite. Note that finite lattices are always complete.

We write $\bot$ for the least element in X and $\top$ for the greatest element. An operator $\mathbb{O} : X \to X$ is *monotonic* if for all $x, y \in X$ such that $x \leq y$ we have $\mathbb{O}(x) \leq \mathbb{O}(y)$, and *preserves suprema* if $\mathbb{O}(\sup_n x_n) = \sup_n \mathbb{O}(x_n)$ for all increasing sequences $(x_n)_{n \in \mathbb{N}}$. The twin notion *preserves infima* is defined accordingly.

We say that x is a *pre-fixed point* if $\mathbb{O}(x) \leq x$ and a *post-fixed point* if $\mathbb{O}(x) \geq x$.

Theorem 4 (Kleene fixed point theorem)**.** *Let $(X, \leq)$ be a complete lattice and $\mathbb{O} : X \to X$ a monotonic operator, then $\mathbb{O}$ has a least fixed point which is also the least pre-fixed point.*

Furthermore:

- *if X is finite the sequence defined by $u_0 = \bot$ and $u_{k+1} = \mathbb{O}(u_k)$ is stationary and its limit is the least fixed point of $\mathbb{O}$;*
- *if $\mathbb{O}$ preserves suprema then the least fixed point of $\mathbb{O}$ is $\sup\left\{\mathbb{O}^k(\bot) : k \in \mathbb{N}\right\}$.*

Under the same assumptions $\mathbb{O}$ has a greatest fixed point which is the greatest post-fixed point and can be computed in similar ways, replacing 'preserves suprema' by 'preserves infima'.

The typical example of a complete lattice and one that we will use often in this book is the powerset of a set equipped with the inclusion between subsets. The least

element is the empty set, the greatest element the full set, and least and greatest upper bounds are given by union and intersection. An example of an infinite complete lattice is $\mathsf{R} \cup \{\pm\infty\}$ equipped with the natural order.

Banach fixed point theorem

Let us consider a set X equipped with some norm $\|\cdot\|$. It is called a *complete space* if all Cauchy sequences converge. The typical example of a complete space is R^d for some $d \in \mathbb{N}$ equipped with the infinity norm $\|x\| = \max_{i\in[1,d]} |x_i|$.

An operator $\mathbb{O} : X \to X$ is *contracting* if there exists $\lambda < 1$ such that for all $x, y \in X$ we have $\|\mathbb{O}(x) - \mathbb{O}(y)\| \leq \lambda \cdot \|x - y\|$.

Theorem 5 (Banach fixed point theorem)**.** *Let $(X, \|\cdot\|)$ be a complete space and $\mathbb{O} : X \to X$ a contracting operator, then $\mathbb{O}$ has a unique fixed point x_*. For any $x_0 \in X$, the sequence $(\mathbb{O}^k(x_0))_{k\in\mathbb{N}}$ converges towards x_* and the rate of convergence is given by*

$$\|\mathbb{O}^k(x_0) - x_*\| \leq \frac{\lambda^k}{1-\lambda} \cdot \|\mathbb{O}(x_0) - x_0\|.$$

1.9 Value iteration algorithms

In this section and the next we discuss two families of fixed point algorithms for solving games. The goal is to highlight the main ingredients for constructing algorithms in these two families. If the descriptions below are too abstract it may be useful to see concrete instantiations, referenced below.

Quantitative games

Let us consider a quantitative game $\mathscr{G}$ with condition $f = \Phi[\mathfrak{c}]$. Assuming that $\mathscr{G}$ is determined it admits a value function

$$\mathrm{val}^{\mathscr{G}} : V \to \mathsf{R} \cup \{\pm\infty\},$$

which is defined as

$$\mathrm{val}^{\mathscr{G}} = \sup_{\sigma} \inf_{\tau} f(\pi^v_{\sigma,\tau}) = \inf_{\tau} \sup_{\sigma} f(\pi^v_{\sigma,\tau}),$$

where σ ranges over strategies of Max, τ over strategies of Min, and $\pi^v_{\sigma,\tau}$ is the play consistent with σ and τ from v. In particular we write $\mathrm{val}^{\sigma}(v)$ for $\inf_{\tau} f(\pi^v_{\sigma,\tau})$ and $\mathrm{val}^{\tau}(v)$ for $\sup_{\sigma} f(\pi^v_{\sigma,\tau})$.

Let us write $Y = \mathsf{R} \cup \{\pm\infty\}$, and note that Y is a total order (and a fortiori a lattice) for the usual order over the reals.

Qualitative games

For a qualitative game $\mathscr{G}$ there is no notion of a value function so the first step in constructing a value iteration algorithm is to define a meaningful notion of value function.

Let us assume that the condition is $\Omega[\mathfrak{c}]$ over the set of colours C. The first ingredient is a lattice $(Y, \leq)$ together with a function $f : C^\omega \to Y$ for evaluating plays. The value function is $\mathrm{val}^{\mathscr{G}} : V \to Y$ defined as

$$\mathrm{val}^{\mathscr{G}}(v) = \sup_{\sigma} \inf_{\tau} f(\pi^v_{\sigma,\tau}),$$

where σ ranges over strategies of Max, τ over strategies of Min, and $\pi^v_{\sigma,\tau}$ is the play consistent with σ and τ from v. As above we write $\mathrm{val}^{\sigma}(v)$ for $\inf_\tau f(\pi^v_{\sigma,\tau})$ and $\mathrm{val}^{\tau}(v)$ for $\sup_\sigma f(\pi^v_{\sigma,\tau})$.

We let $\top$ denote the largest element in Y, and $\bot$ for the smallest. The following principle implies that computing the value function in particular yields the winning regions, relating the condition Ω to the function f:

Property 1 (Characterisation of the winning regions)**.** *For all games $\mathscr{G}$, for all vertices v, Eve wins from v for the qualitative objective $\Omega[\mathfrak{c}]$ if and only if $\mathrm{val}^{\mathscr{G}}(v) \neq \top$.*

This implies that the goal is now to compute or approximate the value function $\mathrm{val}^{\mathscr{G}} : V \to Y$. Indeed choosing $Y = \mathsf{R} \cup \{\pm\infty\}$ covers the quantitative case.

Fixed point

Let us consider a game $\mathscr{G}$. We let F_V denote the set of functions $V \to Y$, it is a lattice when equipped with the componentwise (partial) order induced by Y: we say that $\mu \leq \mu'$ if for all vertices v we have $\mu(v) \leq \mu'(v)$. The main ingredient is an operator $\mathbb{O}^{\mathscr{G}} : F_V \to F_V$. The intent is to obtain the function $\mathrm{val}^{\mathscr{G}}$ as a fixed point of the operator $\mathbb{O}^{\mathscr{G}}$, using the two different approaches for fixed points we introduced above. Whenever $\mathscr{G}$ is clear from the context, we simply write $\mathbb{O}$ instead of $\mathbb{O}^{\mathscr{G}}$.

Fixed point through monotonicity

The first family of algorithms is based on Kleene fixed point theorem as stated in Theorem 4.

Property 2 (Fixed point through monotonicity)**.** *For all games $\mathscr{G}$, the operator $\mathbb{O}^{\mathscr{G}}$ is monotonic, and $\mathrm{val}^{\mathscr{G}}$ is the least fixed point of $\mathbb{O}^{\mathscr{G}}$.*

Remark 5 (Greatest versus least fixed point)**.** *Whenever technically convenient or more intuitive, we will obtain the value function as the greatest fixed point of $\mathbb{O}^{\mathscr{G}}$. For value iteration algorithms, only small adjustments are necessary. This will be different for strategy improvement algorithms, in the next section.*

Theorem 4 further states that $\mathrm{val}^{\mathscr{G}}$ is the limit of the sequence $(\mathbb{O}^k(\bot))_{k\in\mathbb{N}}$. The pseudocode is given in Algorithm 1.1.

Theorem 6 (Generic value iteration algorithm through monotonicity)**.** *Assume Property 2 (fixed point through monotonicity) and that Y is finite. Then the generic value iteration algorithm outputs $\mathrm{val}^{\mathscr{G}}$ within at most $n \cdot |Y|$ iterations.*

A concrete instantiation of this theorem is for energy games, see Section 4.3.

Algorithm 1.1: A generic value iteration algorithm based on fixed point through monotonicity – naive version.

for $v \in V$ **do**
  $\mu(v) \leftarrow \bot$
repeat
  $\mu \leftarrow \mathbb{O}^{\mathscr{G}}(\mu)$
until $\mu = \mathbb{O}^{\mathscr{G}}(\mu)$
return μ

Proof. If Y is a finite lattice then so is F_V. For each v, the sequence $(\mathbb{O}^k(\mu)(v))_{k\in\mathbb{N}}$ is non-decreasing, so it can be strictly decreased at most $|Y|$ times. At each iteration the value of at least one vertex is strictly decreased. Hence there are at most $n \cdot |Y|$ iterations. □

If Y is not finite then the sequence $(\mathbb{O}^k(\mu))_{k\in\mathbb{N}}$ converges towards $\mathrm{val}^{\mathscr{G}}$ but further analysis is required to evaluate the convergence speed.

Fixed point through contraction

The second family of value iteration algorithms is based on Banach fixed point theorem as stated in Theorem 5, let us fix as a goal to approximate $\mathrm{val}^{\mathscr{G}}$. We equip F_V with a norm $\|\cdot\|$.

Property 3 (Fixed point through contraction)**.** *The operator $\mathbb{O}^{\mathscr{G}}$ is contracting in the complete space $(F_V, \|\cdot\|)$, and $\mathrm{val}^{\mathscr{G}}$ is the unique fixed point of $\mathbb{O}^{\mathscr{G}}$.*

The pseudocode of the algorithm is given in Algorithm 1.2.

Algorithm 1.2: A generic value iteration algorithm based on fixed point through contraction – naive version.

Data: Desired precision ε.
Choose $\mu \in F_V$
repeat
  $\mu \leftarrow \mathbb{O}^{\mathscr{G}}(\mu)$
until $\|\mathbb{O}^{\mathscr{G}}(\mu) - \mu\| \leq \varepsilon \cdot (1-\lambda)$
return μ

Theorem 7 (Generic value iteration algorithm through contraction)**.** *Assume Property 3 (fixed point through contraction). Then the generic value iteration algorithm computes an ε-approximation of $\mathrm{val}^{\mathscr{G}}$ within at most $O\left(\frac{\log(\varepsilon)}{\log(\lambda)}\right)$ iterations, where $\lambda \in (0,1)$ is the contraction factor of $\mathbb{O}^{\mathscr{G}}$.*

A concrete instantiation of this theorem is for stochastic reachability games, see Section 6.3.

Proof. Thanks to Theorem 5, for any μ we have

$$\|\mathbb{O}^k(\mu) - \mathrm{val}^{\mathscr{G}}\| \leq \frac{\lambda^k}{1-\lambda} \cdot \|\mathbb{O}^{\mathscr{G}}(\mu) - \mu\|.$$

Let us write $\mu_k = \mathbb{O}^k(\mu_0)$, and set the number of iterations to be $k = \frac{\log\left(\frac{\varepsilon\cdot(1-\lambda)^2}{\|\mathbb{O}^{\mathscr{G}}(\mu_0)-\mu_0\|}\right)}{\log(\lambda)}$. It has been chosen so that

$$\frac{\lambda^k}{1-\lambda} \cdot \|\mathbb{O}^{\mathscr{G}}(\mu_0) - \mu_0\| \leq \varepsilon \cdot (1-\lambda).$$

Thanks to the inequality above for $\mu = \mu_0$, this implies that $\|\mathbb{O}^{\mathscr{G}}(\mu_k) - \mu_k\| \leq \varepsilon \cdot (1-\lambda)$. Again thanks to the inequality above for $\mu = \mu_k$ this implies that $\|\mu_k - \mathrm{val}^{\mathscr{G}}\| \leq \varepsilon$. Thus, after k iterations, the algorithm outputs an ε-approximation of $\mathrm{val}^{\mathscr{G}}$. □

Wrap up

Let us wrap up this section: to construct a value iteration algorithm, one needs:

- for qualitative games, a notion of values induced by a function $f : C^\omega \to Y$ satisfying Property 1;
- in all cases, either a monotonic operator satisfying Property 2 or a contracting operator satisfying Property 3.

Local operator

In the general case above where nothing is known about the operator $\mathbb{O}^{\mathscr{G}}$, we cannot hope for a finer analysis. However in most cases the operator is defined in the following way, called 'local'. Fix a game $\mathscr{G}$. Let us consider a function $\delta : Y \times C \to Y$, which can be though of as the transition function of a deterministic automaton whose set of states is Y and which reads colours. It induces an operator $\mathbb{O}^{\mathscr{G}} : F_V \to F_V$ defined by:

$$\mathbb{O}^{\mathscr{G}}(\mu)(u) = \begin{cases} \max\left\{\delta(\mu(v), c) : u \xrightarrow{c} v \in E\right\} & \text{if } u \in V_{\mathrm{Max}}, \\ \min\left\{\delta(\mu(v), c) : u \xrightarrow{c} v \in E\right\} & \text{if } u \in V_{\mathrm{Min}}. \end{cases}$$

A first interesting fact about this family of operators is that if δ is monotonic, meaning for all $y, y' \in Y$, $c \in C$, if $y \leq y'$ then $\delta(y,c) \leq \delta(y',c)$, then $\mathbb{O}^{\mathscr{G}}$ is monotonic. Let us formulate this assumption explicitly:

Property 4 (Monotonicity of the δ function)**.** *The function δ is monotonic.*

Let us examine the property that $\mathrm{val}^{\mathscr{G}}$ is a fixed point of $\mathbb{O}^{\mathscr{G}}$. We give sufficient conditions in the following lemma:

Lemma 8. *Let $\mathscr{G}$ a game. Assume that:*

- *For all $\rho \in C^\omega$ and $c \in C$, we have $f(c \cdot \rho) = \delta(f(\rho), c)$.*

- *The function δ is monotonic and continuous.*

Then $\mathrm{val}^{\mathscr{G}}$ is a fixed point of $\mathbb{O}^{\mathscr{G}}$.

Proof. We show that $\mathrm{val}^{\mathscr{G}} \geq \mathbb{O}^{\mathscr{G}}(\mathrm{val}^{\mathscr{G}})$. Let $u \in V_{\mathrm{Max}}$, we consider σ, τ two strategies from v. We write $\sigma(u) = u \xrightarrow{c} v$. We let σ', τ' denote the strategies induced by σ, τ after playing $u \xrightarrow{c} v$, $\pi_u^{\sigma,\tau}$ the play consistent with σ and τ from u, and $\pi_v^{\sigma',\tau'}$ the play consistent with σ' and τ' from v. The first property implies that $f(\pi_u^{\sigma,\tau}) = \delta(f(\pi_v^{\sigma',\tau'}), c)$. Carefully using infimum and supremum as well as monotonicity and continuity of δ, we obtain the series of inequalities:

$$\begin{array}{lcl}
f(\pi_u^{\sigma,\tau}) & = & \delta(f(\pi_v^{\sigma',\tau'}), c) \\
f(\pi_u^{\sigma,\tau}) & \geq & \inf_{\tau'} \delta(f(\pi_v^{\sigma',\tau'}), c) = \delta(\inf_{\tau'} f(\pi_v^{\sigma',\tau'}), c) \\
\inf_{\tau} f(\pi_u^{\sigma,\tau}) & \geq & \delta(\inf_{\tau'} f(\pi_v^{\sigma',\tau'}), c) \\
\sup_{\sigma} \inf_{\tau} f(\pi_u^{\sigma,\tau}) & \geq & \delta(\inf_{\tau'} f(\pi_v^{\sigma',\tau'}), c) \\
\sup_{\sigma} \inf_{\tau} f(\pi_u^{\sigma,\tau}) & \geq & \sup_{\sigma'} \delta(\inf_{\tau'} f(\pi_v^{\sigma',\tau'}), c) = \delta(\sup_{\sigma'} \inf_{\tau'} f(\pi_v^{\sigma',\tau'}), c) \\
\mathrm{val}^{\mathscr{G}}(u) & \geq & \max\left\{ \delta(\mathrm{val}^{\mathscr{G}}(v) : u \xrightarrow{c} v \in E \right\}.
\end{array}$$

The reasoning is similar for $u \in V_{\mathrm{Min}}$, and the converse inequality is also proved along the same lines. □

Saying that $\mathrm{val}^{\mathscr{G}}$ is a fixed point of $\mathbb{O}^{\mathscr{G}}$ defined this way is tightly related to the fact the $\mathscr{G}$ is positionally determined, but not equivalent. Indeed, the fact that $\mathrm{val}^{\mathscr{G}} = \mathbb{O}^{\mathscr{G}}(\mathrm{val}^{\mathscr{G}})$ means that from a vertex u, the value $\mathrm{val}^{\mathscr{G}}(u)$ can be computed locally, *i.e.* by considering the maximum or the minimum over all outgoing edges. It is very tempting to define the argmax and argmin (positional) strategies as follows

$$\begin{array}{l}
u \in V_{\mathrm{Max}} : \ \sigma(u) \in \mathrm{argmax}\left\{ \delta(\mathrm{val}^{\mathscr{G}}(v), c) : u \xrightarrow{c} v \in E \right\}, \\
u \in V_{\mathrm{Min}} : \ \tau(u) \in \mathrm{argmin}\left\{ \delta(\mathrm{val}^{\mathscr{G}}(v), c) : u \xrightarrow{c} v \in E \right\}.
\end{array}$$

and to claim that they are optimal. This is unfortunately often not the case, and one needs to be more careful to define optimal positional strategies. We refer to Figure 6.2 for a counter-example for the case of (stochastic) reachability games.

Let us discuss Property 2, which states that $\mathrm{val}^{\mathscr{G}}$ is the least fixed point of $\mathbb{O}^{\mathscr{G}}$, and specifically how to prove that such a property holds. Under the asumptions of Lemma 8 we already know that $\mathrm{val}^{\mathscr{G}}$ is a fixed point of $\mathbb{O}^{\mathscr{G}}$, which implies that $\mathrm{val}^{\mathscr{G}}$ is larger than the least fixed point. Let us mention another approach for proving this inequality: we define a sequence of objectives $(f_k)_{k\in\mathbb{N}}$ and show the following two properties.

- Writing $\mathrm{val}^{\mathscr{G},f_k}$ for the values of the game equipped with objective f_k, we have $\mathrm{val}^{\mathscr{G},f_k} = \mathbb{O}^k(\mu_0)$, meaning that the k^{th} iteration of $\mathbb{O}$ computes the values for f_k.
- We have $f_0 \leq f_1 \leq \cdots \leq f_k \leq f$.

The second property directly implies that $\mathrm{val}^{\mathscr{G},f_0} \leq \mathrm{val}^{\mathscr{G},f_1} \leq \cdots \leq \mathrm{val}^{\mathscr{G},f_k} \leq \mathrm{val}^{\mathscr{G}}$. Since the least fixed point of $\mathbb{O}$ is the limit of $\mathbb{O}^k(\mu_0)$, by the inequality above it is smaller than $\mathrm{val}^{\mathscr{G}}$. This approach is not easier to use than Lemma 8, but it may be useful as it explains what are the iterations of $\mathbb{O}^{\mathscr{G}}$. We refer to Lemma 35 for an example.

Now let us discuss how to prove that $\mathrm{val}^{\mathscr{G}}$ is the least fixed point. For this, we consider a fixed point μ of $\mathbb{O}^{\mathscr{G}}$, and argue that $\mathrm{val}^{\mathscr{G}} \leq \mu$. To this end, we extract from μ a strategy τ for Min, and show that $\mathrm{val}^{\tau} \leq \mu$; since we know that $\mathrm{val}^{\mathscr{G}} \leq \mathrm{val}^{\tau}$, this implies $\mathrm{val}^{\mathscr{G}} \leq \mu$. Again we refer to Lemma 35 for an example.

Refined complexity analysis

The definition of local operators allows us to refine the complexity analysis for the generic value iteration algorithm. Let us say that $u \xrightarrow{c} v \in E$ is incorrect if $\mu(u) < \delta(\mu(v),c)$, and that u is incorrect if either $u \in V_{\mathrm{Min}}$ and all outgoing edges are incorrect, or $u \in V_{\mathrm{Max}}$ and there exists an incorrect outgoing edge. The algorithmic improvement of the upcoming algorithm over the naive implementations above is that instead of applying the operator $\mathbb{O}^{\mathscr{G}}$ to all vertices at each iteration we can keep track of incorrect vertices and only update them. This computes exactly the same sequence of functions, but at a lesser computational cost. For $u \in V_{\mathrm{Max}}$, it is easy to determine whether it is incorrect, as it relies on the existence of a single outgoing edge. This is more complicated for $u \in V_{\mathrm{Min}}$: the idea here is not to keep track of all incorrect edges, but rather to count them.

Let us write Δ for the complexity of computing $\delta(y,c)$ for given $y \in Y$ and $c \in C$, and of determining whether $y \leq y'$ for $y,y' \in Y$.

Theorem 8 (Complexity analysis of the refined generic value iteration algorithm based on fixed point through monotonicity)**.** *Assume Properties 2 and 4 (fixed point through monotonicity and monotonicity of δ), and that Y is finite. Then the generic value iteration algorithm outputs $\mathrm{val}^{\mathscr{G}}$ within at most $n \cdot |Y|$ iterations. The computational cost of a single iteration is $O(m \cdot \Delta)$, so the running time of the whole algorithm is $O(nm \cdot \Delta \cdot |Y|)$.*

The data structure consists of the following objects:

- an element of Y for each vertex, representing the current function $\mu : V \to Y$;
- a set `Incorrect` of vertices (the order in which vertices are stored and retrieved from the set does not matter);
- a table `Count` storing for each vertex of Min a number of edges.

The invariant of the algorithm satisfied before each iteration of the repeat loop is the following:

- for $u \in V_{\mathrm{Min}}$, the value of `Count`(u) is the number of incorrect outgoing edges of u;
- `Incorrect` is the set of incorrect vertices.

The invariant is satisfied initially thanks to the function `Init`. Let us assume that we choose and remove u from `Incorrect`. Since we modify only $\mu(u)$ the only potentially incorrect vertices are in `Incorrect` (minus u) and the incoming edges of u; for the latter each of them is checked and added to `Incorrect`$'$ when required. By monotonicity, incorrect vertices remain incorrect so all vertices in `Incorrect` (minus u) are still incorrect. Hence the invariant is satisfied.

The invariant implies that the algorithm indeed implements Algorithm 1.3 hence returns the least fixed point, but it also has implications on the complexity. Indeed one iteration of the repeat loop over some vertex u involves

$$O\left((|\mathrm{In}^{-1}(u)|+|\mathrm{Out}^{-1}(u)|)\cdot\Delta\right)$$

operations, the first term corresponds to updating $\mu(u)$ and `Incorrect`, which requires for each outgoing edge of u to compute δ, and the second term corresponds to considering all incoming edges of u and treating the incorrect ones. Thus the overall complexity for a single iteration is

$$O\left(\sum_{u\in V}(|\mathrm{In}^{-1}(u)|+|\mathrm{Out}^{-1}(u)|)\cdot\Delta\right)=O(m\cdot\Delta).$$

The pseudocode for the case of fixed point through monotonicity is given in Algorithm 1.3. It can be easily adapted to the case of fixed point through contraction by adding a stopping criterion when $\|\mathbb{O}^{\mathscr{G}}(\mu)-\mu\|$ is small enough.

The correctness of the algorithm is a consequence of the following invariant: for $\mu\in F_Y$, if `Incorrect` is the set of incorrect vertices and for all $u\in V_{\mathrm{Min}}$, `Count`(u) the number of incorrect outgoing edges of u, then applying `Treat`(u) and `Update`(u) to all vertices $u\in$ `Incorrect` results in updating μ to $\mathbb{O}^{\mathscr{G}}(\mu)$ and correctly updating `Incorrect` and `Count`. The counting trick is valid because the edge $v\xrightarrow{c}u$ is considered at most once in calls to the function `Update` between two calls to `Update`(u).

Theorem 9 (Complexity analysis of the refined generic value iteration algorithm based on fixed point through contraction)**.** *Assume Properties 3 and 4 (fixed point through contraction, monotonicity of δ). Then the generic value iteration algorithm computes an ε-approximation of $\mathrm{val}^{\mathscr{G}}$ within at most $O\left(\frac{\log(\varepsilon)}{\log(\lambda)}\right)$ iterations, where $\lambda\in(0,1)$ is the contraction factor of $\mathbb{O}^{\mathscr{G}}$. The computational cost of a single iteration is $O(m\cdot\Delta)$, so the running time of the whole algorithm is $O(nm\cdot\Delta\cdot|Y|)$.*

1.10 Strategy improvement algorithms

Value iteration algorithms manipulate value functions and never construct any strategy, at least explicitly. This is a key difference with strategy improvement algorithms (also called policy iteration algorithms) whose fundamental idea is to maintain and improve a strategy. Let us consider a 'local' operator $\mathbb{O}^{\mathscr{G}}$, induced by a function $\delta: Y\times C\to Y$:

$$\mathbb{O}^{\mathscr{G}}(\mu)(u)=\begin{cases}\max\left\{\delta(\mu(v),c):u\xrightarrow{c}v\in E\right\} & \text{if } u\in V_{\mathrm{Max}},\\ \min\left\{\delta(\mu(v),c):u\xrightarrow{c}v\in E\right\} & \text{if } u\in V_{\mathrm{Min}}.\end{cases}$$

Algorithm 1.3: A generic value iteration algorithm based on fixed point through monotonicity – refined version.

Function `Init()`**:**
- **for** $u \in V$ **do**
  - $\mu(u) \leftarrow \bot$
- **for** $u \in V_{\text{Min}}$ **do**
  - **for** $u \xrightarrow{c} v \in E$ **do**
    - **if** *incorrect:* $\mu(u) < \delta(\mu(v), c)$ **then**
      - $\texttt{Count}(u) \leftarrow \texttt{Count}(u) + 1$
  - **if** $\mathit{Count}(u) = \mathit{Degree}(u)$ **then**
    - Add u to `Incorrect`
- **for** $u \in V_{\text{Max}}$ **do**
  - **for** $u \xrightarrow{c} v \in E$ **do**
    - **if** *incorrect:* $\mu(u) < \delta(\mu(v), c)$ **then**
      - Add u to `Incorrect`

Function `Treat(`u`)`**:**
- **if** $u \in V_{\text{Max}}$ **then**
  - $\mu(u) \leftarrow \max\left\{\delta(\mu(v), c) : u \xrightarrow{c} v \in E\right\}$
- **if** $u \in V_{\text{Min}}$ **then**
  - $\mu(u) \leftarrow \min\left\{\delta(\mu(v), c) : u \xrightarrow{c} v \in E\right\}$

Function `Update(`u`)`**:**
- **if** $u \in V_{\text{Min}}$ **then**
  - $\texttt{Count}(u) \leftarrow 0$
- **for** $v \xrightarrow{c} u \in E$ **do**
  - **if** $v \xrightarrow{c} u$ *is incorrect* **then**
    - **if** $v \in V_{\text{Min}}$ **then**
      - $\texttt{Count}(v) \leftarrow \texttt{Count}(v) + 1$
      - **if** $\mathit{Count}(v) = \mathit{Degree}(v)$ **then**
        - Add v to `Incorrect`$'$
    - **if** $v \in V_{\text{Max}}$ **then**
      - Add v to `Incorrect`$'$

Function `Main()`**:**
- `Init` ()
- **for** $i = 0, 1, 2, \ldots$ **do**
  - `Incorrect`$' \leftarrow \emptyset$
  - **for** $u \in \mathit{Incorrect}$ **do**
    - `Treat` (u)
    - `Update` (u)
  - **if** $\mathit{Incorrect}' = \emptyset$ **then**
    - **return** μ
  - **else**
    - `Incorrect` $\leftarrow$ `Incorrect`$'$

Let us recall from the previous section that we consider two scenarios:

- in the quantitative case, we start from a quantitative condition $f : C^\omega \to \mathsf{R} \cup \{\pm\infty\}$,
- in the qualitative case, we assume the existence of a function $f : C^\omega \to Y$, and thanks to Property 1 computing the value function $\mathrm{val}^{\mathscr{G}}$ yields the winning regions.

For value iteration algorithms it was enough for Y to be a lattice. Here we need a stronger assumption: Y is a total order.

Let us consider a game $\mathscr{G}$ and set as a goal to construct an optimal strategy for Max. The key idea behind strategy improvement is to use val^σ to improve the strategy σ by *switching edges*, which is an operation that creates a new strategy. This involves defining the notion of *improving edges*: let us consider a vertex $u \in V_{\mathrm{Max}}$, we say that $e : u \xrightarrow{c} v$ is an *improving edge* if

$$\delta(\mathrm{val}^\sigma(v), c) > \mathrm{val}^\sigma(u).$$

Intuitively: according to val^σ, playing e is better[2] than playing $\sigma(u)$.

Given a strategy σ and a set of improving edges S (for each $u \in V_{\mathrm{Max}}$, S contains at most one outgoing edge of u), we write $\sigma[S]$ for the strategy

$$\sigma[S](u) = \begin{cases} e & \text{if there exists } e = u \xrightarrow{c} v \in S, \\ \sigma(v) & \text{otherwise.} \end{cases}$$

The difficulty is that an edge being improving does not mean that it is a better move than the current one in any context, but only according to the value function val^σ, so it is not clear that $\sigma[S]$ is better than σ. Strategy improvement algorithms depend on the following two principles:

- **Progress**: updating a strategy using improving edges is a strict improvement,
- **Optimality**: a strategy which does not have any improving edges is optimal.

The pseudocode of the algorithm is given in Algorithm 1.4. The algorithm is nondeterministic, in the sense that both the initial strategy and at each iteration, the choice of improving edge can be chosen arbitrarily. A typical choice, called the "greedy all-switches" rule, choosing for each $u \in V_{\mathrm{Max}}$ a maximal improving edge, meaning

$$\mathrm{argmax} \left\{ \delta(\mathrm{val}^\sigma(v), c) : u \xrightarrow{c} v \in E \right\}.$$

Let us write $\sigma \leq \sigma'$ if for all vertices v we have $\mathrm{val}^\sigma(v) \leq \mathrm{val}^{\sigma'}(v)$, and $\sigma < \sigma'$ if additionally $\neg(\sigma' \leq \sigma)$. Unfortunately, there are no generic correctness proofs for the progress property for this algorithm (we refer to the reference section for further discussion). However, the optimality property can be proved at this level of generality,

[2] We need Y to be totally ordered here.

Algorithm 1.4: The generic strategy improvement algorithm.

```
Choose an initial strategy σ_0 for Max
for i = 0, 1, 2, ... do
    Compute val^{σ_i} and the set of improving edges
    if σ_i does not have improving edges then
        return σ_i
    Choose a non-empty set S_i of improving edges
    σ_{i+1} ← σ_i[S_i]
```

for the two approaches for fixed point computations: this is the object of the next two subsections.

We refer to Section 4.3 for a concrete instantiation for energy games using monotonic fixed point computations, and to Section 4.4 for discounted payoff games using contracting fixed point computations.

Fixed point through monotonicity

Let us start with an important remark: Property 2 (fixed point through monotonicity) assumes that the value function is computed as the least fixed point of the monotonic operator $\mathbb{O}^{\mathscr{G}}$. It is very important in the developments below that this is a least fixed point, and not a greatest fixed point: the proof cannot be easily adapted to this latter case.

Theorem 10 (Optimality property for strategy improvement algorithms based on fixed point through monotonicity)**.** *Assume Properties 2 and 4 (fixed point through monotonicity, monotonicity of δ). Let σ a strategy that has no improving edges, then σ is optimal.*

Proof. We prove the contrapositive: assume that σ is not optimal, we show that it must have some improving edge. The fact that σ is not optimal means that $\mathrm{val}^{\sigma} < \mathrm{val}^{\mathscr{G}}$. Since $\mathrm{val}^{\mathscr{G}}$ is the least fixed point of $\mathbb{O}^{\mathscr{G}}$, it is also its least pre-fixed point. Therefore val^{σ} is not a pre-fixed point: $\neg(\mathrm{val}^{\sigma} \geq \mathbb{O}^{\mathscr{G}}(\mathrm{val}^{\sigma}))$. Hence there exists $u \in V$ such that $\mathrm{val}^{\sigma}(u) < \mathbb{O}^{\mathscr{G}}(\mathrm{val}^{\sigma})(u)$.

We rule out the case that $u \in V_{\mathrm{Min}}$: since val^{σ} is a fixed point of $\mathbb{O}^{\mathscr{G}[\sigma]}$, this implies that for $u \in V_{\mathrm{Min}}$ we have $\mathrm{val}^{\sigma}(u) = \min\left\{\delta(\mathrm{val}^{\sigma}(v), c) : u \xrightarrow{c} v \in E\right\}$, equal to $\mathbb{O}^{\mathscr{G}}(\mathrm{val}^{\sigma})(u)$. Therefore $u \in V_{\mathrm{Max}}$, implying that there exists $u \xrightarrow{c} v$ such that $\mathrm{val}^{\sigma}(u) < \delta(\mathrm{val}^{\sigma}(v), c)$. This is the definition of $u \xrightarrow{c} v$ being an improving edge. □

Fixed point through contraction

Theorem 11 (Optimality property for strategy improvement algorithms based on fixed point through contraction)**.** *Assume Property 3 (fixed point through contraction). Let σ be a strategy that has no improving edges, then σ is optimal.*

Proof. Let σ be a strategy that has no improving edges. We claim that val^σ is a fixed point of $\mathbb{O}^{\mathscr{G}}$, which thanks to Property 3 implies that $\mathrm{val}^\sigma = \mathrm{val}^{\mathscr{G}}$, meaning that σ is optimal.

Thanks to Property 3 val^σ is the unique fixed point of $\mathbb{O}^{\mathscr{G}[\sigma]}$, so for $u \in V_{\mathrm{Min}}$ we have $\mathrm{val}^\sigma(u) = \min\left\{\delta(\mathrm{val}^\sigma(v), c) : u \xrightarrow{c} v \in E\right\}$.

The fact that σ has no improving edges reads: for all $u \in V_{\mathrm{Max}}$, for all $u \xrightarrow{c'} v' \in E$, $\delta(\mathrm{val}^\sigma(v'), c') \leq \delta(\mathrm{val}^\sigma(v), c)$ where $\sigma(u) = u \xrightarrow{c} v$. Since $\mathrm{val}^\sigma(u) = \delta(\mathrm{val}^\sigma(v), c)$, this implies that $\mathrm{val}^\sigma(u) = \max\left\{\delta(\mathrm{val}^\sigma(v'), c) : u \xrightarrow{c} v' \in E\right\}$.

The two equalities above witness that val^σ is the unique fixed point of $\mathbb{O}^{\mathscr{G}}$. □

Bibliographic references

The study of games, usually called game theory, has a very long history rooted in mathematics, logic, and economics, among other fields. Foundational ideas and notions emerged from set theory with for instance backward induction by Zermelo [Zer13], and topology with determinacy results by Martin [Mar75] (stated as Theorem 1 in this chapter), and Banach-Mazur and Gale-Stewart games [GS53].

The topic of this book is a small part of game theory: we focus on infinite duration games played on graphs. In this chapter we defined deterministic games, meaning games with no source of randomness, which will be the focus of Part I. Part II introduces stochastic games, which were initially studied in mathematics. We refer to Section 6.6.3 for more bibliographic references on stochastic games, and focus in this chapter on references for deterministic games.

The model presented in this chapter emerged from the study of automata theory and logic, where it is used as a tool for various purposes. Let us first discuss the role of games in two contexts: for solving the synthesis problem of reactive systems and for automata and logic over infinite trees.

The synthesis problem for non-terminating reactive systems was formulated in general terms by Church [Chu57, Chu62] and is therefore also called Church's problem: from a specification of a step-by step transformation of an input stream given in some logical formalism, construct a system satisfying the specification. The first published paper solving Church's problem for monadic second-order logic was written by Büchi and Landweber [BL69], following a paper by Landweber [Lan67] (then Büchi's PhD student) focussing on solving games. However, the idea of casting the synthesis problem as a game between a system and its environment is due to McNaughton: in the technical report [McN65] McNaughton attempted to give a solution to the synthesis problem using games, initiating many of the most important ideas for analysing games. Unfortunately the proof contained an error which Landweber detected and communicated to McNaughton, who then decided to let Landweber publish his complete solution. One of the most difficult step in the solution of Church's problem for monadic second-order logic by Büchi and Landweber [BL69] is the determinisation procedure from Büchi to Muller automata due to McNaughton [McN66]. We refer to Thomas'

survey [Tho09] for more details on some historical and technical aspects of the early papers on Church's synthesis problem.

Games emerged in another aspect of automata theory: for understanding the difficult result of Rabin [Rab69a] saying that automata over infinite trees can be effectively complemented. This is the key step for proving Rabin's seminal result that the monadic second-order theory of the infinite binary tree is decidable. The celebrated paper of Gurevich and Harrington [GH82] revisits Rabin's result by reducing the complementation question to a determinacy result for games. Interestingly, they credit McNaughton for 'airing the idea' of using games in this context and then for exploiting it to Landweber [Lan67], Büchi and Landweber [BL69], and Büchi [Büc77].

Both lines of work have been highly influential in automata theory and logic; we refer to the reference section in Chapter 2 for more bibliographic references on this connection. They bind automata theory and logic to the study of games on graphs and provide motivations and questions many of which are still open today.

Beyond these two examples there are many applications of games in theoretical computer science and logic in particular. The following quote is due to Hodges [Hod93]:

> An extraordinary number of basic ideas in model theory can be expressed in terms of games.

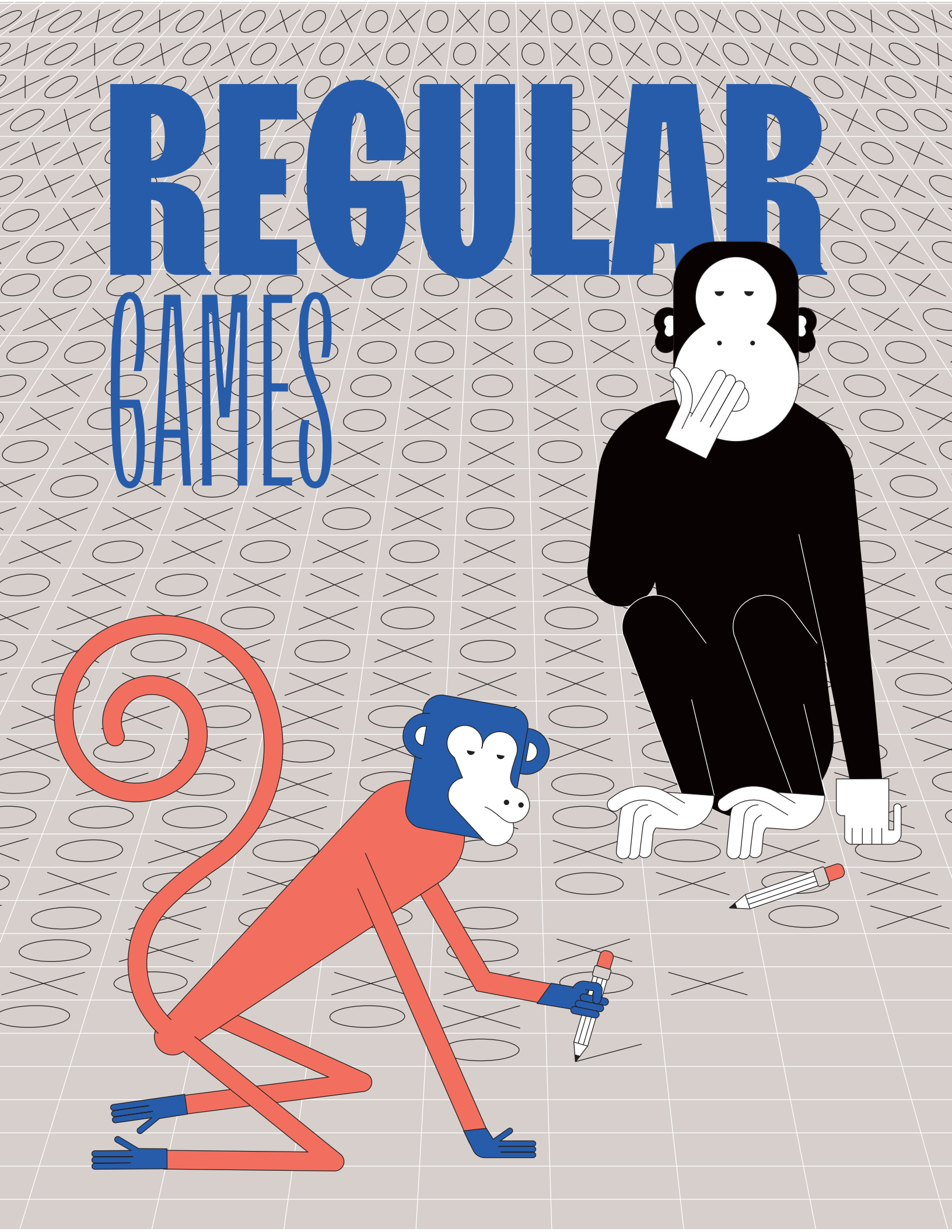
Let us mention model checking games, which are used for checking whether a model satisfies a formula. They often form both a theoretical tool for understanding the model checking problem and proving its properties, as well as an algorithmic backend for effectively deciding properties of a logical formalism (we refer to [Grä02] for a survey on model checking games). Another important construction of a game for understanding logical properties is the Ehrenfeucht-Fraïssé games [Ehr61, Fra50, Fra53] whose goal is to determine whether two models are equivalent against a logical formalism.

Value iteration and strategy improvement algorithms are the most common families of algorithms. The latter are also often called policy improvement or policy iteration. There are many frameworks presenting them in generic terms as we did in this chapter, see for instance [CFGO22] for value iteration and [Ohl21, Koz21] for strategy improvement algorithms. The PhD thesis [Ohl21] shows a generic argument for the progress property to hold in the setting of monotonic universal graphs, but it considers the restricted version where only one edge can be switched at a time.

Part I

Classic

REGULAR GAMES



Chapter 2

Regular Games

NATHANAËL FIJALKOW, FLORIAN HORN

This chapter considers the so-called regular games, which from the easiest to the most complicated are: reachability, Büchi, parity, Rabin, and then Muller games. We develop in Section 2.1 the notion of attractors for solving reachability games. This is the main building block for constructing algorithms throughout the book. The next step is Büchi games in Section 2.2. We then construct a conceptually simple exponential time recursive algorithm for solving parity games in Section 2.3. Section 2.4 is a short digression about a general result for proving positional determinacy of qualitative games. We come back to regular games in Section 2.5, extending the recursive algorithm of parity games to Muller games, and discuss the computational complexities of solving Rabin, Streett, and Muller games. Finally, Section 2.6 is devoted to the combinatorial notion of the Zielonka tree, which beautifully explains the memory requirements for Muller games and gives additional insights into the structures of Rabin and parity objectives.

Remark 6 (Finite versus infinite games)**.** *As in the rest of the book unless otherwise specified we consider finite games. However all positionality and finite memory determinacy results proved in this chapter hold for infinite games. In all cases the proofs we give use the finiteness of the games. In many cases, the proofs can be extended to infinite games with a technical overhead involving in particular a transfinite induction. The difficulty is illustrated before the proof of Theorem 12.*

2.1 Reachability games

Recall that the objective `Reach` requires that the colour Win appears at least once and `Safe` requires that the colour Lose never appears. We identify the colour Win with $\mathtt{c}^{-1}(\text{Win})$ the set of edges labelled Win, so we write $e \in \text{Win}$ when $\mathtt{c}(e) = \text{Win}$, and similarly for Lose. Therefore $\mathtt{Reach}(\mathtt{c})$ can equivalently can described as $\mathtt{Reach}(\text{Win})$.

Theorem 12 (Positional determinacy and complexity of reachability games)**.** *Reachability objectives are uniformly positionally determined. There exists an algorithm for computing the winning regions of reachability games in linear time and space. More precisely the time and space complexity are both $O(m)$.*

The positional determinacy result holds for infinite arenas.

The complexity results are stated in the unit cost RAM model with machine word size $w = \log(m)$ with m the number of edges. We refer to Section 1.2 for more details about the model, which is in subtle ways different than the Turing model. The complexity would be slightly different in the Turing model: an additional $\log(m)$ factor would be incurred for manipulating numbers of order m, which the unit cost RAM model allows us to conveniently hide.

In the literature the complexity $O(n+m)$ is often reported for solving reachability games. Since we make the assumption that every vertex has an outgoing edge this implies that $n \leq m$, so $O(n+m) = O(m)$.

Reachability and safety games are most often defined labelling vertices than edges. As explained in Section 1.1, labelling edges is slightly more general than labelling vertices. To improve readability, let us first consider the case where we label vertices, and then explain how this (seamlessly) extends to labelling edges. The condition is $\texttt{Reach}(\mathrm{Win})$ with Win a set of vertices. Let us introduce some notations. For a subset $X \subseteq V$, we let $\mathrm{Pre}_{\mathrm{Eve}}(X) \subseteq V$ the set of vertices from which Eve can ensure that the next vertex is in X:

$$\begin{array}{rcl} \mathrm{Pre}_{\mathrm{Eve}}(X) & = & \{u \in V_{\mathrm{Eve}} : \exists u \to v \in E, v \in X\} \\ & \cup & \{u \in V_{\mathrm{Adam}} : \forall u \to v \in E,\ v \in X\}. \end{array}$$

Let us define an operator on subsets of vertices:

$$X \mapsto \mathrm{Win} \cup \mathrm{Pre}_{\mathrm{Eve}}(X).$$

We note that this operator is monotonic when equipping the powerset of vertices with the inclusion preorder: if $X \subseteq X'$ then $\mathrm{Pre}_{\mathrm{Eve}}(X) \subseteq \mathrm{Pre}_{\mathrm{Eve}}(X')$. Hence Theorem 4 applies: this operator has a least fixed point which we call the attractor of Win for Eve and write $\mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win})$, and it is computed by the following sequence: we let $\mathrm{Attr}^0_{\mathrm{Eve}}(\mathrm{Win}) = \mathrm{Win}$ and

$$\mathrm{Attr}^{k+1}_{\mathrm{Eve}}(\mathrm{Win}) = \mathrm{Win} \ \cup \ \mathrm{Pre}_{\mathrm{Eve}}(\mathrm{Attr}^k_{\mathrm{Eve}}(\mathrm{Win})).$$

This constructs a sequence $(\mathrm{Attr}^k_{\mathrm{Eve}}(\mathrm{Win}))_{k\in\mathbb{N}}$ of non-decreasing subsets of V. If the game is finite and n is the number of vertices, the sequence stabilises after at most $n-1$ steps, *i.e.* $\mathrm{Attr}^{n-1}_{\mathrm{Eve}}(\mathrm{Win}) = \mathrm{Attr}^n_{\mathrm{Eve}}(\mathrm{Win}) = \mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win})$.

Let us drop the finiteness assumption: if the game is infinite but has finite outdegree, meaning that for any vertex there is a finite number of outgoing edges, then the operator above preserves suprema so thanks to Theorem 4 we have $\mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win}) = \bigcup_{k\in\mathbb{N}} \mathrm{Attr}^k_{\mathrm{Eve}}(\mathrm{Win})$. In full generality the operator does not preserve suprema and the use of ordinals is necessary: we define the sequence $(\mathrm{Attr}^\alpha_{\mathrm{Eve}}(\mathrm{Win}))$ indexed by ordinals up to the cardinal of $\mathscr{G}$, the case of a limit ordinal α being $\mathrm{Attr}^\alpha_{\mathrm{Eve}}(\mathrm{Win}) =$

$\bigcup_{\beta<\alpha} \mathrm{Attr}^{\beta}_{\mathrm{Eve}}(\mathrm{Win})$. We then show that $\mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win})$ is the union of all elements in this sequence. We do not elaborate further this most general case but note that the overhead is mostly technical, the proof below of Lemma 9 can be adapted with little changes using a transfinite induction.

The following lemma shows how the attractor yields a solution to reachability games and directly implies Theorem 12.

Lemma 9 (Characterisation of the winning region of reachability games using attractors)**.** *Let $\mathscr{G}$ a reachability game. Then $W_{\mathrm{Eve}}(\mathscr{G}) = Attr_{\mathrm{Eve}}(Win)$, and:*

- *there exists a uniform positional strategy σ for Eve called the* attractor strategy *defined on $Attr_{\mathrm{Eve}}(Win)$ which ensures to reach Win from any vertex in $Attr_{\mathrm{Eve}}(Win)$, with the property that for any $k \in \mathbb{N}$ all plays consistent with σ from $Attr^{k}_{\mathrm{Eve}}(Win)$ reach Win within k steps and remain in $Attr_{\mathrm{Eve}}(Win)$ until doing so;*
- *there exists a uniform positional strategy τ for Adam called the* counter-attractor strategy *defined on $V \setminus Attr_{\mathrm{Eve}}(Win)$ which ensures never to reach Win from any vertex in $V \setminus Attr_{\mathrm{Eve}}(Win)$, with the property that all plays consistent with τ remain in $V \setminus Attr_{\mathrm{Eve}}(Win)$.*

The following definition is very important: for $v \in V$, the rank of v is the smallest $k \in \mathbb{N}$ such that $v \in \mathrm{Attr}^{k}_{\mathrm{Eve}}(\mathrm{Win})$.

Proof. We first show that $\mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win}) \subseteq W_{\mathrm{Eve}}(\mathscr{G})$. We use the rank to define a positional strategy σ for Eve. Let $u \in V_{\mathrm{Eve}}$ of rank $k+1$, then $u \in \mathrm{Pre}_{\mathrm{Eve}}(\mathrm{Attr}^{k}_{\mathrm{Eve}}(\mathrm{Win}))$, so there exists $u \to v \in E$ such that $v \in \mathrm{Attr}^{k}_{\mathrm{Eve}}(\mathrm{Win})$, define $\sigma(u) = u \to v$. If $u \in V_{\mathrm{Eve}}$ has rank 0, meaning $u \in \mathrm{Win}$, the game is already won.

We argue that σ ensures $\mathtt{Reach}(\mathrm{Win})$. By construction in any play consistent with σ at each step either we are in Win or the rank decreases by at least one. Thus any play consistent with σ from $\mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win})$ reaches Win.

We now show that $W_{\mathrm{Eve}}(\mathscr{G}) \subseteq \mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win})$. For this we actually show

$$V \setminus \mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win}) \subseteq W_{\mathrm{Adam}}(\mathscr{G}).$$

Indeed, $W_{\mathrm{Adam}}(\mathscr{G}) \subseteq V \setminus W_{\mathrm{Eve}}(\mathscr{G})$, because Eve and Adam cannot have a winning strategy from the same vertex. This property is clear and holds for any game, it should not be confused with determinacy.

We define a positional strategy τ for Adam from $V \setminus \mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win})$. Let $u \in V_{\mathrm{Adam}}$ in $V \setminus \mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win})$, there exists $u \to v \in E$ such that $v \in V \setminus \mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win})$, define $\tau(u) = u \to v$. Similarly, if $u \in V_{\mathrm{Eve}}$ in $V \setminus \mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win})$, then for all $u \to v \in E$, we have $v \in V \setminus \mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win})$. It follows that any play consistent with τ remain in $V \setminus \mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win})$ hence never reaches Win, in other words τ ensures $\mathtt{Safe}(\mathrm{Win})$ from $V \setminus \mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win})$. □

Let us note that the proof above extends very easily to the edge labelling case. Let Win a set of edges, we define:

$$\mathrm{Win}' = \{u \in V_{\mathrm{Eve}} : \exists e = u \to v \in E,\ e \in \mathrm{Win}\} \cup \{u \in V_{\mathrm{Adam}} : \forall e = u \to v \in E,\ e \in \mathrm{Win}\}.$$

Eve has a winning strategy for $\texttt{Reach}(\text{Win})$ if and only if she has one for $\texttt{Reach}(\text{Win}')$, which reduces edge labelling to vertex labelling for reachability games. In the pseudocode of Algorithm 2.1 we use the (more general) edge labelling convention. Algorithm 2.1 is an efficient implementation of the attractor computation, and more precisely it computes the ranks of all vertices: it returns a function $\mu : V \to \mathbb{N} \cup \{\infty\}$ such that $\mu(u)$ is the rank of u, as stated in the following theorem.

Theorem 13 (Computing ranks for reachability games)**.** *There exists an algorithm for computing the ranks of all vertices in reachability games in linear time and space. More precisely the time and space complexity are both $O(m)$.*

In Section 4.5 we will generalise this algorithm to a quantitative setting, considering the shortest path objective. We invite the reader to compare this algorithm to the generic value iteration algorithm presented in Algorithm 1.3: it is indeed the instantiation of this framework for the quantitative shortest path objective with all weights equal to one.

The correctness of the algorithm hinges on the following invariant: for $i \geq 1$, before the i^{th} iteration in the `Main` function,

- μ has correctly computed the ranks of vertices strictly less than i,
- `Incorrect` is the set of vertices of rank $i-1$,
- for each $v \in V_{\text{Adam}}$, $\texttt{Count}(v)$ is the number of outgoing edges of v to vertices of ranks strictly less than i.

The function `Init` ensures these properties for $i = 1$. To see that the invariant is preserved, note that each vertex v is updated at most once, and therefore each edge $u \to v$ is considered at most once, so `Count` is correctly updated. To get the overall $O(m)$ complexity, we note that each vertex v is updated at most once over the course of the algorithm.

Remark 7 (RAM versus Turing models of computation)**.** *We note that in the complexity analysis the cost of manipulating (and in particular incrementing) the counters for the number of edges is constant, which holds in the unit cost RAM model of computation. The same algorithm analysed in the Turing model of computation would have an additional $O(\log(n))$ multiplicative factor in the time complexity to take this into account.*

Subgames

Additionally to solving reachability games, the notion of attractors induces a common way of constructing traps and subgames. This discussion is very closely related to the notion of traps defined in Section 1.7, but some extra care is required because of the edge labelling convention.

Let $\mathscr{G}$ a game and $F \subseteq E$ a subset of edges. By definition of the attractor:

- for every $u \in V_{\text{Adam}} \setminus \text{Attr}_{\text{Eve}}(F)$, there exists $u \to v \in E$ which is not in F and $v \in V \setminus \text{Attr}_{\text{Eve}}(F)$, and

Algorithm 2.1: The linear time algorithm for reachability games.

Data: A reachability game
Function `Init()`**:**
  for $u \in V$ **do**
    $\mu(u) \leftarrow \infty$
    if $u \in V_{\text{Adam}}$ **then**
      $\texttt{Count}(u) \leftarrow 0$
  for $u \in V_{\text{Eve}}$ **do**
    for $u \xrightarrow{\mathit{Win}} v \in E$ **do**
      $\mu(u) \leftarrow 0$
      Add u to `Incorrect`
  for $u \in V_{\text{Adam}}$ **do**
    for $u \xrightarrow{\mathit{Win}} v \in E$ **do**
      $\texttt{Count}(u) \leftarrow \texttt{Count}(u) + 1$
    if $\mathit{Count}(u) = \mathit{Degree}(u)$ **then**
      $\mu(u) \leftarrow 0$
      Add u to `Incorrect`

Function `Update(`u`)`**:**
  for $v \to u \in E$ **do**
    if $v \in V_{\text{Adam}}$ **then**
      $\texttt{Count}(v) \leftarrow \texttt{Count}(v) + 1$
      if $\mathit{Count}(v) = \mathit{Degree}(v)$ **then**
        Add v to $\texttt{Incorrect}'$
    if $v \in V_{\text{Eve}}$ **then**
      Add v to $\texttt{Incorrect}'$

Function `Main()`**:**
  `Init` ()
  for $i = 1, 2, \ldots$ **do**
    $\texttt{Incorrect}' \leftarrow \emptyset$
    for $u \in \mathit{Incorrect}$ **do**
      $\mu(u) \leftarrow i$
      `Update` (u)
    if $\mathit{Incorrect}' = \emptyset$ **then**
      return μ
    else
      $\texttt{Incorrect} \leftarrow \texttt{Incorrect}'$

- for every $u \in V_{\text{Eve}} \setminus \text{Attr}_{\text{Eve}}(F)$, for all $e = u \to v \in E$, we have $e \notin F$ and $v \in V \setminus \text{Attr}_{\text{Eve}}(F)$.

This means that we can define the subgame $\mathcal{G} \setminus \text{Attr}_{\text{Eve}}(F)$ as follows: the set of vertices is $V \setminus \text{Attr}_{\text{Eve}}(F)$, and the set of edges $\mathcal{G}$ to the subset of $E \setminus F$ where both incoming and outgoing vertices are in $V \setminus \text{Attr}_{\text{Eve}}(F)$. The colouring function and the condition are naturally induced from $\mathcal{G}$ to $\mathcal{G} \setminus \text{Attr}_{\text{Eve}}(F)$.

Let us emphasise a subtlety here: indeed $V \setminus \text{Attr}_{\text{Eve}}(F)$ is a trap for Eve, so we can define the subgame induced by the set of vertices $V \setminus \text{Attr}_{\text{Eve}}(F)$. But it is not the same as $\mathcal{G} \setminus \text{Attr}_{\text{Eve}}(F)$: in the latter we remove the edges in F, which may still be present in the induced subgame.

Lemma 10 (Attractors induce subgames – statement for Adam)**.** *Let τ a strategy for Adam in the subgame $\mathcal{G} \setminus Attr_{\text{Eve}}(F)$, it induces a strategy τ' in $\mathcal{G}$ such that plays consistent with τ in $\mathcal{G} \setminus Attr_{\text{Eve}}(F)$ are in one-to-one correspondence with plays consistent with τ' in $\mathcal{G}$, in particular any play consistent with τ' stays forever in $\mathcal{G} \setminus Attr_{\text{Eve}}(F)$.*

This very useful lemma is very heavily used when decomposing games, and in a small abuse of notations we identify the strategies τ and τ'.

The analogous statement can be made for the subgame $\mathcal{G} \setminus \text{Attr}_{\text{Adam}}(F)$:

Lemma 11 (Attractors induce subgames – statement for Eve)**.** *Let σ a strategy for Eve in the subgame $\mathcal{G} \setminus Attr_{\text{Adam}}(F)$, it induces a strategy σ' in $\mathcal{G}$ such that plays consistent with σ in $\mathcal{G} \setminus Attr_{\text{Adam}}(F)$ or in one-to-one correspondence with plays consistent with σ' in $\mathcal{G}$, in particular any play consistent with σ' stays forever in $\mathcal{G} \setminus Attr_{\text{Adam}}(F)$.*

2.2 Büchi games

Recall that the objective `Buchi` requires that the colour Win appears infinitely many times and `CoBuchi` requires that the colour Lose appears finitely many times.

Theorem 14 (Positional determinacy and complexity of Buchi games)**.** *Büchi objectives are uniformly positionally determined*[1]*. There exists an algorithm for computing the winning regions of Büchi games in quadratic time, more precisely $O(mn)$, and linear space, more precisely $O(m)$.*

We present two different yet very similar algorithms.

A first algorithm

The following lemma implies Theorem 14.

Lemma 12 (Fixed point characterisation of the winning region for Büchi games)**.** *Let $\mathcal{G}$ be a Büchi game.*

- *If $Attr_{\text{Eve}}(Win) = V$, then $W_{\text{Eve}}(\mathcal{G}) = V$.*

[1] See Remark 6 for the case of infinite games.

- *If* $Attr_{\mathrm{Eve}}(Win) \neq V$*, let* $\mathscr{G}' = \mathscr{G} \setminus Attr_{\mathrm{Adam}}(V \setminus Attr_{\mathrm{Eve}}(Win))$*, then* $W_{\mathrm{Eve}}(\mathscr{G}) = W_{\mathrm{Eve}}(\mathscr{G}')$.

Proof. We prove the first item. Let σ be an attractor strategy ensuring to reach Win from $\mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win}) = V$. We argue that σ ensures $\mathtt{Buchi}(\mathrm{Win})$. Indeed a play consistent with σ can be divided into infinitely many finite plays, each of them consistent with σ until reaching Win, and starting from scratch from the next vertex onwards. Thus σ is winning from V.

We now look at the second item. We first prove that $\mathrm{Attr}_{\mathrm{Adam}}(V \setminus \mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win})) \subseteq W_{\mathrm{Adam}}(\mathscr{G})$. Let τ_a denote an attractor strategy ensuring to reach $V \setminus \mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win})$ from $\mathrm{Attr}_{\mathrm{Adam}}(V \setminus \mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win}))$, and τ_c a counter-attractor strategy ensuring to never reach Win from $V \setminus \mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win})$. We construct the strategy τ as the disjoint union of τ_a and τ_c:

$$\tau(v) = \begin{cases} \tau_a(v) & \text{if } v \in \mathrm{Attr}_{\mathrm{Adam}}(V \setminus \mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win})) \setminus (V \setminus \mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win})), \\ \tau_c(v) & \text{if } v \in V \setminus \mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win}). \end{cases}$$

Any play consistent with τ is first consistent with τ_a until reaching $V \setminus \mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win})$ and then is consistent with τ_c and stays there forever. In this second phase it does not visit Win, implying that the play visits Win finitely many times, so it is winning. Thus we have proved that $\mathrm{Attr}_{\mathrm{Adam}}(V \setminus \mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win})) \subseteq W_{\mathrm{Adam}}(\mathscr{G})$, implying $W_{\mathrm{Eve}}(\mathscr{G}) \subseteq V \setminus \mathrm{Attr}_{\mathrm{Adam}}(V \setminus \mathrm{Attr}_{\mathrm{Eve}}(\mathrm{Win}))$.

We now show that $W_{\mathrm{Eve}}(\mathscr{G}') \subseteq W_{\mathrm{Eve}}(\mathscr{G})$, which implies the converse inclusion. Consider a winning strategy from $W_{\mathrm{Eve}}(\mathscr{G}')$ in $\mathscr{G}'$, thanks to Lemma 11 it induces a winning strategy in $\mathscr{G}$. □

The algorithm is presented in pseudocode in Algorithm 2.2. For the complexity analysis, the algorithm performs at most n recursive calls and each of them involves two attractor computations, implying the time complexity $O(mn)$.

Algorithm 2.2: The first quadratic time algorithm for solving Büchi games.

```
Data: A Büchi game.
Function Solve(𝒢):
    X ← Attr_Eve(Win)
    if X = V then
        return V
    else
        Let 𝒢' = 𝒢 \ Attr_Adam(V \ X)
        return Solve(𝒢')
```

Let us see how uniform positional determinacy follows from Lemma 12. The shortest proof is by induction on the number of vertices, and remark that in both cases in Lemma 12 this number decreases.

A more instructive proof proceeds by unfolding the fixed point computation. Let $\mathscr{G}_0 = \mathscr{G}$ the original game, and applying the computation yields a sequence of subgames

$\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_p$. Let us write V_k for the set of vertices of $\mathcal{G}_k$, we have $\mathrm{Attr}^{\mathcal{G}_p}_{\mathrm{Eve}}(\mathrm{Win}) = V_p$. Thanks to the lemma, we have $W_{\mathrm{Eve}}(\mathcal{G}) = V_p$. The proof constructs a positional uniform winning strategy in $\mathcal{G}_p$ from $W_{\mathrm{Eve}}(\mathcal{G})$, which thanks to Lemma 11 induces a positional uniform winning strategy in $\mathcal{G}$ from $W_{\mathrm{Eve}}(\mathcal{G})$.

The case of Adam is a bit more complicated. For $v \notin V_k$, the rank of v is the smallest $k \in \mathbb{N}$ such that $v \in V \setminus V_k$. Equivalently, $v \in V_{k-1} \setminus V_k$.

For each k, let $\tau_{a,k}$ denote an attractor strategy ensuring to reach $V_k \setminus \mathrm{Attr}^{\mathcal{G}_k}_{\mathrm{Eve}}(\mathrm{Win})$ from $\mathrm{Attr}^{\mathcal{G}_k}_{\mathrm{Adam}}(V_k \setminus \mathrm{Attr}^{\mathcal{G}_k}_{\mathrm{Eve}}(\mathrm{Win}))$, and $\tau_{c,k}$ a counter-attractor strategy ensuring to never reach Win from $V_k \setminus \mathrm{Attr}^{\mathcal{G}_k}_{\mathrm{Eve}}(\mathrm{Win})$. We construct the strategy τ in $\mathcal{G}$ as the disjoint union of all $\tau_{a,k}$ and $\tau_{c,k}$:

$$\tau(v) = \begin{cases} \tau_{a,k}(v) & \text{if } \mathrm{rank}(v) = k \text{ and } v \notin V_k \setminus \mathrm{Attr}^{\mathcal{G}_k}_{\mathrm{Eve}}(\mathrm{Win}), \\ \tau_{c,k}(v) & \text{if } \mathrm{rank}(v) = k \text{ and } v \in V_k \setminus \mathrm{Attr}^{\mathcal{G}_k}_{\mathrm{Eve}}(\mathrm{Win}). \end{cases}$$

and argue that it ensures $\mathtt{CoBuchi}(\mathrm{Win})$. Note that τ is the disjoint union of positional strategies, it is positional. Consider a play consistent with τ starting from a vertex of rank k. In the first phase, the play is consistent with $\tau_{a,k}$. If we were playing in $\mathcal{G}_k$, this would go on until reaching $V \setminus \mathrm{Attr}^{\mathcal{G}_k}_{\mathrm{Eve}}(\mathrm{Win})$. But since we are here playing in $\mathcal{G}$, there is another possibility: that Eve chooses an edge leading outside of $\mathcal{G}_k$. In that case necessarily the next vertex is in $\mathcal{G}_{k-1}$, so it has smaller rank. In the second phase, which starts upon reaching $V \setminus \mathrm{Attr}^{\mathcal{G}_k}_{\mathrm{Eve}}(\mathrm{Win})$, the play is consistent with $\tau_{c,k}$, and again two things can happen. Either the play remains in $\mathcal{G}_k$, so it is consistent with $\tau_{c,k}$ forever, in which case it never sees Win and therefore satisfies $\mathtt{CoBuchi}(\mathrm{Win})$, or it exists $\mathcal{G}_k$, necessarily to reach $\mathcal{G}_{k-1}$, meaning that it reaches a vertex of smaller rank. Along any play consistent with τ the rank never increases, implying that it is eventually consistent with some $\tau_{c,k}$ hence satisfies $\mathtt{CoBuchi}(\mathrm{Win})$.

A second algorithm

The following lemma induces a different algorithm with the same complexity. We define the operator $\mathrm{Pre}^{\mathrm{Win}}_{\mathrm{Eve}}$ on subsets of vertices: for $Y \subseteq V$,

$$\mathrm{Pre}^{\mathrm{Win}}_{\mathrm{Eve}}(Y) = \left\{ v \in V_E : \exists v \xrightarrow{\mathrm{Win}} v' \in E, v' \in Y \right\} \cup \left\{ v \in V_A : \forall v \xrightarrow{\mathrm{Win}} v' \in E,\ v' \in Y \right\}.$$

Lemma 13 (Second fixed point characterisation of the winning region for Buchi games)**.** *Let $\mathcal{G}$ a Büchi game. Then $W_{\mathrm{Eve}}(\mathcal{G})$ is the greatest fixed point of the monotonic operator*

$$Y \mapsto Attr_{\mathrm{Eve}}\left(Pre^{Win}_{\mathrm{Eve}}(Y)\right).$$

Proof. Thanks to Theorem 4 the greatest fixed point is also the greatest post-fixed point, so we need to show two properties:

- $W_{\mathrm{Eve}}(\mathcal{G})$ is a post-fixed point, meaning $W_{\mathrm{Eve}}(\mathcal{G}) \subseteq \mathrm{Attr}_{\mathrm{Eve}}\left(\mathrm{Pre}^{\mathrm{Win}}_{\mathrm{Eve}}(W_{\mathrm{Eve}}(\mathcal{G}))\right)$.
- For all *post-fixed points* Y, we have $Y \subseteq W_{\mathrm{Eve}}(\mathcal{G})$.

We first show that $W_{\text{Eve}}(\mathscr{G}) \subseteq \text{Attr}_{\text{Eve}}\left(\text{Pre}^{\text{Win}}_{\text{Eve}}(W_{\text{Eve}}(\mathscr{G}))\right)$. We actually show that $V \setminus \text{Attr}_{\text{Eve}}\left(\text{Pre}^{\text{Win}}_{\text{Eve}}(W_{\text{Eve}}(\mathscr{G}))\right) \subseteq W_{\text{Adam}}(\mathscr{G})$, implying the inclusion by complementing. Let τ be a counter-attractor strategy ensuring never to reach $\text{Pre}^{\text{Win}}_{\text{Eve}}(W_{\text{Eve}}(\mathscr{G}))$ from $V \setminus \text{Attr}_{\text{Eve}}\left(\text{Pre}^{\text{Win}}_{\text{Eve}}(W_{\text{Eve}}(\mathscr{G}))\right)$. We additionally require that τ chooses Lose edges over Win edges whenever possible: for each $v \in V_{\text{Adam}} \cap (V \setminus \text{Attr}_{\text{Eve}}\left(\text{Pre}^{\text{Win}}_{\text{Eve}}(W_{\text{Eve}}(\mathscr{G}))\right))$, if there exists $v \xrightarrow{\text{Lose}} v'$ with $v' \notin \text{Attr}_{\text{Eve}}\left(\text{Pre}^{\text{Win}}_{\text{Eve}}(W_{\text{Eve}}(\mathscr{G}))\right)$, define $\tau(v) = v \xrightarrow{\text{Lose}} v'$.

Let τ' a winning strategy from $W_{\text{Adam}}(\mathscr{G})$. We play the following strategy: play τ' from $W_{\text{Adam}}(\mathscr{G})$ and τ otherwise. Let us consider a play consistent with this strategy from $V \setminus \text{Attr}_{\text{Eve}}\left(\text{Pre}^{\text{Win}}_{\text{Eve}}(W_{\text{Eve}}(\mathscr{G}))\right)$, and assume that it reaches an edge $v \xrightarrow{\text{Win}} v'$. If $v \in V_{\text{Eve}}$, since $v \notin \text{Pre}^{\text{Win}}_{\text{Eve}}(W_{\text{Eve}}(\mathscr{G}))$ this implies that $v' \in W_{\text{Adam}}(\mathscr{G})$. If $v \in V_{\text{Adam}}$, the additional property of τ' ensures that $v' \in W_{\text{Adam}}(\mathscr{G})$. Hence after reaching Win we switch to a winning strategy, so the strategy is winning from $V \setminus \text{Attr}_{\text{Eve}}\left(\text{Pre}^{\text{Win}}_{\text{Eve}}(W_{\text{Eve}}(\mathscr{G}))\right)$.

Let Y a post-fixed point, meaning $Y \subseteq \text{Attr}_{\text{Eve}}\left(\text{Pre}^{\text{Win}}_{\text{Eve}}(Y)\right)$. We show that $W_{\text{Eve}}(\mathscr{G}) \subseteq Y$. Let σ_a be an attractor strategy ensuring to reach $\text{Pre}^{\text{Win}}_{\text{Eve}}(Y)$ from Y. We also define a strategy σ_p: for $v \in V_{\text{Eve}}$, if $v \in \text{Pre}^{\text{Win}}_{\text{Eve}}(Y)$ there exists $v \xrightarrow{\text{Win}} v' \in E$ such that $v' \in Y$, let us define $\sigma_p(v) = v \to v'$. We define the strategy σ as follows:

$$\sigma(v) = \begin{cases} \sigma_a(v) & \text{if } v \in \text{Attr}_{\text{Eve}}(\text{Pre}^{\text{Win}}_{\text{Eve}}(Y)) \setminus \text{Pre}^{\text{Win}}_{\text{Eve}}(Y), \\ \sigma_p(v) & \text{if } v \in \text{Pre}^{\text{Win}}_{\text{Eve}}(Y). \end{cases}$$

We argue that σ ensures $\texttt{Buchi}(\text{Win})$ from Y. Indeed a play consistent with σ can be divided into infinitely many finite plays, each of them consistent with σ_a until reaching $\text{Pre}^{\text{Win}}_{\text{Eve}}(Y)$, then one step consistent with σ_p reaching Win, before starting from scratch in Y. □

Lemma 12 directly transfers to Algorithm 2.3. We could also obtain uniform positional determinacy from Lemma 13, using a similar unfolding as for Lemma 12.

Algorithm 2.3: The second quadratic time algorithm for solving Büchi games.

Data: A Büchi game.
$Y \leftarrow V$
repeat
  $Y \leftarrow \text{Attr}_{\text{Eve}}\left(\text{Pre}^{\text{Win}}_{\text{Eve}}(Y)\right)$
until $Y = Attr_{\text{Eve}}\left(Pre^{Win}_{\text{Eve}}(Y)\right)$;
return Y

Remark 8 (Comparison of the two algorithms)**.** *Both algorithms have the same complexity but they are not equivalent: the number of recursive calls of the first algorithm may be strictly smaller than the number of iterations of the repeat loop in the second algorithm. Both can be extended into (different) algorithms for parity games and beyond; in this chapter we will work with the first algorithm.*

2.3 Parity games

Recall that the *parity* objective extends Büchi and coBüchi objectives:

$$\texttt{Parity} = \{\rho \in [1,d]^\omega \mid \text{ the largest priority appearing infinitely often in } \rho \text{ is even}\}.$$

Theorem 15 (Positional determinacy and complexity of parity games)**.** *Parity objectives are uniformly positionally determined*[2]*. There exists an algorithm for computing the winning regions of parity games in exponential time, and more precisely of complexity $O(mn^{d-1})$. The space complexity of $O(nd)$.*

Furthermore, solving parity games is in $\mathsf{NP} \cap \mathsf{coNP}$.

To prove Theorem 15 we first construct a recursive algorithm for computing the winning regions of parity games. The algorithm is often called Zielonka's algorithm, or more accurately McNaughton Zielonka's algorithm. We refer to the reference section Section 2.6 for a discussion on this nomenclature. The $\mathsf{NP} \cap \mathsf{coNP}$ complexity bounds will be discussed at the end of this section.

The following lemma induces (half of) the recursive algorithm. Identifying a colour and its set of vertices we write d for the set of vertices of priority d.

Lemma 14 (Fixed point characterisation of the winning regions for parity games)**.** *Let $\mathcal{G}$ be a parity game with priorities in $[1,d]$, and d even. Let $\mathcal{G}' = \mathcal{G} \setminus Attr_{\text{Eve}}(d)$.*

- *If $W_{\text{Adam}}(\mathcal{G}') = \emptyset$, then $W_{\text{Eve}}(\mathcal{G}) = V$.*
- *If $W_{\text{Adam}}(\mathcal{G}') \neq \emptyset$, let $\mathcal{G}'' = \mathcal{G} \setminus Attr_{\text{Adam}}(W_{\text{Adam}}(\mathcal{G}'))$, then $W_{\text{Eve}}(\mathcal{G}) = W_{\text{Eve}}(\mathcal{G}'')$.*

Note that $\mathcal{G}'$ has priorities in $[1,d-1]$ and that if $W_{\text{Adam}}(\mathcal{G}') \neq \emptyset$, then $\mathcal{G}''$ has less vertices than $\mathcal{G}$.

Proof. We prove the first item. Let σ_d be an attractor strategy ensuring to reach d from $\text{Attr}_{\text{Eve}}(d)$. Consider a winning strategy for Eve from $V \setminus \text{Attr}_{\text{Eve}}(d)$ in $\mathcal{G}'$, it induces a strategy σ' in $\mathcal{G}$. We construct a strategy σ in $\mathcal{G}$ as the disjoint union of σ_d on $\text{Attr}_{\text{Eve}}(d)$ and of σ' on $V \setminus \text{Attr}_{\text{Eve}}(d)$. Any play consistent with σ either enters $\text{Attr}_{\text{Eve}}(d)$ infinitely many times, or eventually remains in $V \setminus \text{Attr}_{\text{Eve}}(d)$ and is eventually consistent with σ'. In the first case it sees infinitely many times d, which is even and maximal, hence satisfies $\texttt{Parity}$, and in the other case since σ' is winning the play satisfies $\texttt{Parity}$. Thus σ is winning from V.

We now look at the second item. Let τ_a denote an attractor strategy ensuring to reach $W_{\text{Adam}}(\mathcal{G}')$ from $\text{Attr}_{\text{Adam}}(W_{\text{Adam}}(\mathcal{G}'))$. Consider a winning strategy for Adam from $W_{\text{Adam}}(\mathcal{G}')$ in $\mathcal{G}'$, it induces a strategy τ' in $\mathcal{G}$. Thanks to Lemma 10 τ' is a winning strategy in $\mathcal{G}$. Consider now a winning strategy in the game $\mathcal{G}''$ from $W_{\text{Adam}}(\mathcal{G}'')$, it induces a strategy τ'' in $\mathcal{G}$. The set $V \setminus \text{Attr}_{\text{Adam}}(W_{\text{Adam}}(\mathcal{G}'))$ is not a trap for Eve, so we cannot conclude that τ'' is a winning strategy in $\mathcal{G}$, and it indeed may not be. We construct a strategy τ in $\mathcal{G}$ as the (disjoint) union of the strategy τ_a on $\text{Attr}_{\text{Adam}}(W_{\text{Adam}}(\mathcal{G}')) \setminus W_{\text{Adam}}(\mathcal{G}')$, the strategy τ' on $W_{\text{Adam}}(\mathcal{G}')$ and the strategy τ'' on $W_{\text{Adam}}(\mathcal{G}'')$. We argue that τ is winning from $\text{Attr}_{\text{Adam}}(W_{\text{Adam}}(\mathcal{G}')) \cup W_{\text{Adam}}(\mathcal{G}'')$ in $\mathcal{G}$.

[2] See Remark 6 for the case of infinite games.

Indeed, any play consistent with this strategy in $\mathcal{G}$ either stays forever in $W_{\text{Adam}}(\mathcal{G}'')$ hence is consistent with τ'' or enters $\text{Attr}_{\text{Adam}}(W_{\text{Adam}}(\mathcal{G}'))$, hence is eventually consistent with τ'. In both cases this implies that the play is winning. Thus we have proved that $\text{Attr}_{\text{Adam}}(W_{\text{Adam}}(\mathcal{G}')) \cup W_{\text{Adam}}(\mathcal{G}'') \subseteq W_{\text{Adam}}(\mathcal{G})$.

We now show that $W_{\text{Eve}}(\mathcal{G}'') \subseteq W_{\text{Eve}}(\mathcal{G})$, which implies the converse inclusion. Consider a winning strategy from $W_{\text{Eve}}(\mathcal{G}'')$ in $\mathcal{G}''$, it induces a strategy σ in $\mathcal{G}$. Thanks to Lemma 11, any play consistent with σ stays forever in $W_{\text{Eve}}(\mathcal{G}'')$, implying that σ is winning from $W_{\text{Eve}}(\mathcal{G}'')$ in $\mathcal{G}$. □

To get the full algorithm we need the analogous lemma for the case where the maximal priority is odd. We do not prove the following lemma as it is the exact dual of the previous lemma, and the proof is the same swapping the two players.

Lemma 15 (Dual fixed point characterisation of the winning regions for parity games)**.** *Let $\mathcal{G}$ be a parity game with priorities in $[1,d]$, and d odd. Let $\mathcal{G}' = \mathcal{G} \setminus Attr_{\text{Adam}}(d)$.*

- *If $W_{\text{Eve}}(\mathcal{G}') = \emptyset$, then $W_{\text{Adam}}(\mathcal{G}) = V$.*
- *If $W_{\text{Eve}}(\mathcal{G}') \neq \emptyset$, let $\mathcal{G}'' = \mathcal{G} \setminus Attr_{\text{Eve}}(W_{\text{Eve}}(\mathcal{G}'))$, then $W_{\text{Adam}}(\mathcal{G}) = W_{\text{Adam}}(\mathcal{G}'')$.*

The algorithm is presented in pseudocode in Algorithm 2.4.

The proofs of Lemma 14 and Lemma 15 also imply that parity games are positionally determined. Indeed, winning strategies are defined as disjoint unions of strategies constructed inductively.

We now perform a complexity analysis. Let us write $f(n,d)$ for the number of recursive calls performed by the algorithm on parity games with n vertices and priorities in $[1,d]$. We have $f(n,1) = f(0,d) = 0$, with the general induction:

$$f(n,d) \leq f(n,d-1) + f(n-1,d) + 2.$$

The term $f(n,d-1)$ corresponds to the recursive call to $\mathcal{G}'$ and the term $f(n-1,d)$ to the recursive call to $\mathcal{G}''$. We obtain $f(n,d) \leq n \cdot f(n,d-1) + 2n$, so $f(n,d) \leq 2n(1+n+\cdots+n^{d-2}) = O(n^{d-1})$. In each recursive call we perform two attractor computations so the number of operations in one recursive call is $O(m)$. Thus the overall time complexity is $O(mn^{d-1})$.

We finish the proof of Theorem 15 by sketching the argument that solving parity games is in $\mathsf{NP} \cap \mathsf{coNP}$. The first observation is that computing the winning regions of the one player variants of parity games can be done in polynomial time through a simple graph analysis that we do not detail here. The NP and coNP algorithms are the following: guess a winning positional strategy, and check whether it is winning by computing the winning regions of the one player game induced by the strategy. Guessing a strategy for Eve is a witness that the answer is yes so it yields an NP algorithm, and guessing a strategy for Adam yields a coNP algorithm.

Chapter 3 is devoted to the study of advanced algorithms for parity games.

Algorithm 2.4: A recursive algorithm for computing the winning regions of parity games.

Data: A parity game $\mathcal{G}$ with priorities in $[1,d]$

Function `SolveEven(`$\mathcal{G}$`)`**:**

- $\mathcal{G}' \leftarrow \mathcal{G} \setminus \mathrm{Attr}^{\mathcal{G}}_{\mathrm{Eve}}(d)$
- $X \leftarrow$ `SolveOdd(`$\mathcal{G}'$`)` `// `$\mathcal{G}'$ `has one less priority`
- **if** $X = \emptyset$ **then**
  - **return** V
- **else**
  - $\mathcal{G}'' \leftarrow \mathcal{G} \setminus \mathrm{Attr}^{\mathcal{G}}_{\mathrm{Adam}}(X)$
  - **return** `SolveEven(`$\mathcal{G}''$`)` `//` $\mathcal{G}''$ `has less vertices`

Function `SolveOdd(`$\mathcal{G}$`)`**:**

- **if** $d = 1$ **then**
  - **return** V
- $\mathcal{G}' \leftarrow \mathcal{G} \setminus \mathrm{Attr}^{\mathcal{G}}_{\mathrm{Adam}}(d)$
- $X \leftarrow$ `SolveEven(`$\mathcal{G}'$`)` `//` $\mathcal{G}'$ `has one less priority`
- **if** $X = \emptyset$ **then**
  - **return** V
- **else**
  - $\mathcal{G}'' \leftarrow \mathcal{G} \setminus \mathrm{Attr}^{\mathcal{G}}_{\mathrm{Eve}}(X)$
  - **return** `SolveOdd(`$\mathcal{G}''$`)` `//` $\mathcal{G}''$ `has less vertices`

if *d is even* **then**

- `SolveEven(`$\mathcal{G}$`)`

else

- `SolveOdd(`$\mathcal{G}$`)`

2.4 Proving half-positional determinacy for qualitative games

We define here a generic approach to proving half-positional determinacy results for qualitative objectives. Let us say that an objective Ω is *submixing* if:

$$\begin{array}{lccccccccccc}
\text{if} & \rho_1 & = & \rho_1^0 & & \rho_1^1 & & \cdots & \rho_1^\ell & & \cdots & \notin \Omega \\
\text{and} & \rho_2 & = & & \rho_2^0 & & \rho_2^1 & \cdots & & \rho_2^\ell & \cdots & \notin \Omega, \\
\text{then:} & \rho_1 \bowtie \rho_2 & = & \rho_1^0 & \rho_2^0 & \rho_1^1 & \rho_2^1 & \cdots & \rho_1^\ell & \rho_2^\ell & \cdots & \notin \Omega.
\end{array}$$

Theorem 16 (Submixing property implies uniform half-positional determinacy)**.** *Every prefix independent submixing objective is uniformly half-positionally determined over finite arenas.*

Proof. We proceed by induction over the following quantity: total outdegree of vertices controlled by Eve minus number of vertices controller by Eve. Since we assume that every vertex has an outgoing edge, the base case is when each vertex of Eve has only one successor. In that case Eve has only one strategy and it is positional, so the property holds.

In the inductive step, we consider a game $\mathcal{G}$ where Eve has a winning strategy σ. Let $v \in V_{\text{Eve}}$ with at least two successors. We partition the outgoing edges of v in two non-empty subsets which we call E_1^v and E_2^v. Let us define two games $\mathcal{G}_1$ and $\mathcal{G}_2$: the game $\mathcal{G}_1$ is obtained from $\mathcal{G}$ by removing the edges from E_2^v, and symmetrically for $\mathcal{G}_2$.

We claim that Eve has a winning strategy in either $\mathcal{G}_1$ or $\mathcal{G}_2$. Let us assume towards contradiction that this is not the case: then there exist τ_1 and τ_2 two strategies for Adam which are winning in $\mathcal{G}_1$ and $\mathcal{G}_2$ respectively. We construct a strategy τ for Adam in $\mathcal{G}$ as follows: it has two modes, 1 and 2. The initial mode is 1, and the strategy simulates τ_1 from the mode 1 and τ_2 from the mode 2. Whenever v is visited, the mode is adjusted: if the outgoing edge is in E_1^v then the new mode is 1, otherwise it is 2. To be more specific: when simulating τ_1 we play ignoring the parts of the play using mode 2, so removing them yields a play consistent with τ_1. The same goes for τ_2.

Consider a play π consistent with σ and τ. Since σ is winning, the play π is winning. It can be decomposed following which mode the play is in:

$$\begin{array}{ll}
\text{mode 1} & \overbrace{v_0 \cdots v}^{\pi_1^0} \qquad \overbrace{v \cdots v}^{\pi_1^1} \qquad \cdots \\
\text{mode 2} & \qquad\quad \underbrace{v \cdots v}_{\pi_2^0} \qquad\quad \underbrace{v \cdots v}_{\pi_2^1} \quad \cdots
\end{array}$$

where $\pi_1 = \pi_1^0 \pi_1^1 \cdots$ is consistent with τ_1 and $\pi_2 = \pi_2^0 \pi_2^1 \cdots$ is consistent with τ_2. Since τ_1 and τ_2 are winning strategies for Adam, π_1 and π_2 do not satisfy Ω.

There are two cases: the decomposition is either finite or infinite. If it is finite we get a contradiction: since π is winning and Ω is prefix independent any suffix of π is winning as well, contradicting that it is consistent with either τ_1 or τ_2 hence cannot be winning. In the second case, the submixing property directly yields a contradiction: neither π_1 nor π_2 satisfy Ω, yet their shuffle π does. □

2.5 Rabin, Streett, and Muller games

The prefix independent objectives we studied so far are Büchi, CoBüchi, and their joint extension the parity objectives. The definition of the latter may seem a bit arbitrary; the study of Muller objectives will show how parity objectives naturally emerge as a well-behaved class of objectives.

Let us start with a very general class of infinitary objectives, where infinitary means that the objective only considers the set of colours appearing infinitely many times. For a sequence ρ, we let $\texttt{Inf}(\rho)$ denote the set of colours appearing infinitely many times in ρ. The *Muller* objective is over the set of colours $C = [1,d]$ and is parametrised by some $\mathscr{F} \subseteq 2^C$, *i.e.* a family of subsets of C. The objective is defined as follows:

$$\texttt{Muller}(\mathscr{F}) = \{\rho \in C^\omega : \texttt{Inf}(\rho) \in \mathscr{F}\}.$$

Muller objectives include any objective specifying the set of colours appearing infinitely often. There are different possible representations for a Muller objective, for instance using logical formulas or circuits. We will here consider the most natural one which simply consists in listing the elements of $\mathscr{F}$. Note that $\mathscr{F}$ can have size up to 2^{2^d}, and each element of $\mathscr{F}$ (which is a subset of C) requires up to d bits to be identified, so the representation of $\mathscr{F}$ can be very large.

We note that the complement of a Muller objective is another Muller objective: $C^\omega \setminus \texttt{Muller}(\mathscr{F}) = \texttt{Muller}(2^C \setminus \mathscr{F})$. In particular if Eve has a Muller objective then Adam also has a Muller objective.

To define subclasses of Muller objectives we make assumptions on $\mathscr{F} \subseteq 2^C$. We say that $\mathscr{F}$ is closed under union if whenever $X, Y \in \mathscr{F}$ then $X \cup Y \in \mathscr{F}$. Let us define *Streett* objectives as the subclass of Muller objectives given by $\mathscr{F}$ closed under union. The following purely combinatorial lemma gives a nice characterisation of these objectives.

Lemma 16 (Characterisation of Streett among Muller objectives)**.** *A collection $\mathscr{F} \subseteq 2^C$ is closed under union if and only if there exists a set of pairs $(R_i, G_i)_{i\in[1,d]}$ with $R_i, G_i \subseteq C$ such that $X \in \mathscr{F}$ is equivalent to for all $i \in [1,d]$, if $X \cap R_i \neq \emptyset$ then $X \cap G_i \neq \emptyset$.*

We will see in Section 2.6 a natural and optimised way to construct these pairs using the Zielonka tree. In the meantime let us give a direct proof of this result.

Proof. Let $\mathscr{F}$ closed under union. We note that for any $S \notin \mathscr{F}$, either no subsets of S are in $\mathscr{F}$ or there exists a maximal subset S' of S in $\mathscr{F}$: indeed it is the union of all subsets of S in $\mathscr{F}$. It directly follows that for a subset X we have the following equivalence: $X \in \mathscr{F}$ if and only if for any $S \notin \mathscr{F}$, if $X \subseteq S$ then $X \subseteq S'$. This is rewritten equivalently as: if $X \cap (C \setminus S') \neq \emptyset$ then $X \cap (C \setminus S) \neq \emptyset$. Hence a suitable set of pairs satisfying the required property is $\{(C \setminus S', C \setminus S) : S \notin \mathscr{F}\}$. □

Thanks to this lemma we can give a direct definition of Streett objectives. The set of colours is $C = [1,d]$, and we consider a family of subsets $G_1, R_1, \ldots, G_d, R_d \subseteq C$.

$$\texttt{Streett} = \{\rho \in C^\omega : \forall i \in [1,d],\ R_i \cap \texttt{Inf}(\rho) \neq \emptyset \implies G_i \cap \texttt{Inf}(\rho) \neq \emptyset\}.$$

It is customary to call R_i the i^{th} request and G_i the corresponding response; with this terminology the Streett objective requires that every request made infinitely many times must be responded to infinitely many times.

The *Rabin* objectives are the complement of the Streett objectives:

$$\texttt{Rabin} = \{\rho \in C^\omega : \exists i \in [1,d],\ R_i \cap \texttt{Inf}(\rho) \neq \emptyset \wedge G_i \cap \texttt{Inf}(\rho) = \emptyset\}.$$

McNaughton algorithm: an exponential time algorithm for Muller games

Theorem 17 (Finite memory determinacy and complexity for Muller games)**.** *Muller objectives are determined with finite memory strategies of size $d!$[3]. There exists an algorithm for computing the winning regions of Muller games in exponential time, and more specifically of complexity $O(dm(dn)^{d-1})$, and in polynomial space, and more specifically $O(dm)$.*

The complexity will be improved later in this chapter. The presentation of the recursive algorithm for computing the winning regions of Muller games follows the exact same lines as for parity games: indeed, the Muller objective extends the parity objective, and specialising the algorithm for Muller games to parity games yields the algorithm we presented above.

The following lemma induces the recursive algorithm for computing the winning regions of Muller games.

Lemma 17 (Fixed point characterisation of the winning regions for Muller games)**.** *Let $\mathcal{G}$ be a Muller game such that $C \in \mathcal{F}$. For each $c \in C$, let $\mathcal{G}_c = \mathcal{G} \setminus Attr_{\text{Eve}}(c)$.*

- *If for all $c \in C$, we have $W_{\text{Adam}}(\mathcal{G}_c) = \emptyset$, then $W_{\text{Eve}}(\mathcal{G}) = V$.*
- *If there exists $c \in C$ such that $W_{\text{Adam}}(\mathcal{G}_c) \neq \emptyset$, let $\mathcal{G}' = \mathcal{G} \setminus Attr_{\text{Adam}}(W_{\text{Adam}}(\mathcal{G}_c))$, then $W_{\text{Eve}}(\mathcal{G}) = W_{\text{Eve}}(\mathcal{G}')$.*

Proof. We prove the first item.

For each $c \in C$, let σ_c be an attractor strategy ensuring to reach c from $\text{Attr}_{\text{Eve}}(c)$, and consider a winning strategy for Eve from $V \setminus \text{Attr}_{\text{Eve}}(c)$ in $\mathcal{G}_c$, it induces a strategy σ'_c in $\mathcal{G}$. We construct a strategy σ in $\mathcal{G}$ which will simulate the strategies above in turn; to do so it uses C as top-level memory states. (We note that the strategies σ'_c may use memory as well, so σ may actually use more memory than just C.) The strategy σ with memory c simulates σ_c from $\text{Attr}_{\text{Eve}}(c)$ and σ'_c from $V \setminus \text{Attr}_{\text{Eve}}(c)$, and if it ever reaches c it updates its memory state to $c+1$ and 1 if $c = d$. Any play consistent with σ either updates its memory state infinitely many times, or eventually remains in $V \setminus \text{Attr}_{\text{Eve}}(c)$ and is eventually consistent with σ'_c. In the first case it sees each colour infinitely many times, and since $C \in \mathcal{F}$ the play satisfies $\texttt{Muller}(\mathcal{F})$, and in the other case since σ'_c is winning the play satisfies $\texttt{Muller}(\mathcal{F})$. Thus σ is winning from V.

We now look at the second item.

Let τ_a denote an attractor strategy from $\text{Attr}_{\text{Adam}}(W_{\text{Adam}}(\mathcal{G}_c)) \setminus W_{\text{Adam}}(\mathcal{G}_c)$. Consider a winning strategy for Adam from $W_{\text{Adam}}(\mathcal{G}_c)$ in $\mathcal{G}_c$, it induces a strategy τ_c in

[3] See Remark 6 for the case of infinite games.

$\mathcal{G}$. Thanks to Lemma 10, this implies that τ_c is a winning strategy in $\mathcal{G}$. Consider now a winning strategy in the game $\mathcal{G}'$ from $W_{\text{Adam}}(\mathcal{G}')$, it induces a strategy τ' in $\mathcal{G}$. The set $V \setminus \text{Attr}_{\text{Adam}}(W_{\text{Adam}}(\mathcal{G}_c))$ may not be a trap for Eve, so we cannot conclude that τ' is a winning strategy in $\mathcal{G}$, and it indeed may not be. We construct a strategy τ in $\mathcal{G}$ as the (disjoint) union of the strategy τ_a on $\text{Attr}_{\text{Adam}}(W_{\text{Adam}}(\mathcal{G}_c)) \setminus W_{\text{Adam}}(\mathcal{G}_c)$, the strategy τ_c on $W_{\text{Adam}}(\mathcal{G}_c)$ and the strategy τ' on $W_{\text{Adam}}(\mathcal{G}')$. We argue that τ is winning from $\text{Attr}_{\text{Adam}}(W_{\text{Adam}}(\mathcal{G}_c)) \cup W_{\text{Adam}}(\mathcal{G}')$ in $\mathcal{G}$. Indeed, any play consistent with this strategy in $\mathcal{G}$ either stays forever in $W_{\text{Adam}}(\mathcal{G}')$ hence is consistent with τ' or enters $\text{Attr}_{\text{Adam}}(W_{\text{Adam}}(\mathcal{G}_c))$, so it is eventually consistent with τ_c. In both cases this implies that the play is winning. Thus we have proved that $\text{Attr}_{\text{Adam}}(W_{\text{Adam}}(\mathcal{G}_c)) \cup W_{\text{Adam}}(\mathcal{G}') \subseteq W_{\text{Adam}}(\mathcal{G})$.

We now show that $W_{\text{Eve}}(\mathcal{G}') \subseteq W_{\text{Eve}}(\mathcal{G})$, which implies the converse inclusion. Consider a winning strategy from $W_{\text{Eve}}(\mathcal{G}')$ in $\mathcal{G}'$, it induces a strategy σ in $\mathcal{G}$. Thanks to Lemma 11, any play consistent with σ stays forever in $W_{\text{Eve}}(\mathcal{G}')$, implying that σ is winning from $W_{\text{Eve}}(\mathcal{G}')$ in $\mathcal{G}$. □

To get the full algorithm we need the analogous lemma for the case where $C \notin \mathcal{F}$. We do not prove it as it is the exact dual of the previous lemma, and the proof is the same swapping the two players.

Lemma 18 (Dual fixed point characterisation of the winning regions for Muller games)**.** *Let $\mathcal{G}$ be a Muller game such that $C \notin \mathcal{F}$. For each $c \in C$, let $\mathcal{G}_c = \mathcal{G} \setminus Attr_{\text{Adam}}(c)$.*

- *If for all $c \in C$, we have $W_{\text{Eve}}(\mathcal{G}_c) = \emptyset$, then $W_{\text{Adam}}(\mathcal{G}) = V$.*
- *If there exists $c \in C$ such that $W_{\text{Eve}}(\mathcal{G}_c) \neq \emptyset$, let $\mathcal{G}' = \mathcal{G} \setminus Attr_{\text{Eve}}(W_{\text{Eve}}(\mathcal{G}_c))$, then $W_{\text{Adam}}(\mathcal{G}) = W_{\text{Adam}}(\mathcal{G}')$.*

The algorithm is presented in pseudocode in Algorithm 2.5. We only give the case where $C \in \mathcal{F}$, the other case being symmetric. The base case is when there is only one colour c, in which case Eve wins everywhere if $\mathcal{F} = \{c\}$ and Adam wins everywhere if $\mathcal{F} = \emptyset$.

We now perform a complexity analysis of the algorithm. Let us write $f(n,d)$ for the number of recursive calls performed by the algorithm on Muller games with n vertices and d colours. We have $f(n,1) = f(0,d) = 0$, with the general induction:

$$f(n,d) \leq d \cdot f(n,d-1) + f(n-1,d) + d + 1.$$

The term $d \cdot f(n,d-1)$ corresponds to the recursive calls to $\mathcal{G}_c$ for each $c \in C$ and the term $f(n-1,d)$ to the recursive call to $\mathcal{G}'$. We obtain $f(n,d) \leq dn \cdot f(n,d-1) + (d+1)n$, so $f(n,d) \leq (d+1)n(1+dn+(dn)^2+\cdots+(dn)^{d-2}) = O((dn)^{d-1})$. In each recursive call we perform $d+1$ attractor computations so the number of operations in one recursive call is $O(dm)$. Thus the overall time complexity is $O(dm(dn)^{d-1})$.

The proofs of Lemma 17 and Lemma 18 also imply that Muller games are determined with finite memory of size $d!$. We do not make it more precise here because an improved analysis of the memory requirements will be conducted in Section 2.6 using a variant of this algorithm.

Algorithm 2.5: A recursive algorithm for computing the winning regions of Muller games.

Data: A Muller game $\mathcal{G}$ over C
Function `SolveIn(`$\mathcal{G}$`)`**:**
  `// Assumes` $C \in \mathcal{F}$
  if $C = \{c\}$ **then**
    return V
  for $c \in C$ **do**
    $\mathcal{G}_c \leftarrow \mathcal{G} \setminus \mathrm{Attr}^{\mathcal{G}}_{\mathrm{Eve}}(c)$
    if $C \setminus \{c\} \in \mathcal{F}$ **then**
      $W_{\mathrm{Eve}}(\mathcal{G}_c) \leftarrow$ `SolveIn(`$\mathcal{G}_c$`)` `// ` $\mathcal{G}_c$ `has one colour less`
    else
      $W_{\mathrm{Eve}}(\mathcal{G}_c) \leftarrow$ `SolveOut(`$\mathcal{G}_c$`)` `// ` $\mathcal{G}_c$ `has one colour less`
  if $\forall c \in C, W_{\mathrm{Adam}}(\mathcal{G}_c) = \emptyset$ **then**
    return V
  else
    Let c such that $W_{\mathrm{Adam}}(\mathcal{G}_c) \neq \emptyset$
    $\mathcal{G}' \leftarrow \mathcal{G} \setminus \mathrm{Attr}^{\mathcal{G}}_{\mathrm{Adam}}(W_{\mathrm{Adam}}(\mathcal{G}_c))$
    return `SolveIn(`$\mathcal{G}'$`)` `// ` $\mathcal{G}'$ `has less vertices`

Function `SolveOut(`$\mathcal{G}$`)`**:**
  `// Symmetric to SolveIn, assumes` $C \notin \mathcal{F}$

if $C \in \mathcal{F}$ **then**
  `SolveIn(`$\mathcal{G}$`)`
else
  `SolveOut(`$\mathcal{G}$`)`

Half-positional determinacy for Rabin games

Theorem 18 (Half-positional determinacy for Rabin games)**.** *Rabin games are uniformly half-positionally determined.*

Theorem 18 is a direct corollary of Theorem 16, since the Rabin objective is prefix independent and submixing. Indeed, let $(\rho_1^\ell)_{\ell\in\mathbb{N}}$ and $(\rho_2^\ell)_{\ell\in\mathbb{N}}$ such that:

$$\begin{array}{lccccccccccc}
 & \rho_1 & = & \rho_1^0 & & \rho_1^1 & & \cdots & \rho_1^\ell & & \cdots & \notin \texttt{Rabin} \\
\text{and} & \rho_2 & = & & \rho_2^0 & & \rho_2^1 & \cdots & & \rho_2^\ell & \cdots & \notin \texttt{Rabin}, \\
\text{then:} & \rho_1 \bowtie \rho_2 & = & \rho_1^0 & \rho_2^0 & \rho_1^1 & \rho_2^1 & \cdots & \rho_1^\ell & \rho_2^\ell & \cdots & \notin \texttt{Rabin}.
\end{array}$$

Since neither ρ_1 nor ρ_2 satisfy `Rabin`, in both for all $i \in [1,d]$ if $R_i \cap \texttt{Inf}(\rho) \neq \emptyset$, then $G_i \cap \texttt{Inf}(\rho) \neq \emptyset$. Since $\texttt{Inf}(\rho_1 \bowtie \rho_2) = \texttt{Inf}(\rho_1) \cup \texttt{Inf}(\rho_2)$, this implies that $\rho_1 \bowtie \rho_2$ does not satisfy `Rabin`.

Theorem 18 holds for infinite games. However the proof using the submixing property only applies to finite games and does not easily extend to infinite ones. A different approach is required to obtain the positional determinacy result for infinite games, see Theorem 25.

The complexity of solving Rabin games

Theorem 19 (Complexity of solving Rabin games)**.** *Solving Rabin games is* NP*-complete.*

Proof. The proof that solving Rabin games is in NP follows the same lines as for solving parity games: the algorithm guesses a positional strategy and checks whether it is indeed winning. This requires proving that solving Rabin games where Adam control all vertices can be done in polynomial time, which is indeed true and easy to see so we will not elaborate further on this.

To prove the NP-hardness we reduce the satisfiability problem for boolean formulas in conjunctive normal form (SAT) to solving Rabin games.

Let Φ be a formula in conjunctive normal form with n variables $x_1 \dots x_n$ and m clauses $C_1 \dots C_m$, where each C_j is of the form $\ell_{j_1} \vee \ell_{j_2} \vee \ell_{j_3}$:

$$\Phi = \bigwedge_{j=1}^{m} \ell_{j_1} \vee \ell_{j_2} \vee \ell_{j_3}.$$

A literal ℓ is either a variable x or its negation $\bar{x}$, and we write $\bar{\ell}$ for the negation of a literal. The question whether Φ is satisfiable reads: does there exist a valuation $\mathbf{v} : \{x_1, \dots, x_n\} \to \{0,1\}$ satisfying Φ.

We construct a Rabin game $\mathcal{G}$ with $m+1$ vertices (one per clause, all controlled by Eve, plus a unique vertex controlled by Adam), $4m$ edges (4 per clause), and $2n$ Rabin pairs (one per literal). We will show that the formula Φ is satisfiable if and only if Eve has a winning strategy in the Rabin game $\mathcal{G}$.

We first describe the Rabin condition. There is a Rabin pair (R_ℓ, G_ℓ) for each literal ℓ, so the Rabin condition requires that there exists a literal ℓ such that R_ℓ is visited infinitely many times and G_ℓ is not. Let us now describe the arena. A play

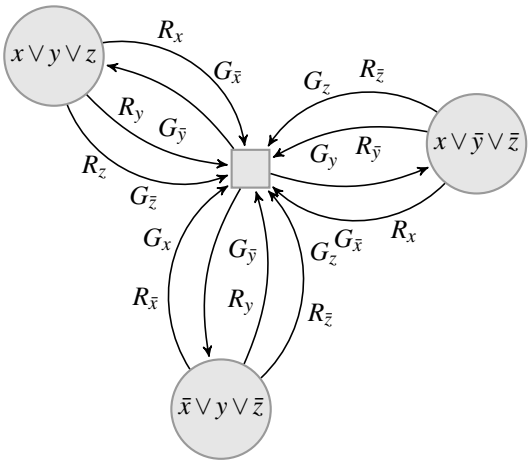


Figure 2.1: The Rabin game for $\Phi = (x \vee y \vee z) \bigwedge (x \vee \bar{y} \vee \bar{z}) \bigwedge (\bar{x} \vee y \vee \bar{z})$.

consists in an infinite sequence of rounds, where in each round first Adam chooses a clause and second Eve chooses a literal in this clause. When Eve chooses a literal ℓ she visits R_ℓ and $G_{\bar{\ell}}$. This completes the description of the Rabin game $\mathscr{G}$, it is illustrated in Figure 2.1. Let us now prove that this yields a reduction from SAT to solving Rabin games.

Let us first assume that Φ is satisfiable, and let $\mathbf{v}$ be a satisfying assignment: there is a literal ℓ in each clause satisfied by $\mathbf{v}$. Let σ be the memoryless strategy choosing such a literal in each clause. We argue that in any play consistent with σ there is at least one literal ℓ that Eve chooses infinitely many times without ever choosing $\bar{\ell}$: this implies that R_ℓ is visited infinitely often and G_ℓ is not. Indeed, some clause is chosen infinitely many times, so the corresponding literal chosen by Eve is also chosen infinitely many times. Since all the literals chosen by Eve satisfy the same assignment $\mathbf{v}$ she does not choose both a literal and its negation, so she never chooses $\bar{\ell}$. It follows that σ is a winning strategy for Eve.

Conversely, let us assume that Eve has a winning strategy. Thanks to Theorem 18 she has a positional winning strategy σ. We argue that σ cannot choose some literal ℓ in some clause C and the literal $\bar{\ell}$ in another clause C'. If this would be the case, consider the strategy of Adam alternating between the two clauses C and C' and play it against σ: both ℓ and $\bar{\ell}$ are chosen infinitely many times, and no other literals. Hence $R_\ell, G_\ell, R_{\bar{\ell}}$, and $G_{\bar{\ell}}$ are all visited infinitely many times, implying that this play does not satisfy `Rabin`, contradicting that σ is winning.

There exists a valuation $\mathbf{v}$ which satisfies each literal chosen by Eve, implying that it satisfies Φ which is then satisfiable. □

The complexity of solving Muller games

Theorem 20 (Complexity of solving Muller games)**.** *Solving Muller games is* PSPACE*-complete.*

As for the previous reduction, in the Muller game constructed in the reduction below we label edges rather than vertices, and some edges have more than one colour. As for Rabin games this can be reduced to the original definition of colouring functions (labelling vertices with exactly one colour each) with a polynomial increase in size.

Proof. The PSPACE algorithm was constructed in Theorem 17.

To prove the PSPACE-hardness we reduce the evaluation of quantified boolean formulas in disjunctive normal form (QBF) to solving Muller games.

Let Ψ be a quantified boolean formula in disjunctive normal form with n variables $x_1 \dots x_n$ and m clauses $C_1 \dots C_m$, where each C_j is of the form $\ell_{j_1} \wedge \ell_{j_2} \wedge \ell_{j_3}$:

$$\Psi = \exists x_1, \forall x_2, \dots, \exists x_n,\ \Phi(x_1, \dots, x_n) \text{ and } \Phi(x_1, \dots, x_n) = \bigvee_{j=1}^{m} \ell_{j_1} \wedge \ell_{j_2} \wedge \ell_{j_3}.$$

We construct a Muller game $\mathscr{G}$ with $m+1$ vertices (one per clause, all controlled by Adam, plus a unique vertex controlled by Eve), $4m$ edges (4 per clause), and $2n$ colours (one per literal). We will show that the formula Ψ evaluates to true if and only if Eve has a winning strategy in the Muller game $\mathscr{G}$.

We first describe the Muller condition. The set of colours is the set of literals. We let x denote the lowest quantified variable such that x or $\bar{x}$ is visited infinitely many times. The Muller condition requires that:

- either x is existential and only one of x and $\bar{x}$ is visited infinitely many times,
- or x is universal and both x and $\bar{x}$ are visited infinitely many times,

and for all variables y quantified after x, both y and $\bar{y}$ are visited infinitely many times. Formally, let $S_{>i} = \{x_q, \bar{x_q} : q > p\}$ and:

$$\mathscr{F} = \{S_{>p},\ \{x_p\} \cup S_{>p},\ \{\bar{x_p}\} \cup S_{>p} : x_p \text{ existential}\} \cup \{S_{\geq p} : x_p \text{ universal}\}.$$

Note that $\mathscr{F}$ contains $O(n)$ elements.

Let us now describe the arena. A play consists in an infinite sequence of rounds, where in each round first Eve chooses a clause and second Adam chooses a literal ℓ in this clause corresponding to some variable x_p, and visits the colour ℓ as well as each colour in $S_{>p}$.

The reduction is illustrated in Figure 2.2. Note that the edges from the vertex controlled by Eve to the other ones do not have a colour, which does not fit our definitions. For this reason we introduce a new colour c and colour all these edges by c. We define a new Muller objective by adding c to each set in $\mathscr{F}$: since every play in the game visit c infinitely many times, the two games are equivalent. We note that this construction works for this particular game but not in general.

For a valuation $\mathbf{v} : \{x_1, \ldots, x_n\} \to \{0,1\}$ and $p \in [1,n]$, we write $\Psi_{\mathbf{v},p}$ for the formula obtained from Ψ by fixing the variables $x_1, \ldots, x_{p-1}$ to $\mathbf{v}(x_1), \ldots, \mathbf{v}(x_{p-1})$ and quantifying only over the remaining variables. Let us say that a valuation $\mathbf{v}$ is *positive* if for every $p \in [1,n]$, the formula $\Psi_{\mathbf{v},p}$ evaluates to true, and similarly a valuation is *negative* if for every $p \in [1,n]$, the formula $\Psi_{\mathbf{v},p}$ evaluates to false.

Let us first assume that Ψ evaluates to true. We construct a winning strategy σ for Eve. It uses *positive* valuations over the variables $x_1, \ldots, x_n$ as memory states. Note that the fact that Ψ evaluates to true implies that there exists a positive valuation. Let us choose an arbitrary positive valuation as initial valuation. We first explain what the strategy σ does and then how to update its memory.

Assume that the current valuation is $\mathbf{v}$, since it is positive there exists a clause satisfying $\mathbf{v}$, the strategy σ chooses such a clause. Therefore, any literal that Adam chooses is necessarily true under $\mathbf{v}$.

The memory is updated as follows: assume that the current valuation is $\mathbf{v}$ and that Adam chose a literal corresponding to the variable x_p. If x_p is existential the valuation is unchanged. If x_p is universal, we construct a new positive valuation as follows. We swap the value of x_p in $\mathbf{v}$ and write $\mathbf{v}[x_p]$ for this new valuation. Since $\mathbf{v}$ is positive and x_p is universally quantified, the formula $\Psi_{\mathbf{v}[x_p],p+1}$ evaluates to true, so there exists a positive valuation $\mathbf{v}_{p+1} : \{x_{p+1}, \ldots, x_n\} \to \{0,1\}$ for this formula. The new valuation is defined as follows:

$$\mathbf{v}'(x_q) = \begin{cases} \mathbf{v}(x_q) & \text{if } q < p, \\ \overline{\mathbf{v}(x_q)} & \text{if } q = p, \\ \mathbf{v}_{p+1}(x_q) & \text{if } q > p, \end{cases}$$

it is positive by construction.

Let π be a play consistent with σ and x_p be the lowest quantified variable chosen infinitely many times by Adam. First, all colours in $S_{>p}$ are visited infinitely many times (when visiting x or $\bar{x}$). Let us look at the sequence $(\mathbf{v}_i(x_p))_{i \in \mathbb{N}}$ where $\mathbf{v}_i$ is the valuation in the i^{th} round. If x_p is existential, the sequence is ultimately constant as it can only change when a lower quantified variable is visited. If x_p is universal, the value changes each time the variable x_p is chosen. Since any literal that Adam chooses is necessarily true under the current valuation, this implies that in both cases π satisfies $\texttt{Muller}(\mathscr{F})$.

For the converse implication we show that if Ψ evaluates to false, then there exists a winning strategy τ for Adam. The construction is similar but using *negative* valuations. The memory states are negative valuations. The initial valuation is any negative valuation. If the current valuation is $\mathbf{v}$ and Eve chose the clause C, since the valuation is negative $\mathbf{v}$ does not satisfy C, the strategy τ chooses a literal in C witnessing this failure. The memory is updated as follows: assume that the current valuation is $\mathbf{v}$ and that the strategy τ chose a literal corresponding to the variable x_p. If x_p is universal the valuation is unchanged. If x_p is existential, we proceed as above to construct another negative valuation where the value of x_p is swapped.

Let π be a play consistent with τ and x be the lowest quantified variable chosen infinitely many times by Adam. As before, we look at the sequence $(\mathbf{v}_i(x))_{i \in \mathbb{N}}$ where

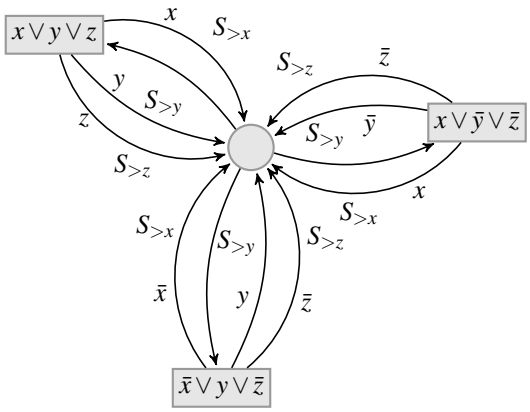


Figure 2.2: The Muller game for $\Psi = \exists x, \forall y, \exists z, (x \wedge y \wedge z) \bigvee (x \wedge \bar{y} \wedge \bar{z}) \bigvee (\bar{x} \wedge y \wedge \bar{z})$. For a variable v we write $S_{>v}$ for the set of literals corresponding to variables quantified after v, so for instance $S_{>x} = \{y, \bar{y}, z, \bar{z}\}$.

$\mathbf{v}_i$ is the valuation in the i^{th} round. If x is existential, the value changes each time the variable x is chosen. If x is universal, the sequence is ultimately constant. Since any literal that Adam chooses is necessarily false under the current valuation, this implies that in both cases π does not satisfy $\texttt{Muller}(\mathscr{F})$. □

2.6 Zielonka tree

The Zielonka tree is a combinatorial structure associated with a Muller objective which very neatly exposes its properties. As a warm-up we first present its predecessor the LAR construction, and then show the properties of Zielonka trees. As we will see, the key feature of the Zielonka tree of a Muller objective $\texttt{Muller}(\mathscr{F})$ is to characterise its exact memory requirements.

The latest appearance record

Muller objectives can be reduced to parity objectives, see Section 1.6 for an introduction to reductions between objectives.

Theorem 21 (Latest Appearance Record (LAR) construction)**.** *Let $C = [1,d]$ be a set of colours and $\texttt{Muller}(\mathscr{F})$ a Muller objective. There exists a deterministic parity automaton $\text{LAR}_{\mathscr{F}}$ over the alphabet C defining $\texttt{Muller}(\mathscr{F})$. It has $d!$ states and has priorities in $[1,2d]$.*

LAR stands for *Latest Appearance Record*. In the literature the number of states is often $d \cdot d!$ instead of $d!$, the multiplicative factor d is saved since we consider transition-based acceptance conditions for automata.

Proof. We define the automaton $\mathrm{LAR}_{\mathscr{F}}$. The set of states is the set of lists of all colours of C without repetitions. We represent a list by $(c_1,\dots,c_d)$. The initial state is irrelevant because $\mathtt{Muller}(\mathscr{F})$ is prefix independent. The transition function is defined as follows: $\delta(\ell,c)$ is ℓ' obtained from ℓ by pushing c to the first position (hence shifting to the right the elements to the left of c). This is best understood on an example:

$$\delta((4,1,2,3),2) = (2,4,1,3).$$

Let j be the position of c in ℓ, the priority of this transition is defined by:

$$\mathfrak{c}((\ell,c,\ell')) = \begin{cases} 2j & \text{if } \ell([1,j]) \in \mathscr{F}, \\ 2j-1 & \text{otherwise.} \end{cases}$$

We now show that the automaton $\mathrm{LAR}_{\mathscr{F}}$ defines $\mathtt{Muller}(\mathscr{F})$. Let $\rho = c_0c_1\dots$ be an infinite word over the alphabet C. Let us consider the run of $\mathrm{LAR}_{\mathscr{F}}$ over ρ:

$$(\ell_0,c_0,\ell_1)(\ell_1,c_1,\ell_2)\dots$$

Let us write j_i for the position of c_i in ℓ_i. We consider $\mathtt{Inf}(\rho)$ the set of colours appearing infinitely many times and write j for its cardinal. From some point onwards the lists ℓ_i are of the form

$$(\underbrace{c_1,\dots,c_j}_{\mathtt{Inf}(\rho)},\ \underbrace{c_{j+1},\dots,c_d}_{C\setminus\mathtt{Inf}(\rho)}).$$

From this point on j_i is smaller than or equal to j, and it reaches j infinitely many times. It follows that the largest priority appearing infinitely many times in the run is $2j$ if $\mathtt{Inf}(\rho) \in \mathscr{F}$ and $2j-1$ if $\mathtt{Inf}(\rho) \notin \mathscr{F}$. Thus ρ is accepted by $\mathrm{LAR}_{\mathscr{F}}$ if and only if $\mathtt{Inf}(\rho) \in \mathscr{F}$, as desired. □

The Zielonka tree

Theorem 21 implies a reduction from Muller games to parity games as explained in Section 1.6. This yields a small improvement from the complexity results we already obtained for Muller games in Theorem 17, but not for the memory requirements. One weakness of the LAR construction is that its size depends only on the number of colours, and not on the properties of $\mathscr{F}$. The Zielonka tree is an improved take on the LAR.

Definition 3 (Zielonka tree)**.** *Let $\mathtt{Muller}(\mathscr{F})$ be a Muller objective over the set of colours C. The Zielonka tree $T_{\mathscr{F}}$ of $\mathtt{Muller}(\mathscr{F})$ is a rooted tree with nodes labelled by subsets of colours, it is constructed inductively as follows:*

- *the root is labelled C,*
- *the children of a node labelled S are the maximal subsets $S_1,\dots,S_k$ of S such that $S_i \in \mathtt{Muller}(\mathscr{F}) \iff S \notin \mathtt{Muller}(\mathscr{F})$.*

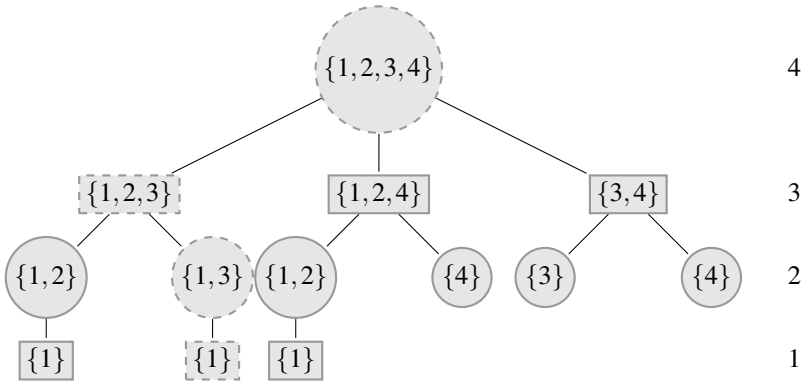


Figure 2.3: The Muller tree for $\texttt{Muller}(\mathscr{F})$. By convention nodes labelled by a set in $\mathscr{F}$ are represented by a circle and the others by a square. The numbers on the right hand side and the dashed nodes (describing a branch) are both used in the proof of Theorem 22.

Figure 2.3 represents the Zielonka tree for $\texttt{Muller}(\mathscr{F})$ with

$$\mathscr{F} = \{\{2\},\{3\},\{4\},\{1,2\},\{1,3\},\{1,3,4\},\{2,3,4\},\{1,2,3,4\}\}.$$

We note that there are two nodes labelled $\{1\}$; in general there may be several nodes with the same label. Also, not all branches have the same length.

The first use of the Zielonka tree is to induce an improved reduction from Muller to parity objectives. A branch in a tree is a path from the root to a leaf.

Theorem 22 (Reduction from Muller to parity games using the Zielonka tree automaton)**.** *Let $C = [1,d]$ be a set of colours and $\texttt{Muller}(\mathscr{F})$ a Muller objective. There exists a deterministic parity automaton $\text{Zielonka}_{\mathscr{F}}$ over the alphabet C defining $\texttt{Muller}(\mathscr{F})$. Its number of states is the number of branches of $T_{\mathscr{F}}$ and its parity condition uses d priorities.*

Here again we take advantage of the fact that the acceptance conditions on automata are transition based; using stated based transitions we would have added a multiplicative factor d.

Proof. Without loss of generality $C \in \mathscr{F}$: if this is not the case we consider the complement $\texttt{Muller}(2^C \setminus \mathscr{F})$. We number the levels of $T_{\mathscr{F}}$ from the leaves to the root such that nodes labelled by sets in $\mathscr{F}$ are even and the other ones odd (this will be used for defining the parity condition). See Figure 2.3 for a possible numeration of the levels (on the right hand side), the other options being shifts of this numeration by an even number.

The set of states of $\text{Zielonka}_{\mathscr{F}}$ is the set of branches of $T_{\mathscr{F}}$. We represent a branch by $(S_1,\dots,S_k)$ where S_1 is the set labelling the root and S_k the set labelling a leaf. Note that $k \leq d$. For the sake of simplicity we identify nodes with their labels, which is an abuse since two different nodes may have the same label but will be convenient and harmless in our reasoning.

The initial state is irrelevant because $\texttt{Muller}(\mathscr{F})$ is prefix independent. We define the support $\text{supp}(b,c)$ of a branch b and a colour c to be the lowest node of b which contains c. The transition function is defined as follows: $\delta(b,c)$ is the next branch (in the lexicographic order from left to right and in a cyclic way) which coincides with b up to $\text{supp}(b,c)$. The priority of this transition is given by the level on which $\text{supp}(b,c)$ sits.

This is best understood on an example: on Figure 2.3 consider the branch b represented by dashed nodes, reading the colour 2 we consider branches starting with $(\{1,2,3,4\},\{1,2,3\})$ because $\text{supp}(b,2) = \{1,2,3\}$.

The next branch after b is $(\{1,2,3,4\},\{1,2,3\},\{1,2\},\{1\})$ (because we cycle: the node after $\{1,3\}$ is $\{1,2\}$). The priority of this transition is 3 corresponding to the level where $\{1,2,3\}$ sits.

We now show that the automaton $\text{Zielonka}_{\mathscr{F}}$ defines $\texttt{Muller}(\mathscr{F})$. Let $\rho = c_0c_1\dots$ be an infinite word over the alphabet C. Let us consider the run of $\text{Zielonka}_{\mathscr{F}}$ over ρ:

$$(b_0,c_0,b_1)(b_1,c_1,b_2)\dots$$

We consider $\texttt{Inf}(\rho)$ the set of colours appearing infinitely many times. Let us look at the largest prefix $(S_1,\dots,S_p)$ of a branch which is eventually common to all the branches b_i. We make two claims:

- $\texttt{Inf}(\rho)$ is included in S_p;
- $\texttt{Inf}(\rho)$ is not included in any child of S_p.

For the first claim, let $c \in \texttt{Inf}(\rho)$, since eventually the branch b_i starts with $(S_1,\dots,S_p)$, the support of b_i and c is lower than or equal to S_p, meaning that $c \in S_p$.

For the second claim, we first note that by maximality of $(S_1,\dots,S_p)$ the support of b_i and c_i is infinitely many times S_p. Indeed from some point onwards it is lower than or equal to S_p, and if it would be eventually strictly lower then the corresponding child of S_p would be common to all branches b_i from there on. This implies that all children of S_p appear infinitely many times in the branches b_i: each time the support of b_i and c_i is S_p, the branch switches to the next child of S_p. Now since each child S_{p+1} of S_p is left infinitely many times this implies that there exists $c \in \texttt{Inf}(\rho)$ with $c \notin S_{p+1}$. Hence $\texttt{Inf}(\rho)$ is not included in S_{p+1}.

By definition of the Zielonka tree, this implies that $\texttt{Inf}(\rho) \in \mathscr{F}$ if and only if $S_p \in \mathscr{F}$, thus ρ is accepted by $\text{Zielonka}_{\mathscr{F}}$ if and only if $\texttt{Inf}(\rho) \in \mathscr{F}$, as desired. □

Since Theorem 22 is a reduction from Muller to parity objectives, it implies a reduction from Muller games to parity games as explained in Section 1.6, improving over Theorem 21. Since solving parity games is in $\mathsf{NP} \cap \mathsf{coNP}$, if we represent the Muller condition by a Zielonka tree then the automaton constructed in Theorem 22 is of polynomial size, implying the following result.

Theorem 23 (Complexity of solving Muller games represented by the Zielonka tree)**.** *Solving Muller games where the condition is represented by a Zielonka tree is in* $\mathsf{NP} \cap \mathsf{coNP}$.

As observed above different nodes of the Zielonka tree may be labelled by the same set of colours. Hence it is tempting to represent a Muller condition not with its Zielonka tree but rather with the Zielonka DAG (Directed Acyclic Graph) where nodes labelled by the same set of colours are identified. However with this representation solving Muller games is again PSPACE-complete:

Theorem 24 (Complexity of solving Muller games represented by the Zielonka DAG)**.** *Solving Muller games where the condition is represented by a Zielonka DAG is* PSPACE*-complete.*

The algorithm presented in Theorem 17 runs in polynomial space for this representation. To obtain the PSPACE-hardness we observe that in the reduction from QBF constructed in Theorem 20, the Muller objective is of polynomial size when represented by a Zielonka DAG (but of exponential size when represented by a Zielonka tree).

The exact memory requirements

The second and most interesting use of the Zielonka tree is for characterising the memory requirements.

Note that a node in the Zielonka tree $T_{\mathscr{F}}$ represents another Muller objective, over the set of colours labelling this node. For instance in Figure 2.3 the node labelled $\{1,2,3\}$ corresponds to $\texttt{Muller}(\mathscr{F}')$ with $\mathscr{F}' = \{\{2\},\{3\},\{1,2\},\{1,3\}\}$.

Definition 4 (Memory requirements for Muller objectives)**.** *Let* $\texttt{Muller}(\mathscr{F})$ *be a Muller objective over the set of colours C. We define* $m_{\mathscr{F}}$ *by induction:*

- *if the tree consists of a single leaf, then* $m_{\mathscr{F}} = 1$*;*
- *otherwise, let* $\mathscr{F}_1, \ldots, \mathscr{F}_k$ *be the families induced by the children of the root, there are two cases:*
  - *if* $C \in \mathscr{F}$*, then* $m_{\mathscr{F}}$ *is the sum of* $m_{\mathscr{F}_1}, \ldots, m_{\mathscr{F}_k}$*;*
  - *if* $C \notin \mathscr{F}$*, then* $m_{\mathscr{F}}$ *is the maximum of* $m_{\mathscr{F}_1}, \ldots, m_{\mathscr{F}_k}$*.*

For the Muller objective represented in Figure 2.3, we have $m_{\mathscr{F}} = 3$.

Theorem 25 (Memory requirements for Muller games)**.** *Muller objectives* $\texttt{Muller}(\mathscr{F})$ *are determined with finite memory strategies of size* $m_{\mathscr{F}}$*. This bound is tight: there exists a game with objective* $\texttt{Muller}(\mathscr{F})$ *where Eve wins using* $m_{\mathscr{F}}$ *memory states but not with less.*

We will not construct the lower bound, meaning the game where Eve needs $m_{\mathscr{F}}$ memory states to win. However, we will now prove the upper bound. To this end we revisit the recursive algorithm presented in Lemma 17 and Lemma 18. This algorithm was removing colours one by one and relying on the recursive solutions. We show that we can adapt the algorithm to follow instead the structure of the Zielonka tree: for solving a Muller game, it is enough to recursively solve the induced Muller games corresponding to the children of the root of the Zielonka tree. The following

lemma is an improved variant of Lemma 17. The corresponding pseudocode is given in Algorithm 2.6.

Lemma 19 (Fixed point characterisation of the winning regions for Muller games using the Zielonka tree)**.** *Let $\mathcal{G}$ be a Muller game with objective $\texttt{Muller}(\mathcal{F})$ such that $C \in \mathcal{F}$. Let $C_1, \dots, C_k$ be the maximal subsets of C such that $C_i \notin \mathcal{F}$. We let $\mathcal{F}_1, \dots, \mathcal{F}_k$ be the corresponding induced families, and define $\mathcal{G}_i$ be the subgame of $\mathcal{G}$ induced by $V \setminus Attr_{\mathrm{Eve}}(C_i)$ with objective $\texttt{Muller}(\mathcal{F}_i)$.*

- *If for all $i \in [1,k]$, we have $W_{\mathrm{Adam}}(\mathcal{G}_i) = \emptyset$, then $W_{\mathrm{Eve}}(\mathcal{G}) = V$.*
- *If there exists $i \in [1,k]$ such that $W_{\mathrm{Adam}}(\mathcal{G}_i) \neq \emptyset$, let $\mathcal{G}'$ be the subgame of $\mathcal{G}$ induced by $V \setminus Attr_{\mathrm{Adam}}(W_{\mathrm{Adam}}(\mathcal{G}_i))$, then $W_{\mathrm{Eve}}(\mathcal{G}) = W_{\mathrm{Eve}}(\mathcal{G}')$.*

We will prove the memory requirement at the same time inductively. Note that by duality, the case where $C \notin \mathcal{F}$ corresponds to the memory requirement for Adam when $C \in \mathcal{F}$:

$$m_{2^C \setminus \mathcal{F}} = \max_{i \in [1,k]} m_{2^{C_i} \setminus \mathcal{F}_i}.$$

Proof. We prove the first item.

For each $i \in [1,k]$, let σ_i be an attractor strategy ensuring to reach C_i from $\mathrm{Attr}_{\mathrm{Eve}}(C_i)$, and consider a winning strategy for Eve from $V \setminus \mathrm{Attr}_{\mathrm{Eve}}(C_i)$ in $\mathcal{G}_i$, it induces a strategy σ_i' in $\mathcal{G}$. We construct a strategy σ in $\mathcal{G}$ which will simulate the strategies above in turn; to do so it uses $[1,k]$ as top-level memory states. (We will look at more closely at the memory structure at the end of the proof.) The strategy σ with memory i simulates σ_i from $\mathrm{Attr}_{\mathrm{Eve}}(C_i)$ and σ_i' from $V \setminus \mathrm{Attr}_{\mathrm{Eve}}(C_i)$, and if it ever reaches a vertex in C_i it updates its memory state to $i+1$ and 1 if $i = k$. Any play consistent with σ either updates its memory state infinitely many times, or eventually remains in $V \setminus \mathrm{Attr}_{\mathrm{Eve}}(C_i)$ and is eventually consistent with σ_i'. In the first case it sees a colour from each C_i infinitely many times, so by definition of the C_i's and since $C \in \mathcal{F}$ the play satisfies $\texttt{Muller}(\mathcal{F})$, and in the other case since σ_i' is winning the play satisfies $\texttt{Muller}(\mathcal{F})$. Thus σ is winning from V.

Let us now discuss how many memory states are necessary to implement the strategy σ. By induction hypothesis, each of the strategies σ_i' uses $m_{\mathcal{F}_i}$ memory states. Using a disjoint union of the memory structures we implement σ using $\sum_{i \in [1,k]} m_{\mathcal{F}_i}$ memory states, corresponding to the definition of $m_{\mathcal{F}}$.

We now look at the second item.

Consider a winning strategy for Adam from $W_{\mathrm{Adam}}(\mathcal{G}_i)$ in $\mathcal{G}_i$, it induces a strategy τ_i in $\mathcal{G}$. Thanks to Lemma 10 τ_i is a winning strategy in $\mathcal{G}$. Let τ_a denote an attractor strategy from $\mathrm{Attr}_{\mathrm{Adam}}(W_{\mathrm{Adam}}(\mathcal{G}_i)) \setminus W_{\mathrm{Adam}}(\mathcal{G}_i)$. Consider now a winning strategy in the game $\mathcal{G}'$ from $W_{\mathrm{Adam}}(\mathcal{G}')$, it induces a strategy τ' in $\mathcal{G}$. The set $V \setminus \mathrm{Attr}_{\mathrm{Adam}}(W_{\mathrm{Adam}}(\mathcal{G}_i))$ may not be a trap for Eve, so we cannot conclude that τ' is a winning strategy in $\mathcal{G}$, and it indeed may not be. We construct a strategy τ in $\mathcal{G}$ as the (disjoint) union of the strategy τ_a on $\mathrm{Attr}_{\mathrm{Adam}}(W_{\mathrm{Adam}}(\mathcal{G}_i)) \setminus W_{\mathrm{Adam}}(\mathcal{G}_i)$, the strategy τ_i on $W_{\mathrm{Adam}}(\mathcal{G}_i)$ and the strategy τ' on $W_{\mathrm{Adam}}(\mathcal{G}')$. We argue that τ is winning from $\mathrm{Attr}_{\mathrm{Adam}}(W_{\mathrm{Adam}}(\mathcal{G}_i)) \cup W_{\mathrm{Adam}}(\mathcal{G}')$ in $\mathcal{G}$. Indeed, any play consistent

with this strategy in $\mathcal{G}$ either stays forever in $W_{\text{Adam}}(\mathcal{G}')$ hence is consistent with τ' or enters $\text{Attr}_{\text{Adam}}(W_{\text{Adam}}(\mathcal{G}_i))$, so it is eventually consistent with τ_i. In both cases this implies that the play is winning. Thus we have proved that $\text{Attr}_{\text{Adam}}(W_{\text{Adam}}(\mathcal{G}_c)) \cup W_{\text{Adam}}(\mathcal{G}') \subseteq W_{\text{Adam}}(\mathcal{G})$.

We now show that $W_{\text{Eve}}(\mathcal{G}') \subseteq W_{\text{Eve}}(\mathcal{G})$, which implies the converse inclusion. Consider a winning strategy from $W_{\text{Eve}}(\mathcal{G}')$ in $\mathcal{G}'$, it induces a strategy σ in $\mathcal{G}$. Thanks to Lemma 11 σ is winning from $W_{\text{Eve}}(\mathcal{G}')$ in $\mathcal{G}$.

Let us now discuss how many memory states are necessary to implement the strategy τ. By induction hypothesis, the strategy τ_i uses $m_{2^{C_i} \setminus \mathcal{F}_i}$ memory states and the strategy τ' uses $\max_{j \neq i} m_{2^{C_j} \setminus \mathcal{F}_j}$ memory states. Since τ is a disjoint union of strategies the memory can be reused so we can implement τ using $\max_{i \in [1,k]} m_{2^{C_i} \setminus \mathcal{F}_i}$ memory states, corresponding to the definition of $m_{2^C \setminus \mathcal{F}}$. □

The corresponding lemma when $C \notin \mathcal{F}$ is stated below, its proof is analogous to the previous one by swapping the two players.

Lemma 20 (Dual fixed point characterisation of the winning regions for Muller games using the Zielonka tree)**.** *Let $\mathcal{G}$ be a Muller game such that $C \notin \mathcal{F}$. Let $C_1, \dots, C_k$ be the maximal subsets of C such that $C_i \in \mathcal{F}$. We let $\mathcal{F}_1, \dots, \mathcal{F}_k$ be the corresponding induced Muller objectives, and define $\mathcal{G}_i$ be the subgame of $\mathcal{G}$ induced by $V \setminus Attr_{\text{Adam}}(C_i)$ with objective* $\texttt{Muller}(\mathcal{F}_i)$.

- *If for all $i \in [1,k]$, we have $W_{\text{Eve}}(\mathcal{G}_i) = \emptyset$, then $W_{\text{Adam}}(\mathcal{G}) = V$.*
- *If there exists $i \in [1,k]$ such that $W_{\text{Eve}}(\mathcal{G}_i) \neq \emptyset$, let $\mathcal{G}'$ be the subgame of $\mathcal{G}$ induced by $V \setminus Attr_{\text{Eve}}(W_{\text{Eve}}(\mathcal{G}_i))$, then $W_{\text{Adam}}(\mathcal{G}) = W_{\text{Adam}}(\mathcal{G}')$.*

Revisiting Streett, Rabin, and parity objectives

Let us look at the Streett, Rabin, and parity objectives under the new light shed by Theorem 25. It is instructive to look at the Zielonka tree of a Rabin objective, illustrated in Figure 2.4. It has a simple recursive structure: the Zielonka tree of the Rabin objective for d pairs contains d copies of the Zielonka tree of the Rabin objective for $d-1$ pairs. Naturally, this implies that $m_{\texttt{Rabin}} = 1$, so Theorem 25 implies the half-positional determinacy result stated in Theorem 18. Note that the two proofs are very different: the proof of Theorem 25 is by induction over the Zielonka tree and can be extended to infinite games, while the proof of Theorem 16 applies only to finite games but gives a general sufficient condition for half-positional determinacy.

Recall that we defined Streett objectives using closure under union, and Rabin objectives as the complement of Streett objectives.

Theorem 26 (Positionally determined Muller objectives)**.** *Let* $\texttt{Muller}(\mathcal{F})$ *be a Muller objective.*

- $\texttt{Muller}(\mathcal{F})$ *is half-positionally determined if and only if* $\texttt{Muller}(\mathcal{F})$ *is a Rabin objective;*

Algorithm 2.6: A recursive algorithm for computing the winning regions of Muller games following the Zielonka tree.

Data: A Muller game $\mathscr{G}$ over C
Function `SolveIn(`$\mathscr{G}$`)`**:**
  `// Assumes` $C \in \mathscr{F}$
  if $C = \{c\}$ **then**
    return V
  Let $C_1, \ldots, C_k$ the labels of the children of the root of the Zielonka tree of `Muller(`$\mathscr{F}$`)`
  for $i \in [1,k]$ **do**
    $\mathscr{G}_i \leftarrow \mathscr{G} \setminus \mathrm{Attr}^{\mathscr{G}}_{\mathrm{Eve}}(C_i)$
    $W_{\mathrm{Eve}}(\mathscr{G}_i) \leftarrow$ `SolveOut(`$\mathscr{G}_i$`) // The Zielonka tree of` $\mathscr{G}_i$ `is the` i`-th subtree`
  if $\forall i \in [1,k], W_{\mathrm{Adam}}(\mathscr{G}_i) = \emptyset$ **then**
    return V
  else
    Let i such that $W_{\mathrm{Adam}}(\mathscr{G}_i) \neq \emptyset$
    $\mathscr{G}' \leftarrow \mathscr{G} \setminus \mathrm{Attr}^{\mathscr{G}}_{\mathrm{Adam}}(W_{\mathrm{Adam}}(\mathscr{G}_i))$
    return `SolveIn(`$\mathscr{G}'$`) //` $\mathscr{G}'$ `has less vertices`

Function `SolveOut(`$\mathscr{G}$`)`**:**
  `// Symmetric to SolveIn, assumes` $C \notin \mathscr{F}$

if $C \in \mathscr{F}$ **then**
  `SolveIn(`$\mathscr{G}$`)`
else
  `SolveOut(`$\mathscr{G}$`)`

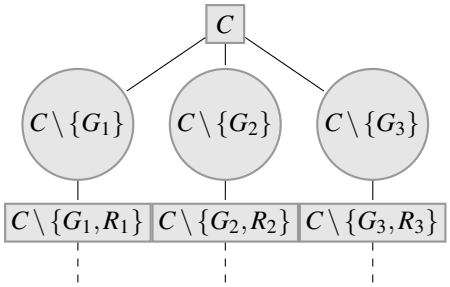


Figure 2.4: The (beginning of the) Zielonka tree for `Rabin` with three pairs: $C = \{G_1, R_1, G_2, R_2, G_3, R_3\}$.

- $\texttt{Muller}(\mathscr{F})$ *is positionally determined if and only if* $\texttt{Muller}(\mathscr{F})$ *is a parity objective.*

This theorem gives a characterisation of Rabin and parity objectives: they form the class of Muller objectives which are respectively half-positional and positional.

Proof. Thanks to Theorem 25 the objective $\texttt{Muller}(\mathscr{F})$ is half-positionally determined if and only if $m_{\mathscr{F}} = 1$, which is equivalent to saying that all nodes labelled $S \in \mathscr{F}$ in the Zielonka tree of $\mathscr{F}$ have at most one child. Indeed, for such nodes the number m is obtained as the sum of the numbers for the children, so there can be at most one, and conversely if this is the case then $m_{\mathscr{F}} = 1$. This characterisation of the Zielonka tree is equivalent to the complement of $\mathscr{F}$ being closed under union:

- Assume that the complement of $\mathscr{F}$ is closed under union and let $S \in \mathscr{F}$ be a node in the Zielonka tree of $\mathscr{F}$. Let $S_1, \dots, S_k$ be the children of S, by definition they are the maximal subsets of S such that $S_i \notin \mathscr{F}$. The union $\bigcup_i S_i$ is a subset of S and by closure under union of the complement of $\mathscr{F}$ it is in the complement of $\mathscr{F}$, implying by maximality that it is one of the children, so they are all equal and $k = 1$.
- Conversely, assume that all nodes labelled $S \in \mathscr{F}$ in the Zielonka tree of $\mathscr{F}$ have at most one child. Let $S_1, S_2 \notin \mathscr{F}$, towards contradiction assume that $S_1 \cup S_2 \in \mathscr{F}$. By definition of the Zielonka tree, if $S_1 \cup S_2$ is included into a node $S \notin \mathscr{F}$, then $S_1 \cup S_2$ is included into one of its children. Starting from the root and applying this we find a node $S \in \mathscr{F}$ such that $S_1 \cup S_2 \subseteq S$ and $S_1 \cup S_2 \not\subseteq S'$ with S' the only child of S (the case where S does not have any children is easy and treated separately). By definition of the Zielonka tree, since $S_1, S_2 \notin \mathscr{F}$ and $S_1, S_2 \subseteq S$, then $S_1, S_2 \subseteq S'$, implying that $S_1 \cup S_2 \subseteq S'$, a contradiction.

We have proved the first equivalence: $\texttt{Muller}(\mathscr{F})$ is half-positionally determined if and only if the complement of $\mathscr{F}$ is closed under union, which is the definition of Rabin objectives.

For the second equivalence, we already have that $\texttt{Muller}(\mathscr{F})$ is positionally determined if and only if all nodes in the Zielonka tree of $\mathscr{F}$ have at most one child. The Zielonka tree is in this case a chain:

$$S_1 \subseteq S_2 \subseteq S_3 \subseteq S_4 \subseteq \cdots \subseteq S_{2d-1} \subseteq S_{2d} \subseteq C,$$

with $S_{2i} \in \mathscr{F}$ and $S_{2i-1} \notin \mathscr{F}$. Then $X \in \mathscr{F}$ is equivalent to asking that the largest $i \in [1, d]$ such that if $X \cap S_i \neq \emptyset$ is even. Assigning priority i to S_i we get that $X \in \texttt{Muller}(\mathscr{F})$ if and only if the largest priority appearing infinitely many times in X is even: this is the definition of the parity objective over the set of priorities $[1, 2d]$. Conversely, we observe that the Zielonka tree of a parity objective is indeed a chain. □

Bibliographic references

The interest in reachability objectives goes beyond automata theory and logic. The attractor computation presented in Section 2.1 is inspired by the backward induction

principle due to Zermelo [Zer13], which was used to show that well founded games (*i.e.* where all plays are finite) are determined. The word 'attractor' (together with 'traps' and 'subgames') first appeared in Zielonka's work on Muller games [Zie98], but without the algorithmic point of view. A naive implementation of the attractor would have a quadratic time complexity. It is difficult to give credit for the linear time algorithm since the problem being very natural it has appeared in several contexts, for instance in database theory as an inference algorithm by Beeri and Bernstein [BB79] or in the framework of computing least fixed points over transition systems by Arnold and Crubillé [AC88].

The other objectives studied in this chapter are called ω-regular, let us discuss their relevance in automata theory and logic. An important application of automata theory is to make logic effective: by translating, sometimes called compiling, a logical formula into an equivalent automaton, we can solve problems such as satisfiability or model-checking by reducing them to analysing automata and in particular their underlying graph structures. In this context, the reachability objective is used for automata over finite words: the classical definition is that a run is accepting if the last state is accepting. Monadic second-order logic over finite words can be effectively translated into finite automata, marking the beginning of a close connection between logic and automata theory.

Considering logics over infinite structures led to the study of automata over infinite structures such as words and trees. The first objective to be studied in this context was Büchi objective, introduced by Büchi [Büc62]: a run is accepting if it visits infinitely many times an accepting state. Unfortunately the class of languages of infinite words recognised by deterministic Büchi automata is not closed under projection (corresponding in logic to existential quantification), said differently non-deterministic Büchi automata are strictly more expressive than deterministic ones hence not equivalent to monadic second-order logic over infinite words. Muller [Mul63] introduced the Muller objectives and attempted to prove the closure under projection for deterministic Muller automata. Alas, the proof had a flaw. The first correct proof of the result is due to McNaughton [McN66].

The correspondence between monadic second-order logic and Muller automata was extended from infinite words to infinite binary trees by Rabin [Rab69a], yielding the celebrated decidability of monadic second-order logic over infinite trees. Rabin introduced and worked with Rabin objectives; his proof is arguably very complicated and a lot of subsequent works focussed on finding the right notions and tools for better understanding his approach. Streett [Str81] suggested to use the complement of Rabin objectives, now called Streett objectives, for translating temporal logics to Streett automata. As discussed in Section 1.10, a key step was made by applying determinacy results for games to complementation results for automata. The parity objectives appeared in this context as a (and in fact, the) subclass of Muller objectives which is positionally determined. They have been defined (with some variants) independently by several authors: Wagner [Wag79], Mostowski [Mos84] who called them 'Rabin chain', Emerson and Jutla [EJ91] who first used the name parity, and McNaughton [McN93]. The idea can be traced back to the 'difference hierarchy' by Hausdorff [Hau14]. The proof of the positionality was obtained independently by Mostowski [Mos91], Emer-

son and Jutla [EJ91], and McNaughton [McN93] (the latter proof is for finite games). Later Walukiewicz [Wal02] gave another very elegant proof.

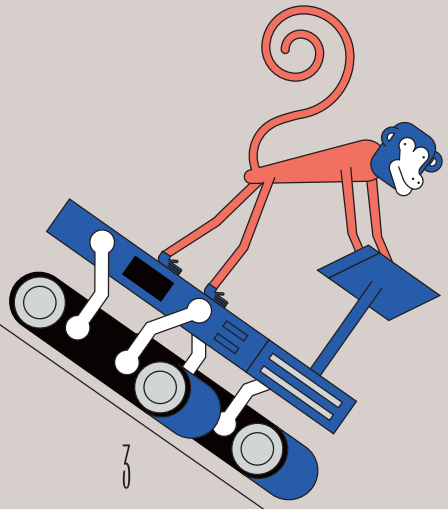
McNaughton [McN93] introduced the idea of solving Muller games by induction on the colours, leading to McNaughton algorithm as presented in Section 2.5. To some extent, the algorithms for solving Büchi, CoBüchi, and parity games are all special cases of McNaughton algorithm. Taking a step back in time, McNaughton already proposed the *Latest Appearance Record* (LAR) discussed in Section 2.6 for solving Muller games in his flawed attempt to solve the synthesis problem [McN65] (see Section 1.10). The LAR was later used by Gurevich and Harrington [GH82] as memory for winning strategies in Muller games. Thomas [Tho95] showed that the LAR can be used to reduce Muller games to parity games. Zielonka [Zie98] greatly contributed to the study of Muller objectives and their subclasses through his illuminating analysis of Zielonka trees. One of the many contributions of Zielonka's landmark paper [Zie98] was to follow McNaughton's approach for constructing a recursive algorithm for solving parity games, and show that it implies their positionality. We follow in Section 2.3 Zielonka's presentation of the algorithm, which is sometimes called Zielonka algorithm but more accurately McNaughton Zielonka algorithm. The characterisation result showing how Zielonka tree captures the exact memory requirements of Muller objectives is due to Dziembowski, Jurdziński, and Walukiewicz [DJW97].

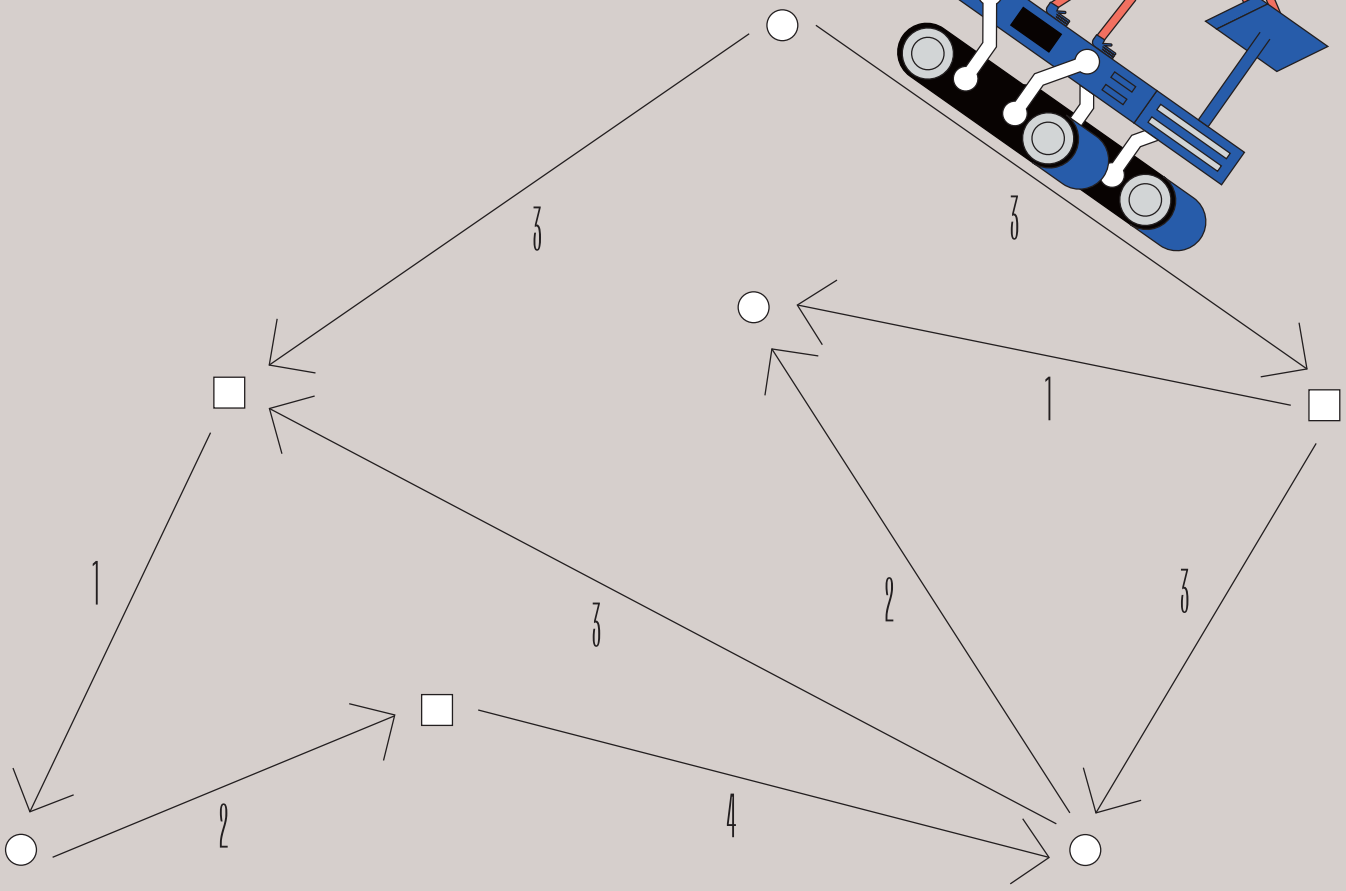
The NP-completeness stated in Theorem 19 for solving Rabin games is due to Emerson and Jutla [EJ88]. The study of the complexity of solving Muller games is due to Dawar and Hunter [HD05]. The PSPACE-completeness results stated in Theorems 20 and 24 only concern two representations for Muller objectives. There are several others, which are not equally succinct. For all representations but one the PSPACE-completeness result holds; the only exception is the explicit representation where the condition is specified by listing all sets of vertices in $\mathscr{F}$. Surprisingly, solving Muller games with the explicit representation is in P as shown by Horn [Hor08].

The complexity results stated for Muller games are not optimal, as they predate the quasi-polynomial time algorithms for parity games. The state of the art for Rabin and Muller games in terms of theoretical complexity is obtained by extending these algorithms.

Theorem 16 stating that submixing prefix independent objectives are half-positionally determined over finite arenas is inspired by the fairly mixing property of Gimbert and Zielonka [GZ04]. The word 'concave' is used in lieu of 'submixing' by Kopczyński [Kop06, Kop08]. Gimbert and Zielonka [GZ05] further refined the submixing property to give a characterisation of objectives which are positionally determined over finite games (they work in the more general framework of preference relations, which includes both qualitative and quantitative objectives). We refer to Section 4.1 for general approaches for proving half-positional determinacy for quantitative objectives.

PARITY GAMES





Chapter 3

Parity Games

JOHN FEARNLEY, NATHANAËL FIJALKOW

We will construct several algorithms for solving parity games.

- We start in Section 3.1 with an exponential time *strategy improvement algorithm*. The algorithm constructs a sequence of improving strategies until reaching an optimal strategy.
- We continue in Section 3.2 by constructing a quasi-polynomial time algorithm *attractor decomposition algorithms*, which decompose a game through a sequence of attractor computations. The first and archetypical example is McNaughton Zielonka algorithm defined in Section 2.3, which we improve here to obtain a quasi-polynomial runtime.
- We introduce in Section 3.3 the framework of *separating automata* and give a quasi-polynomial time algorithm as an instantiation of it. Separating automata formalise a family of algorithms for reducing parity games to safety games. The algorithm has mildly smaller runtime than the algorithm from the previous section.
- We proceed in Section 3.4 with the construction of a quasi-polynomial time and quasi-linear space *value iteration algorithms*, which finds an optimal strategy through the computation of a value function. This is based on the notion of universal trees.

As a conclusion Section 3.5 discusses the relationships between the different algorithms: in what sense are separating automata and value iteration algorithms equivalent through the notion of universal trees, and how does this family compare to the other two families of algorithms described above.

Remark 9 (Positional determinacy)**.** *We already proved in Theorem 15 that parity games are positionally determined, so in this chapter when considering a strategy we implicitly assume that it is positional.*

3.1 An exponential time strategy improvement algorithm

Theorem 27 (Strategy improvement)**.** *There exists a strategy improvement algorithm for solving parity games in exponential time.*

One can identify from the literature at least four strategy improvement algorithms for solving parity games, all running in exponential time. Two of them are better explained as consequences of their counterparts for energy games and discounted payoff games: we first reduce parity games to the corresponding games, and apply the strategy improvement algorithm in that setting. We refer the reader to Chapter 4 for them. The other two are specific to parity games, although they are closely related to the first two algorithms.

In a nutshell: the algorithm constructs a sequence of strategies, the next one being an improvement over the current one, until reaching an optimal strategy.

Adding the option of stopping the game. Let $\mathscr{G}$ a parity game with n vertices and priorities in $[1,d]$. Let us give Eve an extra move $-$ that indicates that the game should stop and that she can play from any vertex of hers. So a strategy for Eve is now a function $\sigma : V_{\text{Eve}} \to E \cup \{-\}$ where $\sigma(v) = -$ indicates that Eve has chosen to stop the game, and $\sigma(v) \neq -$ should be interpreted as normal. Adam is not allowed to stop the game, so strategies for Adam remain unchanged. We say that a play ending with $-$ is stopped.

Evaluating a strategy. The first question is: given a strategy σ, how to evaluate it? Let $Y = \mathbb{N}^d \cup \{\top\}$. We first explain how to evaluate plays, using a function $f : [1,d]^\omega \cup [1,d]^* \to Y$:

- Stopped plays: if $\rho \in [1,d]^*$, then $f(\rho)$ is the tuple $(x_1, x_2, \ldots, x_d)$ where x_p is the number of times that priority p appears in ρ.
- Infinite plays: if $\rho \in [1,d]^\omega$, then $f(\rho)$ is $\top$.

The evaluation for finite plays can be computed using an automaton as follows: we define $\delta : Y \times [1,d] \to Y$ by

$$\delta((x_1, x_2, \ldots, x_d), p) = (x_1, x_2, \ldots, x_p + 1, x_{p+1}, \ldots, x_d)$$

and $\delta(\top, p) = \top$. Extending to $\delta : Y \times [1,d]^* \to Y$, we obtain the following fact.

Fact 9. *For $\rho \in [1,d]^*$ we have $f(\rho) = \delta((0,\ldots,0), \rho)$.*

We now define a total order $\leq$ on Y making it a complete lattice. First, $\top$ is the greatest element. For $t, t' \in \mathbb{N}^d$, we say that $t < t'$ if and only if p is the largest index such that $t_p \neq t'_p$ and

- either p is even and $t_p < t'_p$,
- or p is odd and $t_p > t'_p$.

Then $\leq$ is the reflexive closure of $<$: we have $t \leq t'$ if $t < t'$ or $t = t'$. These conditions specify which priorities Eve likes to see along a play. Given a choice, Eve would prefer to see even priorities and to avoid odd priorities, but these preferences are hierarchical: assuming that d is even, Eve would like to see as many edges of priority d as possible. If two plays see d the same number of times, Eve prefers the play that visits the fewest vertices of priority $d-1$, and if two plays see d and $d-1$ the same number of times, then Eve would like to maximise the number of times that $d-2$ is visited, and so on recursively.

We can now define the value of a strategy σ:

$$\mathrm{val}^{\sigma}(v) = \inf_{\tau} f(\pi_v^{\sigma,\tau}),$$

where τ ranges over (general) strategies for Adam.

Since Y is totally ordered, we can naturally cast the computation of the value function as a shortest path problem: $\mathrm{val}^{\sigma}(v)$ is the value of the shortest path[1] (with respect to Y) in $\mathscr{G}[\sigma]$. Thus, any algorithm for the shortest path problem can be applied, such as the Bellman-Ford algorithm. In particular computing val^{σ} can be done in cubic time, and even more efficiently through a refined analysis which we do not perform here.

Recall that we say that a parity graph (without stopping option) *satisfies* parity from v if all infinite paths from v satisfy parity. Then a strategy σ is winning from v if and only if the parity graph $\mathscr{G}[\sigma]$ satisfies parity from v. We extend this definition to parity graphs with the stopping option: a strategy σ satisfies parity if all infinite plays consistent with σ satisfy parity. Crucially, this does not require anything of stopped plays. We say that a cycle is even if the maximal priority along the cycle is even, and it is odd otherwise. The characterisation of satisfying parity using cycles extends to this setting:

Fact 10 (Strategy satisfying parity in the presence of stopped plays)**.** *A strategy σ satisfies parity if and only if all cycles in $\mathscr{G}[\sigma]$ are even.*

The algorithm will only manipulate strategies satisfying parity.

Improving a strategy. We reach the last item in the construction of the algorithm: the notion of improving edges. Let σ a strategy. Let us consider a vertex $u \in V_{\mathrm{Eve}}$, we say that $e : u \xrightarrow{p} v$ is an *improving edge* if

$$\delta(\mathrm{val}^{\sigma}(v), p) > \mathrm{val}^{\sigma}(u).$$

Intuitively: according to val^{σ}, playing e is better than playing $\sigma(u)$. Given a strategy σ and a set of improving edges S, we write $\sigma[S]$ for the strategy

$$\sigma[S](u) = \begin{cases} e & \text{if there exists } e = u \xrightarrow{p} v \in S, \\ \sigma(u) & \text{otherwise.} \end{cases}$$

[1] We use the usual albeit sometimes confusing terminology from graph algorithms: "shortest path" means of smallest value with respect to Y, not in number of edges.

The algorithm. The algorithm starts with a specified initial strategy, which is the strategy σ_0 where $\sigma_0(v) = -$ for all vertices $v \in V_{\text{Eve}}$. It may not hold that σ_0 satisfies parity since $\mathscr{G}$ may contain odd cycles fully controlled by Adam. This can be checked in linear time and the attractor to the corresponding vertices removed from the game. After this preprocessing σ_0 indeed satisfies parity.

The pseudocode of the algorithm is given in Algorithm 3.1.

Algorithm 3.1: The strategy improvement algorithm for parity games.

```
Data: A parity game 𝒢
for v ∈ V_Eve do
    σ_0(v) ← −
C ← {v ∈ V : v contained in an odd cycle in 𝒢[σ_0]}
𝒢 ← 𝒢 \ Attr_Adam(C)
repeat
    for i = 0, 1, 2, ... do
        Compute val^{σ_i} and the set of improving edges
        if σ_i does not have improving edges then
            return {v ∈ V : val^{σ_i}(v) = ⊤}
        Choose a non-empty set S_i of improving edges
        σ_{i+1} ← σ_i[S_i]
```

Proof of correctness. As per usual for strategy improvement algorithms, the correctness depend on the following two principles:

- **Progress**: updating a strategy using improving edges is a strict improvement,
- **Optimality**: a strategy which does not have any improving edges is optimal.

The difficulty is that an edge being improving does not mean that it is a better move than the current one in any context, but only according to the value function val^σ, so it is not clear that $\sigma[S]$ is better than σ. Let us write $\sigma \leq \sigma'$ if for all vertices v we have $\text{val}^\sigma(v) \leq \text{val}^{\sigma'}(v)$, and $\sigma < \sigma'$ if additionally $\neg(\sigma' \leq \sigma)$.

We start by stating a very simple property of δ.

Fact 11 (Key property of δ)**.** *Let $t \in Y$ and $p_1, \ldots, p_k \in [1,d]$ with $\delta(t, p_1 \ldots p_k) \neq \top$. The following holds:*

- $t \neq \delta(t, p_1 \ldots p_k)$.
- $t < \delta(t, p_1 \ldots p_k)$ *if and only if* $\max\{p_1, \ldots, p_k\}$ *is even.*
- $t > \delta(t, p_1 \ldots p_k)$ *if and only if* $\max\{p_1, \ldots, p_k\}$ *is odd.*

The following lemma states the two important properties of $(Y, \leq)$ and δ.

Lemma 21. *Let G a parity graph (with no stopping option).*

- *If there exists $\mu : V \to Y$ such that for all vertices u we have $\mu(u) \neq \top$ and for all edges $u \xrightarrow{p} v \in E$ we have $\mu(u) \leq \delta(\mu(v), p)$, then G satisfies parity.*
- *If there exists $\mu : V \to Y$ such that for all vertices v we have $\mu(u) \neq \top$ and for all edges $u \xrightarrow{p} v \in E$ we have $\mu(u) \geq \delta(\mu(v), p)$, then G satisfies the complement of parity.*

Proof. We prove the first property, the second is proved in exactly the same way. Thanks to the characterisation using cycles it is enough to show that all cycles in G are even. Let us consider a cycle in G:

$$\pi = v_0 \xrightarrow{p_0} v_1 \xrightarrow{p_1} v_2 \cdots v_{k-1} \xrightarrow{p_{k-1}} v_0.$$

For all $i \in [0, k-1]$ we have $\mu(v_i) \leq \delta(\mu(v_{i+1 \mod k}), p_i)$. By monotonicity of δ this implies $\mu(v_0) \leq \delta(\mu(v_0), p_{k-1} \cdots p_0)$. Thanks to Fact 11 this implies that the maximum priority in $\{p_0, \ldots, p_{k-1}\}$ is even. □

Let us fix a strategy σ. We let F_V^σ denote the set of functions $\mu : V \to Y$ such that $\mu(v) = (0, \ldots, 0)$ if $\sigma(v) = -$, it is a lattice when equipped with the componentwise (partial) order induced by Y: we say that $\mu \leq \mu'$ if for all vertices v we have $\mu(v) \leq \mu'(v)$. We then define an operator $\mathbb{O}^\sigma : F_V^\sigma \to F_V^\sigma$ by

$$\mathbb{O}^\sigma(\mu)(u) = \begin{cases} \min\left\{\delta(\mu(v), p) : u \xrightarrow{p} v \in E\right\} & \text{if } u \in V_{\text{Adam}}, \\ \delta(\mu(v), p) & \text{if } u \in V_{\text{Eve}} \text{ and } \sigma(u) = u \xrightarrow{p} v, \\ (0, \ldots, 0) & \text{if } u \in V_{\text{Eve}} \text{ and } \sigma(u) = -. \end{cases}$$

Lemma 22. *For all σ satisfying parity, val^σ is a fixed point of $\mathbb{O}^\sigma$. Further, $val^\sigma(u) = \top$ if and only if all paths consistent with σ from u are infinite, and if $val^\sigma(u) < \top$, then there exists π stopped play such that $val^\sigma(u) = f(\pi)$.*

We now rely on Lemmata 21 and 22 to prove the two principles: progress and optimality.

Lemma 23 (Progress)**.** *Let σ a strategy satisfying parity and S a set of improving edges. We let σ' denote $\sigma[S]$. Then σ' satisfies parity and $\sigma < \sigma'$.*

Proof. We first argue that σ' satisfies parity. To this end let us consider the parity graph $\mathscr{G}[\sigma']$. We claim that for all edges $e = u \xrightarrow{p} v$ in $\mathscr{G}[\sigma']$, we have $\text{val}^\sigma(u) \leq \delta(\text{val}^\sigma(v), p)$:

- Either e is an edge in $\mathscr{G}[\sigma]$ and this is because val^σ is a fixed point of $\mathbb{O}^\sigma$,
- Or e was an improving edge, in which case $\text{val}^\sigma(u) < \delta(\text{val}^\sigma(v), p)$.

Since σ satisfies parity, for all vertices u such that $\text{val}^\sigma(u) = \top$ all paths starting from u satisfy parity. Let us write G for the graph obtained from $\mathscr{G}[\sigma']$ by removing all such vertices. The first item of Lemma 21 implies that G satisfies parity, hence $\mathscr{G}[\sigma']$ satisfies parity.

At this point we know that σ' satisfies parity, let us show that $\mathrm{val}^{\sigma} \leq \mathrm{val}^{\sigma'}$. Let $u_0 \in V$, let π' consistent with σ' from u_0 such that $\mathrm{val}^{\sigma'}(u_0) = f(\pi')$, we show that $\mathrm{val}^{\sigma}(u_0) \leq f(\pi')$.

If $f(\pi') = \top$ this is clear, so let us assume that $f(\pi') < \top$, meaning π' is finite. We reason by induction on the number of improving edges in π'. If there are none, then π' is consistent with σ and we are done. Let us consider the induction step: we write $\pi' = \pi'_1 \cdot u_1 \xrightarrow{p_1} v_1 \cdot \pi'_2$ with $u_1 \xrightarrow{p_1} v_1$ the first improving edge in π'. By definition of an improving edge we have $\mathrm{val}^{\sigma}(u_1) < \delta(\mathrm{val}^{\sigma}(v_1), p_1)$. Note that $\mathrm{val}^{\sigma'}(v_1) = f(\pi'_2)$, by induction hypothesis this implies that $\mathrm{val}^{\sigma}(v_1) \leq f(\pi'_2)$. Since $\mathrm{val}^{\sigma'}(u_1) = \delta(\mathrm{val}^{\sigma'}(v_1), p_1)$, we obtain $\mathrm{val}^{\sigma}(u_1) \leq \mathrm{val}^{\sigma'}(u_1)$. Since π'_1 does not contain any improving edges, it is consistent with σ, hence an easy induction implies that $\mathrm{val}^{\sigma}(u_0) \leq \mathrm{val}^{\sigma'}(u_0)$.

We now show that $\mathrm{val}^{\sigma} < \mathrm{val}^{\sigma'}$. Let us consider $e = u \xrightarrow{p} v$ an improving edge. Using $\mathrm{val}^{\sigma}(v) \leq \mathrm{val}^{\sigma'}(v)$ and the monotonicity of δ we obtain that $\delta(\mathrm{val}^{\sigma}(v), p) \leq \delta(\mathrm{val}^{\sigma'}(v), p)$. Since $\mathrm{val}^{\sigma'}$ is a fixed point of $\mathbb{O}^{\sigma'}$ we have $\mathrm{val}^{\sigma'}(u) = \delta(\mathrm{val}^{\sigma'}(v), p)$ and since e is improving we have $\mathrm{val}^{\sigma}(u) < \delta(\mathrm{val}^{\sigma}(v), p)$. This implies that $\mathrm{val}^{\sigma}(u) < \mathrm{val}^{\sigma'}(u)$. □

Lemma 24 (Optimality)**.** *Let σ a strategy satisfying parity that has no improving edges, then σ is winning from all vertices of $W_{\mathrm{Eve}}(\mathcal{G})$.*

Proof. The fact that σ satisfies parity means that it is a winning strategy from all vertices v such that $\mathrm{val}^{\sigma}(v) = \top$. We now prove that Adam has a winning strategy from all vertices v such that $\mathrm{val}^{\sigma}(v) \neq \top$. We construct a strategy of Adam by

$$\forall u \in V_{\mathrm{Adam}},\ \tau(u) \in \operatorname{argmin} \left\{ \delta(\mathrm{val}^{\sigma}(v), p) : u \xrightarrow{p} v \in E \right\}.$$

We argue that τ ensures the complement of parity from all vertices v such that $\mathrm{val}^{\sigma}(v) \neq \top$. Let us consider G the subgraph of $\mathcal{G}[\tau]$ restricted to such vertices. We argue that for all edges $u \xrightarrow{p} v$ in G, we have $\mathrm{val}^{\sigma}(u) \geq \delta(\mathrm{val}^{\sigma}(v), p)$. Once this is proved we conclude using the second item of Lemma 21 implying that G satisfies the complement of parity.

The first case is when $u \in V_{\mathrm{Eve}}$. Let $\sigma(u) = u \xrightarrow{p'} v'$. Consider an edge $e = u \xrightarrow{p} v$, since it is not improving we have $\delta(\mathrm{val}^{\sigma}(v), p) \leq \delta(\mathrm{val}^{\sigma}(v'), p')$. Since val^{σ} is a fixed point of $\mathbb{O}^{\sigma}$ we have $\mathrm{val}^{\sigma}(u) = \delta(\mathrm{val}^{\sigma}(v'), p')$, implying the desired inequality.

The second case is when $u \in V_{\mathrm{Adam}}$, it holds by definition of τ and because val^{σ} is a fixed point of $\mathbb{O}^{\sigma}$. □

Complexity analysis. One aspect of the algorithm we do not elaborate on here is the choice of improving edges at each iteration. Many possible rules for choosing this set have been studied, as for instance the *greedy all-switches* rule, choosing maximal improving edges. This impacts the number of iterations of the algorithm, meaning the length of the sequence $\sigma_0, \sigma_1, \ldots$. It is at most exponential since it is bounded by the number of strategies (which is bounded aggressively by m^n). There are lower bounds

showing that the sequence can be of exponential length, which apply to different rules for choosing sets of improving edges. Hence the overall complexity is exponential

The other strategy improvement algorithm for parity games. The main drawback of the algorithm we constructed is that it requires the initial strategy to be the one stopping at every vertex. This is in contrast with many practical scenarios, where the fact that the algorithm can be initialised with any strategy (hence a potentially good one) is a key factor for performances. As discussed at the beginning of this section, there are several other strategy improvement algorithms. All but one are obtained by reducing parity games to another class of games and applying a strategy improvement algorithm there. The remaining one is defined directly on parity games. The valuation of a play is a triple:

- the maximum priority p seen infinitely many times,
- the set of higher priorities seen before the first occurrence of p,
- the length of the prefix before the first occurrence of p.

The key benefit of this algorithm is that it can be initialised with any strategy. We do not present here a correctness proof. Let us note that the algorithm is actually closely related to the one we built here, in the sense that after adding a stopping option the two algorithms coincide. Both algorithms have exponential complexity, with exponential lower bounds on the number of iterations.

3.2 A quasi-polynomial time attractor decomposition algorithm

Theorem 28 (Quasi-polynomial McNaughton Zielonka algorithm)**.** *There exists a quasi-polynomial time algorithm for solving parity games, and more specifically of complexity* $n^{O(\log n)}$.

We revisit the exponential recursive algorithm presented in Section 2.3. We refer to Algorithm 3.2 for an equivalent presentation of this algorithm, where we make explicit all recursive calls involving the maximal priority d. The benefit of doing this is to make the following observation: during the i^{th} recursive call for d, the algorithm removes from the game $\mathscr{G}$ the subset $X_i = \text{Attr}^{\mathscr{G}}_{\text{Adam}}(W_{\text{Adam}}(\mathscr{G}_i))$. Note that X_i is a trap for Eve in $\mathscr{G}$ and a subset of the winning region of Adam in $\mathscr{G}$: we say that X_i is a dominion for Eve. More generally, given a game $\mathscr{G}$, a set X of vertices is a *dominion* for Eve if it is a trap for Adam and a subset of the winning region of Eve.

Let i_∞ be the final value of i, the sets $X_0, \ldots, X_{i_\infty - 1}$ are disjoint. The *key idea* is that this implies that at most one of them can have size more than half the total size.

To take advantage of this, we change the specification of the algorithm: the new algorithm takes as input a parity game and two parameters s_{Eve} and s_{Adam}. As before, they are two mutually recursive procedures, *SolveE* and *SolveA*. At an intuitive level,

Algorithm 3.2: The recursive algorithm for computing the winning region of parity games.

Data: A parity game $\mathscr{G}$ with priorities in $[1,d]$
Function `SolveEven(`$\mathscr{G}$`)`**:**
  $i \leftarrow -1$
  $V_0 \leftarrow V$
  repeat
    $i \leftarrow i+1$
    Let $\mathscr{G}_i$ the subgame of $\mathscr{G}$ induced by $V_i \setminus \mathrm{Attr}^{\mathscr{G}_i}_{\mathrm{Eve}}(d)$
    $W_{\mathrm{Eve}}(\mathscr{G}_i) \leftarrow$ `SolveOdd(`$\mathscr{G}_i$`)`
    $V_{i+1} \leftarrow V_i \setminus \mathrm{Attr}^{\mathscr{G}}_{\mathrm{Adam}}(W_{\mathrm{Adam}}(\mathscr{G}_i))$
  until $W_{\mathrm{Adam}}(\mathscr{G}_i) = \emptyset$
  return V_i

Function `SolveOdd(`$\mathscr{G}$`)`**:**
  if $d = 1$ **then**
    return V
  $i \leftarrow -1$
  $V_0 \leftarrow V$
  repeat
    $i \leftarrow i+1$
    Let $\mathscr{G}_i$ the subgame of $\mathscr{G}$ induced by $V_i \setminus \mathrm{Attr}^{\mathscr{G}_i}_{\mathrm{Adam}}(d)$
    $W_{\mathrm{Adam}}(\mathscr{G}_i) \leftarrow$ `SolveEven(`$\mathscr{G}_i$`)`
    $V_{i+1} \leftarrow V_i \setminus \mathrm{Attr}^{\mathscr{G}}_{\mathrm{Eve}}(W_{\mathrm{Eve}}(\mathscr{G}_i))$
  until $W_{\mathrm{Eve}}(\mathscr{G}_i) = \emptyset$
  return V_i

if *d is even* **then**
  return `SolveEven(`$\mathscr{G}$`)`
else
  return `SolveOdd(`$\mathscr{G}$`)`

the objective of $\mathit{SolveE}(\mathscr{G}, s_{\text{Eve}}, s_{\text{Adam}})$ is to return a (non-empty whenever possible) dominion for Eve of size at most s_{Eve}.

We spell out the pseudocode of *SolveE* in Algorithm 3.3, leaving out the perfectly symmetric *SolveA*. The base cases are when there is only one priority, in which case Eve wins everywhere if the priority is even, and Adam wins everywhere if the priority is odd.

Algorithm 3.3: A recursive quasi-polynomial algorithm for computing the winning regions of parity games – the procedure *SolveE*.

Data: A parity game $\mathscr{G}$ with priorities in $[1,d]$, and d even, two parameters s_{Eve} and s_{Adam}

Let $i = 0$ and $V_0 = V$

Let $\mathscr{H}_0$ the subgame of $\mathscr{G}$ induced by V_0

Let $\mathscr{G}_0$ the subgame of $\mathscr{H}_0$ induced by $V_0 \setminus \text{Attr}^{\mathscr{H}_0}_{\text{Eve}}(d)$

Function `Treat` (X_i) **:**

  Let $V_{i+1} = V_i \setminus \text{Attr}^{\mathscr{H}_i}_{\text{Adam}}(X_i)$

  Let $\mathscr{H}_{i+1}$ the subgame of $\mathscr{H}_i$ induced by V_{i+1}

  Let $\mathscr{G}_{i+1}$ the subgame of $\mathscr{H}_{i+1}$ induced by $V_{i+1} \setminus \text{Attr}^{\mathscr{H}_{i+1}}_{\text{Eve}}(d)$

  $i = i+1$

`// recursive calls for small dominions`

while $X_i = \mathit{SolveA}(\mathscr{G}_i, s_{\text{Eve}}, \lfloor s_{\text{Adam}}/2 \rfloor) \neq \emptyset$ **do**

  `Treat` (X_i)

`// one recursive call for a large dominion`

if $X_i = \mathit{SolveA}(\mathscr{G}_i, s_{\text{Eve}}, s_{\text{Adam}}) \neq \emptyset$ **then**

  `Treat` (X_i)

`// recursive calls for small dominions`

while $X_i = \mathit{SolveA}(\mathscr{G}_i, s_{\text{Eve}}, \lfloor s_{\text{Adam}}/2 \rfloor) \neq \emptyset$ **do**

  `Treat` (X_i)

return V_i

We need three simple facts about traps.

Fact 12 (Facts about traps)**.**

- *Let S be a trap for Eve in the game $\mathscr{G}$ and X a set of vertices, then $S \setminus \mathit{Attr}_{\text{Eve}}(X)$ is a trap for Eve in the subgame of $\mathscr{G}$ induced by $V \setminus \mathit{Attr}_{\text{Eve}}(X)$.*
- *Let S be a trap for Eve in the game $\mathscr{G}$ and X a set of vertices such that $S \cap X = \emptyset$, then $S \subseteq V \setminus \mathit{Attr}_{\text{Adam}}(X)$ and S is a trap for Eve in the subgame of $\mathscr{G}$ induced by $V \setminus \mathit{Attr}_{\text{Adam}}(X)$.*
- *Let S be a trap for Eve in the game $\mathscr{G}$ and Z a trap for Eve in the subgame of $\mathscr{G}$ induced by S, then Z is a trap for Eve in $\mathscr{G}$.*

The following lemma implies the correctness of the algorithm.

Lemma 25 (Correctness of the quasi-polynomial McNaughton Zielonka algorithm)**.**

- *For all dominions S for Eve, if $|S| \leq s_{\text{Eve}}$, then $S \subseteq \mathit{SolveE}(\mathscr{G}, s_{\text{Eve}}, s_{\text{Adam}})$.*
- *For all dominions S for Adam, if $|S| \leq s_{\text{Adam}}$, then $S \cap \mathit{SolveE}(\mathscr{G}, s_{\text{Eve}}, s_{\text{Adam}}) = \emptyset$.*

Indeed, $W_{\text{Eve}}(\mathscr{G})$ and $W_{\text{Adam}}(\mathscr{G})$ are dominions for Eve and Adam in $\mathscr{G}$, so Lemma 25 implies that $\mathit{SolveE}(\mathscr{G}, n, n) = W_{\text{Eve}}(\mathscr{G})$.

Proof. The proof is by induction on the number of priorities: indeed all recursive calls to *SolveA* are for games with one less priority. It follows that by induction hypothesis the following two properties hold, for all i.

- For all dominions S in $\mathscr{G}_i$ for Adam, if $|S| \leq s_{\text{Adam}}$, then
$$S \subseteq \mathit{SolveA}(\mathscr{G}_i, s_{\text{Eve}}, s_{\text{Adam}}).$$
- For all dominions S in $\mathscr{G}_i$ for Eve, if $|S| \leq s_{\text{Eve}}$, then
$$S \cap \mathit{SolveA}(\mathscr{G}_i, s_{\text{Eve}}, s_{\text{Adam}}) = \emptyset.$$

Since there will be an induction inside this induction, we refer to the induction above as the external induction, and the second one as the internal induction.

We write i_∞ for the final value of i, *i.e.* such that $\mathit{SolveE}(\mathscr{G}, s_{\text{Eve}}, s_{\text{Adam}}) = V_{i_\infty}$. Note that the first item reads $S \subseteq V_{i_\infty}$.

Let S be a dominion for Eve in $\mathscr{G}$ such that $|S| \leq s_{\text{Eve}}$. We show by internal induction on i that $S \subseteq V_i$.

For $i = 0$ this is by definition. We now assume that $S \subseteq V_i$. Recall that $\mathscr{G}_i$ is the subgame of $\mathscr{H}_i$ induced by $V_i \setminus \text{Attr}^{\mathscr{H}_i}_{\text{Eve}}(d)$. It follows from the first item of Fact 12 that $S \setminus \text{Attr}^{\mathscr{H}_i}_{\text{Eve}}(d)$ is a dominion for Eve in $\mathscr{G}_i$. Since $S \setminus \text{Attr}^{\mathscr{H}_i}_{\text{Eve}}(d)$ has size at most s_{Eve}, the second item of the external induction hypothesis implies that $S \setminus \text{Attr}^{\mathscr{G}_i}_{\text{Eve}}(d)$ has an empty intersection with $X_i = \mathit{SolveA}(\mathscr{G}_i, s_{\text{Eve}}, s_{\text{Adam}})$, implying that $S \cap X_i = \emptyset$. It follows from the second item of Fact 12 that $S \subseteq V_i \setminus \text{Attr}^{\mathscr{H}_i}_{\text{Adam}}(X_i) = V_{i+1}$. This finishes the internal induction, and implies the first item.

We now prove the second item.

Let S be a (non-empty) dominion for Adam in $\mathscr{G}$ such that $|S| \leq s_{\text{Adam}}$. We write $S_i = S \cap V_i$. Note that the second item reads $S_{i_\infty} = \emptyset$.

We first show by internal induction on i that S_i is a dominion for Adam in $\mathscr{H}_i$. For $i = 0$ this is by definition. We now assume that S_i is a dominion for Adam in $\mathscr{H}_i$. Recall that $\mathscr{H}_{i+1}$ is the subgame of $\mathscr{H}_i$ induced by $V_{i+1} = V_i \setminus \text{Attr}^{\mathscr{H}_i}_{\text{Adam}}(X_i)$. It follows from the first item of Fact 12 applied to S_i (swapping the roles of Eve and Adam) that $S_i \setminus \text{Attr}^{\mathscr{H}_i}_{\text{Adam}}(X_i) = S_{i+1}$ is a dominion for Adam in $\mathscr{H}_{i+1}$. This finishes the internal induction.

We showed that S_i is a dominion for Adam in $\mathscr{H}_i$ for each i. To apply the external inductive hypothesis, we need to exhibit dominions for Adam in $\mathscr{G}_i$ for each i. We consider $\mathscr{H}_i'$ the subgame of $\mathscr{H}_i$ induced by S_i. Let $Z_i = W^{\mathscr{H}_i'}_{\text{Adam}}(\texttt{Parity} \cup \texttt{Reach}(d))$: in words, Z_i is the subset of vertices of S_i from where Adam can ensure that the parity objective is not satisfied and never to see priority d. We prove two properties:

- If $Z_i = \emptyset$, then $S_i = \emptyset$.

  The fact that Z_i is empty implies that Eve has a strategy σ in $\mathscr{G}_i$ which from S_i ensures `Parity` or to see priority d. Since S_i is a dominion for Adam in $\mathscr{H}_i$, there exists a strategy τ for Adam which from S_i ensures the complement of `Parity` and to stay in S_i. Playing σ against τ yields a contradiction, since σ ensures `Parity` if the play remains in S_i.

- Z_i is a dominion for Adam in $\mathscr{G}_i$.

  That Z_i is a subset of the winning region of Adam in $\mathscr{G}_i$ is clear. To see that Z_i is a trap for Eve in $\mathscr{G}_i$, we first note that since S_i is a trap for Eve in $\mathscr{H}_i$ and Z_i is a trap for Eve in $\mathscr{H}_i'$, the subgame of $\mathscr{H}_i$ induced by S_i, then Z_i is a trap in $\mathscr{H}_i$ by the third item of Fact 12. Now, since Z_i has an empty intersection with d, by the second item of Fact 12 this implies that Z_i is a trap for Eve in the subgame of $\mathscr{H}_i$ induced by $V_i \setminus \mathrm{Attr}^{\mathscr{H}_i}_{\mathrm{Eve}}(d)$, which is exactly $\mathscr{G}_i$.

We are now fully equipped to prove that $S_{i_\infty} = \emptyset$. Let i_ℓ be the value of i for which the algorithm performs a recursive call for a large dominion. Since the sequence $(S_i)_i$ is non-increasing, if S_i is empty for some i, then S_{i_∞} as well.

The first while loop was exited for $i = i_\ell$, implying that

$$\mathit{SolveA}(\mathscr{G}_{i_\ell}, s_{\mathrm{Eve}}, \lfloor s_{\mathrm{Adam}}/2 \rfloor) = \emptyset.$$

We apply the external inductive hypothesis to the dominion Z_{i_ℓ} for Adam in $\mathscr{G}_{i_\ell}$ for the parameter $\lfloor s_{\mathrm{Adam}}/2 \rfloor$: if $|Z_{i_\ell}| \leq \lfloor s_{\mathrm{Adam}}/2 \rfloor$, then $Z_{i_\ell} \subseteq \mathit{SolveA}(\mathscr{G}_{i_\ell}, s_{\mathrm{Eve}}, \lfloor s_{\mathrm{Adam}}/2 \rfloor)$, implying that Z_{i_ℓ} is empty, which by the first property above implies that S_{i_ℓ} is empty, thus S_{i_∞} as well, proving the second item of Lemma 25. Excluding this case, we then analyse the situation where $|Z_{i_\ell}| > \lfloor s_{\mathrm{Adam}}/2 \rfloor$.

We apply again the external inductive hypothesis to Z_{i_ℓ}, but this time for the parameter s_{Adam}: if $|Z_{i_\ell}| \leq s_{\mathrm{Adam}}$, then $Z_{i_\ell} \subseteq X_{i_\ell}$. Since $Z_{i_\ell} \subseteq S_{i_\ell} \subseteq S$ and $|S| \leq s_{\mathrm{Adam}}$, the premise is satisfied, so $Z_{i_\ell} \subseteq X_{i_\ell}$. Since Z_{i_ℓ} is non-empty, so is X_{i_ℓ}: the search for a large dominion was successful, and in particular i is incremented at this stage, implying that $i_\ell < i_\infty$.

Consider $S_{i_\ell+1} = S_{i_\ell} \setminus \mathrm{Attr}^{\mathscr{H}_{i_\ell}}_{\mathrm{Adam}}(X_{i_\ell})$. In particular $S_{i_\ell+1} \subseteq S_{i_\ell} \setminus X_{i_\ell} \subseteq S_{i_\ell} \setminus Z_{i_\ell}$, so

$$|S_{i_\ell+1}| \leq |S_{i_\ell}| - |Z_{i_\ell}| \leq \lfloor s_{\mathrm{Adam}}/2 \rfloor.$$

The second while loop was exited for $i = i_\infty$, implying that

$$\mathit{SolveA}(\mathscr{G}_{i_\infty}, s_{\mathrm{Eve}}, \lfloor s_{\mathrm{Adam}}/2 \rfloor) = \emptyset.$$

We apply the external inductive hypothesis to the dominion Z_{i_∞} for Adam in $\mathscr{G}_{i_\infty}$ for the parameter $\lfloor s_{\mathrm{Adam}}/2 \rfloor$: if $|Z_{i_\infty}| \leq \lfloor s_{\mathrm{Adam}}/2 \rfloor$, then $Z_{i_\infty} \subseteq \mathit{SolveA}(\mathscr{G}_{i_\infty}, s_{\mathrm{Eve}}, \lfloor s_{\mathrm{Adam}}/2 \rfloor)$. The premise holds because $|S_{i_\ell+1}| \leq \lfloor s_{\mathrm{Adam}}/2 \rfloor$, the sequence $(S_i)_i$ is non-increasing, $i_\ell < i_\infty$, and $Z_{i_\infty} \subseteq S_{i_\infty}$. Hence Z_{i_∞} is empty. Thanks to the first property above for Z_i, this implies that S_{i_∞} is empty. This finishes the proof of the second item of Lemma 25. □

We obtain an algorithm for computing the winning regions of parity games using $\mathit{SolveE}(\mathscr{G}, n, n)$, where $\mathscr{G}$ is a parity game with n vertices.

We now perform a complexity analysis. Let us write $f(m,n,d,s_{\text{Eve}},s_{\text{Adam}})$ for the complexity of the algorithm over parity games with m edges, n vertices, d priorities, and parameters s_{Eve} and s_{Adam}. We have $f(m,n,1,s_{\text{Eve}},s_{\text{Adam}}) = O(n)$ and $f(m,0,d,s_{\text{Eve}},s_{\text{Adam}}) = f(m,n,d,0,s_{\text{Adam}}) = f(m,n,d,s_{\text{Eve}},0) = O(1)$, with the general induction

$$\begin{aligned} f(m,n,d,s_{\text{Eve}},s_{\text{Adam}}) \quad \leq \quad & n \cdot f(m,n,d-1,s_{\text{Eve}},\lfloor s_{\text{Adam}}/2 \rfloor) \\ & + f(m,n,d-1,s_{\text{Eve}},s_{\text{Adam}}) \\ & + O(nm). \end{aligned}$$

The term $n \cdot f(m,n,d-1,s_{\text{Eve}},\lfloor s_{\text{Adam}}/2 \rfloor)$ corresponds to the recursive calls for small dominions, the term $f(m,n,d-1,s_{\text{Eve}},s_{\text{Adam}})$ to the recursive call for a large dominion, and $O(nm)$ for the computation of the attractors. We obtain

$$f(m,n,d,n,n) \leq m \cdot n^{1+2\lfloor \log n \rfloor} \cdot \binom{d+2\lfloor \log n \rfloor}{2\lfloor \log n \rfloor}.$$

which implies a quasi-polynomial upper bound of $n^{O(\log n)}$.

3.3 A quasi-polynomial time separating automata algorithm

The separation framework

We describe a general approach for reducing parity games to safety games. Section 1.6 constructs reductions between objectives using automata: in the case at hand parity reduces to safety if there exists a deterministic automaton $\mathbf{A}$ over the alphabet $[1,d]$ with acceptance objective $\texttt{Safe}$ and defining $\texttt{Parity}([1,d])$, meaning $L(\mathbf{A}) = \texttt{Parity}([1,d])$. With such an automaton in hand Lemma 6 implies the following reduction: from a parity game $\mathscr{G}$ construct the safety game $\mathscr{G} \times \mathbf{A}$ satisfying that Eve has a winning strategy in $\mathscr{G}$ from v_0 if and only if she has a winning strategy in $\mathscr{G} \times \mathbf{A}$ from (v_0, q_0).

Unfortunately, it can be shown (using a topological argument) that there is no such automaton. The separation framework defines a (weaker) sufficient condition for the reduction above to be correct. Instead of insisting that $L(\mathbf{A}) = \texttt{Parity}([1,d])$, it is enough to have $W_{\text{Eve}}(\mathscr{A}, \texttt{Parity}[\mathfrak{c}]) = W_{\text{Eve}}(\mathscr{A}, L(\mathbf{A})[\mathfrak{c}])$. To ensure this equality we will only require that $L(\mathbf{A})$ *separates* winning plays from losing plays. The two *key* ideas are first to take advantage of the positionality of parity objectives by restricting further winning plays to *positional* winning plays, and second to require

$$W_{\text{Eve}}(\mathscr{A}, \texttt{Parity}[\mathfrak{c}]) = W_{\text{Eve}}(\mathscr{A}, L(\mathbf{A})[\mathfrak{c}])$$

not for all parity games, but only for parity games with n vertices and priorities in $[1,d]$.

A deterministic safety automaton over the alphabet $[1,d]$ is given by

$$\mathbf{A} = (Q, q_0, \delta : Q \times [1,d] \to Q, \texttt{Safe}[\mathfrak{c}_\mathbf{A}]),$$

where $\mathfrak{c}_\mathbf{A} : Q \times [1,d] \to \{\text{Win}, \text{Lose}\}$. A word $w \in [1,d]^\omega$ is accepted if the run ρ over w only contains transitions in Win. We use the following simplifying convention for safety automata: we distinguish a special rejecting state $\perp$ and assume that all transitions are accepting except the ones leading to $\perp$. Said differently, a word w is accepted if and only if it the run over w does not contain the state $\perp$. Hence we do not need to specify $\mathfrak{c}_\mathbf{A}$: in the next two sections by an automaton we mean a deterministic safety automaton given by $\mathbf{A} = (Q, q_0, \delta : Q \times [1,d] \to Q)$.

Let us now give a sufficient condition for the automaton $\mathbf{A}$ to imply the correctness of the reduction from the parity game $\mathscr{G}$ to the safety game $\mathscr{G} \times \mathbf{A}$. The condition that we define now is that the automaton $\mathbf{A}$ is (n,d)-separating; it relies on the notion of parity graphs. Recall a parity graph *satisfies* parity from v if all infinite paths from v satisfy parity, and that a strategy σ is winning from v if and only if the parity graph $\mathscr{G}[\sigma]$ satisfies parity from v.

We say that an automaton reads, accepts, or rejects a path π in a parity graph, which is an abuse because what the automaton reads is the induced sequence of colours $\mathfrak{c}(\pi)$.

Definition 5 (Separating automata)**.** *An automaton* $\mathbf{A}$ *is* (n,d)*-separating if the two following properties hold.*

- *For all parity graphs G with n vertices and priorities in* $[1,d]$ *satisfying parity, all paths from v are accepted by* $\mathbf{A}$*.*
- *All words accepted by* $\mathbf{A}$ *satisfy parity.*

We define the objective $\texttt{Parity}_{|n}$ over the set of colours $[1,d]$ as:

$$\texttt{Parity}_{|n} = \left\{ \pi : \begin{array}{l} \pi \text{ is a path starting in some parity graph} \\ \text{with } n \text{ vertices and priorities in } [1,d] \text{ satisfying } \texttt{Parity} \end{array} \right\}.$$

The definition of (n,d)-separating automata is illustrated in Figure 3.1 and can be summarised as $\texttt{Parity}_{|n} \subseteq L(\mathbf{A}) \subseteq \texttt{Parity}$.

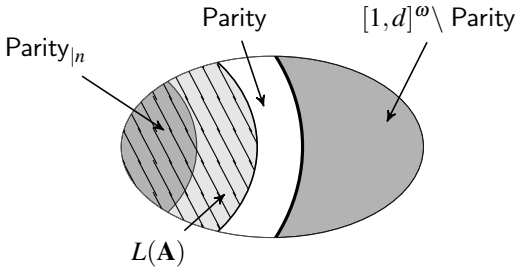


Figure 3.1: The separation problem.

The following lemma shows the definition of separating automata in action.

Lemma 26 (Game equivalence using separating automata)**.** *Let* $\mathbf{A}$ *an* (n,d)*-separating automaton. Then for all parity games* $\mathscr{G} = (\mathscr{A}, \mathtt{Parity}[\mathfrak{c}])$ *with n vertices and priorities in* $[1,d]$*, we have*

$$W_{\mathrm{Eve}}(\mathscr{G}) = W_{\mathrm{Eve}}(\mathscr{A}, L(\mathbf{A})[\mathfrak{c}]).$$

Proof. The inclusion $W_{\mathrm{Eve}}(\mathscr{G}) \subseteq W_{\mathrm{Eve}}(\mathscr{A}, L(\mathbf{A})[\mathfrak{c}])$ follows from positional determinacy and the inclusion $\mathtt{Parity}_{|n} \subseteq L(\mathbf{A})$. Let $v \in W_{\mathrm{Eve}}(\mathscr{G})$. Consider σ a positional strategy ensuring parity from v and construct the parity graph $\mathscr{G}[\sigma]$, it satisfies parity from v. Hence the strategy σ also ensures $L(\mathbf{A})[\mathfrak{c}]$ from v, so $v \in W_{\mathrm{Eve}}(\mathscr{A}, L(\mathbf{A})[\mathfrak{c}])$.

Conversely, $L(\mathbf{A}) \subseteq \mathtt{Parity}$ implies $L(\mathbf{A})[\mathfrak{c}] \subseteq \mathtt{Parity}[\mathfrak{c}]$, which in turn implies $W_{\mathrm{Eve}}(\mathscr{A}, L(\mathbf{A})[\mathfrak{c}]) \subseteq W_{\mathrm{Eve}}(\mathscr{G})$. □

The last step is to explain how to solve a game with objective $L(\mathbf{A})$, as already discussed in Section 1.6. Let $\mathscr{G} = (\mathscr{A}, L(\mathbf{A})[\mathfrak{c}])$. We construct a safety game by making the synchronised product of the arena with the automaton:

$$\mathscr{G} \times \mathbf{A} = (\mathscr{A} \times \mathbf{A}, \mathtt{Safe}[\mathfrak{c}']),$$

where the safety condition ensures that the play is accepted by $\mathbf{A}$. Formally, we construct the arena $\mathscr{A} \times \mathbf{A}$ as follows. We first define the graph $G \times Q$ whose set of vertices is $V \times Q$ and set of edges is defined as follows: for every edge $u \xrightarrow{p} v \in E$ and state $q \in Q$ there is an edge from (u,q) to $(v, \delta(q,p))$. The arena is $\mathscr{A} \times \mathbf{A} = (G \times Q, V_{\mathrm{Eve}} \times Q, V_{\mathrm{Adam}} \times Q)$. Using the convention for safety automata that the rejecting transitions are precisely those leading to the rejecting state $\perp$, the colouring function is defined by $\mathfrak{c}'(v,q) = \mathrm{Win}$ if $q \neq \perp$, and Lose otherwise.

Fact 13 (Reduction to safety games using separating automata)**.** *Eve has a winning strategy in* $\mathscr{G}$ *from* v_0 *if and only if she has a winning strategy in* $\mathscr{G} \times \mathbf{A}$ *from* (v_0, q_0)*.*

Theorem 29 (Algorithm using separating automata)**.** *Let* $\mathbf{A}$ *an* (n,d)*-separating automaton. There exists an algorithm for solving parity games of complexity* $O(m \cdot |\mathbf{A}|)$*.*

Proof. Let $\mathscr{G}$ a parity game with n vertices and priorities in $[1,d]$. Thanks to Lemma 26 and Fact 13, solving $\mathscr{G}$ is equivalent to solving the safety game $\mathscr{G} \times \mathbf{A}$. Thanks to Theorem 12 solving safety games can be done in time linear in the number of edges. This yields an algorithm for solving parity games whose running time is $O(m \cdot |\mathbf{A}|)$. □

In the remainder of this section we give a construction for a quasi-polynomial (n,d)-separating automaton.

The original separating automaton

Theorem 30 (The original separating automaton)**.** *There exists an* (n,d)*-separating automaton of size* $n^{O(\log d)}$*, inducing an algorithm for solving parity games of complexity* $n^{O(\log d)}$*.*

***i*-sequences.** The key definition used by the original separating automaton is an *i-sequence*. Let $\pi = p_1, p_2, \ldots, p_t$ a finite sequence of priorities. An *i*-sequence is a set of *indices* that splits π into sub-sequences. An *i*-sequence consists of exactly 2^i indices $1 \leq j_1 < j_2 < \cdots < j_{2^i} \leq t$, where each j_k is an integer that refers to the priority p_{j_k} from the sequence π. An *i*-sequence is required to satisfy the following properties.

- **Evenness.** Each index (except possibly the last index) refers to an even priority, meaning that p_{j_k} is an even priority for all $k < 2^i$.
- **Inner domination.** The subsequence of priorities between any two indices j_k and j_{k+1} is dominated by either p_{j_k} or $p_{j_{k+1}}$. Formally, this means that whenever $j_k < l < j_{k+1}$, we have that $p_l \leq p_{j_k}$ or we have that $p_l \leq p_{j_{k+1}}$.
- **Outer domination.** The final subsequence between $p_{j_{2^i}}$ and p_t is dominated by $p_{j_{2^i}}$, meaning that for all $l > j_{2^i}$ we have $p_l \leq p_{j_{2^i}}$.

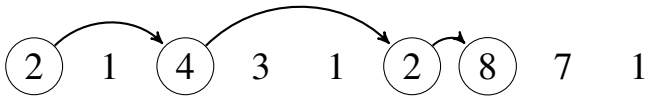


Figure 3.2: A 2-sequence.

Figure 3.2 gives an example of a 2-sequence. The circled priorities are the indices used in the sequence. Note that there are exactly $2^2 = 4$ indices used, and that every circled priority is even. Inner domination is satisfied because every priority that is between two circled priorities is lower than one of the two end points, and outer domination is satisfied because the final circled priority 8 is larger than all the priorities that come after it.

The relationship to parity games. The relationship between *i*-sequences and parity games is explained by the following lemma.

Lemma 27 (Completeness for the separating automaton)**.** *Suppose that Adam and Eve play positional strategies in the parity game, and let π the resulting play.*

- *If Eve wins the parity game, then there exists prefixes of π that contain arbitrarily long i-sequences.*
- *If Adam wins the parity game, then no prefix of π will contain a $\lceil \log n \rceil$-sequence.*

Proof. If Eve wins the parity game then the largest priority occurring in π infinitely often is even. Let p this priority. To construct a prefix containing an *i*-sequence, we find the first index j after which no priority $q > p$ is seen. We take as our indices $j_1 < j_2 < \cdots < j_{2^i}$ the first 2^i occurrences of priority p after index j. Evenness is trivially satisfied, and both inner and outer domination are satisfied because no priority larger than p is seen after index j.

We prove the second claim by contradiction. Suppose that Adam wins the game, but that there is a prefix of π that contains a $\lceil \log n \rceil$-sequence. Since the sequence indexes $2^{\lceil \log n \rceil} \geq n$ vertices of the game, it must index the same vertex v twice. Thus our i-sequence must contain a cycle passing through v. Note that inner domination ensures that the largest priority on the cycle that passes through v is even. However, no even cycle can be formed when Adam wins the game by playing a positional winning strategy, and so we have arrived at our contradiction. □

To summarise, if Eve wins the game, then she has a strategy that ensures that arbitrarily long i-sequences occur, while if Adam wins the game then he has a strategy that ensures that no $\lceil \log n \rceil$-sequence occurs. So to solve the parity game, it is sufficient to determine whether Eve can force a $\lceil \log n \rceil$-sequence to occur.

A data structure for recognising i-sequences. We will build a quasi-polynomial sized automaton that reads a sequence of priorities, and determines whether that sequence of priorities contains a k-sequence. The automaton is defined by a data structure that we call a *record*, which contains information about the i-sequences that have been seen so far in the sequence.

A record is a sequence $b_k, b_{k-1}, \ldots, b_1, b_0$, where each b_i is either a priority, or the special symbol $-$. The value of b_i has the following meaning:

- **Witnessing.** If $b_i \neq -$, then we have seen an i-sequence, and the final priority on that i-sequence is b_i.
- **Order.** If $b_i \neq -$ and $b_j \neq -$ and $j < i$, then the first index of the j-sequence witnessed by b_j occurs after the last index of the i-sequence witnessed by b_i.

Note that, although each element b_i records the existence of an i-sequence, the record data structure *does not* store the 2^i indices of this i-sequence, it only stores the priority of the final index of that sequence.

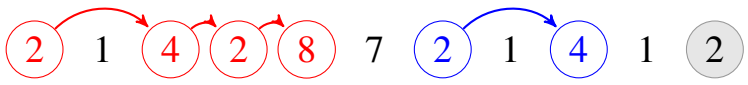


Figure 3.3: An example sequence that corresponds to the record -842.

Figure 3.3 shows an example sequence that is consistent with the record that sets $b_3 = -$, $b_2 = 8$, $b_1 = 4$, and $b_0 = 2$. The red 2-sequence is represented by $b_2 = 8$, which is the last priority of the 2-sequence. The blue 1-sequence starts after the end of the 2-sequence, and it is represented by $b_1 = 4$. Likewise the grey 0-sequence starts after the end of the 1-sequence, and is represented by $b_0 = 2$. There is no 3-sequence in the example, and this is represented by setting $b_3 = -$.

The update rule. Suppose that we have a record that represents the i-sequences in a finite sequence of priorities, and that we then read the next priority p in that sequence. We need to update the record to take this priority into account. We do this by applying

the following *update rule*. The update rule consists of two steps, which occur one after the other.

- **Step 1.** In this step, we find the largest index i such that b_j is even for all $j \leq i$. If $b_i = -$, or $b_i < p$, then we create a new record $b'_k, b'_{k-1}, \ldots, b'_0$ by setting:

$$b'_j = \begin{cases} b_j & \text{if } j > i, \\ p & \text{if } j = i, \\ - & \text{if } j < i. \end{cases}$$

  If there is no index i that satisfies the conditions, then we do not modify the record.

- **Step 2.** In step 2, we take the output of step 1, and we find the largest index i such that $p > b_i$ and we create a new record $b'_k, b'_{k-1}, \ldots, b'_0$ by setting:

$$b'_j = \begin{cases} b_j & \text{if } j > i, \\ p & \text{if } j = i, \\ - & \text{if } j < i. \end{cases}$$

  Again, if there is no such index i, then the record is not modified.

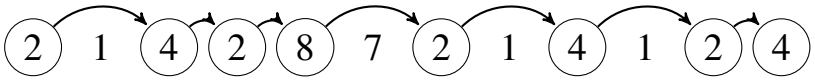


Figure 3.4: An example of a Step 1 update applied to the sequence and record from Figure 3.3

Intuitively, Step 1 attempts to combine the i-sequences in the existing record into a longer i-sequence. Suppose that we have read the sequence shown in Figure 3.3, that we have compute the record -842, and that the next priority in the sequence is 4. Figure 3.4 shows the result of applying Step 1 to this situation. Observe that 3 is the largest index i such that for all $j < i$ we have that b_j is even, so Step 1 will output the record $4---$.

So in this circumstance, Step 1 claims that we have now seen a 3-sequence. Figure 3.4 shows why this is correct: the 0-sequence of b_0, the 1-sequence of b_1, and the 2-sequence of b_2 can be merged together, along with the new priority, to create a 3-sequence. Observe that inner domination in this new 3-sequence is satisfied due to the outer domination property for each of the i-sequences that it was constructed from. For example, the 2-sequence ends at priority 8, and the 1-sequence begins at priority 2, and we know that 8 must dominate all priorities between the 8 and the 2 because 8 is required to dominate *all* priorities that follow it.

Step 2 ensures that the outer domination property holds. In Figure 3.5, we show the result of applying Step 2 to the record -842 that corresponds to the sequence shown

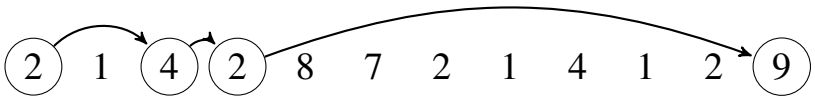


Figure 3.5: An example of a Step 2 update applied to the sequence and record from Figure 3.3

in Figure 3.3, when the next priority in the sequence is 9. Observe that since $9 > 8$, the outer domination property for the 2-sequence ending at 8 now fails to hold, and likewise for the sequences ending at 4 and 2. Hence, Step 2 deletes the 0-sequence and 1-sequence from the record, and updates the 2-sequence to end at 9, thereby restoring outer domination. The resulting record is $-9--$.

Correctness. To compute a record for a particular sequence of priorities, we start with the record $--\ldots-$, and then process the sequence one priority at a time, using the update rule that we have described.

We must now argue that the record data structure and update rule is sufficient to decide the winner of a parity game. The following lemma states that a record will never falsely claim that an i-sequence has occurred.

Lemma 28 (Correctness for the separating automaton)**.** *Let $b_k, b_{k-1}, \ldots, b_0$ the record for a sequence of priorities π. If $b_i \neq -$, then π contains an i-sequence.*

Proof. This can be proved by induction over the components of the record. In fact we will prove the slightly stronger order property that we mentioned earlier: the i-sequence corresponding to b_i starts after the j sequence corresponding to b_j whenever $i < j$ and $b_i \neq -$ and $b_j \neq -$.

The base case is trivially true, since the value of b_0 asserts the existence of a 0-sequence, and any priority by itself is a 0-sequence. So when Step 1 or Step 2 updates b_0, the corresponding 0-sequence is the new priority, and this clearly starts after all other i-sequences in the record.

For the inductive step, we must prove that the two steps of the update rule are correct.

- For Step 1 updates, we can use the inductive hypothesis to argue that, for each $j < i$, the j-sequence corresponding to b_j exists, and that they appear in order in the sequence, and that it ends before the sequence corresponding to b_{j-1} starts. Furthermore, the outer domination property the j-sequence ensures that all priorities between the end of the j-sequence and the start of the $(j-1)$-sequence are dominated by the last priority in the j-sequence, which must be even according to the definition of a Step 1 update. Hence, we can combine all of the j-sequences with $j < i$ together, along with the new priority, to create an i-sequence. This new i-sequence starts at exactly the same point as the sequence corresponding to b_{i-1}, and so the order property still holds.

- For Step 2 updates, we only need to argue that the value of b'_i correspond to an i sequence. This can be constructed as we showed in Figure 3.5: take the i-sequence that corresponds to b_i, and replace the final priority with the new priority. Observe that the final priority of an i-sequence is permitted to be odd, and so this new sequence satisfies all of the requirements of an i-sequence. Furthermore, the starting point of this sequence has not changed, and so the order property is preserved.

□

As a consequence of the lemma above, if Adam has a strategy to ensure that no k-sequence occurs in the game, then Adam has a strategy to ensure that the b_k component of the record is never set so that $b_k \neq -$.

It can be shown that the other direction is also true: if an i-sequence has occurred, then there will be some index $j \geq i$ such that $b_j \neq -$. However, the proof is somewhat tedious, and this statement is actually stronger than what we need. To argue that the record can determine the winner of a parity game, the following weaker lemma suffices.

Lemma 29 (Weaker correctness for the separating automaton)**.** *Let π an infinite play that is winning for Eve. For all k, there exists a prefix of π such that $b_k \neq -$.*

Proof. Let p the largest even priority that is seen infinitely often, and let j be the first index after which no priority larger than p is visited. We argue that after index j has been reached, the record will eventually set $b_i \neq -$ for all i.

To see why, observe that after index j has been reached, Step 2 cannot replace any component b_j with $b_j = p$, since Step 2 can only overwrite the priority in b_j when the new priority p' satisfies $p' > b_j$, but no priority $p' > p$ is seen after index j.

On the other hand, Step 1 will always be triggered whenever we visit the priority p. Step 1 will always set some component of the record to p, and as we have observed this cannot be overwritten by Step 2. Moreover, since p is even, repeated application of Step 1 will build a longer and longer i-sequences whose outer domination priority is p. Thus, after we have made 2^k visits to p, we will have set $b_k = p \neq -$, if we have not done so already. □

Hence, if Eve wins the parity game, then she has a strategy to eventually ensure that $b_k \neq -$. Combining the two lemmas above, with Lemma 27 gives the following corollary.

Corollary 4 (Correctness of the reduction for the separating automaton)**.** *Suppose that we monitor the play of a parity game with a record $b_{\lceil \log n \rceil}, \ldots, b_0$. Eve has a strategy that ensures $b_{\lceil \log n \rceil} \neq -$ if and only if Eve wins the parity game.*

The size of the automaton. The record data structure and update rule can be encoded as a deterministic finite automaton that reads the play. Each state of the automaton is associated with some configuration of $b_{\lceil \log n \rceil}, \ldots, b_0$, and the transitions of the automaton are defined by the update rule.

This automaton has quasi-polynomial size. The number of states used in the automaton is the number of possible configurations of $b_{\lceil \log n \rceil}, \ldots, b_0$. Each b_i can be one

of the d priorities in the game, or the symbol $-$, and so there are $d+1$ possible values that it can take. Moreover there are $\log n + 1$ components of the record, so the total number of configurations is at most $(d+1)^{\log n+1} = n^{O(\log d)}$.

3.4 A quasi-polynomial time value iteration algorithm

Theorem 31 (Value iteration)**.** *There exists a value iteration algorithm for solving parity games in time*

$$O\left(nm \cdot \log(n)\log(d) \cdot n^{2.45+\log_2\left(1+\frac{d/2-1}{\log_2(n)}\right)}\right),$$

which is quasi-polynomial in general and polynomial if $d = O(\log_2(n))$. The space complexity of the algorithm is quasi-linear, and more precisely $O(m + n\log_2(d))$.

The presentation follows the introduction to value iteration algorithms given in Section 1.9, although it does not technically rely on it. Let $\mathscr{G} = (\mathscr{A}, \texttt{Parity}[\mathfrak{c}])$ a parity game with n vertices and priorities in $[1,d]$, and without loss of generality d is even.

The first step is to define a notion of value function $\text{val}^{\mathscr{G}} : V \to Y$ with $(Y, \leq)$ a lattice satisfying the characterisation principle: for all vertices u we have that Eve wins from u if and only if $\text{val}^{\mathscr{G}}(u) \neq \top$, where $\top$ is the largest element in Y. The goal of the algorithm is to compute $\text{val}^{\mathscr{G}}$, from which we then easily obtain the winning region thanks to the characterisation principle.

We let F_V be the lattice of functions $V \to Y$ equipped with the componentwise order induced by Y. We are looking for a monotonic function $\delta : Y \times [1,d] \to Y$ inducing the operator $\mathbb{O} : F_V \to F_V$ defined by:

$$\mathbb{O}(\mu)(u) = \begin{cases} \min\left\{\delta(\mu(v), p) : u \xrightarrow{p} v \in E\right\} & \text{if } u \in V_{\text{Eve}}, \\ \max\left\{\delta(\mu(v), p) : u \xrightarrow{p} v \in E\right\} & \text{if } u \in V_{\text{Adam}}, \end{cases}$$

such that $\text{val}^{\mathscr{G}}$ is the least fixed point of $\mathbb{O}$. The algorithm will then simply use Kleene's fixed point theorem (Theorem 4) to compute $\text{val}^{\mathscr{G}}$ by iterating the operator $\mathbb{O}$.

Let us look at this question using the notion of progress measures, which are pre-fixed points of $\mathbb{O}$, meaning μ such that $\mathbb{O}(\mu) \leq \mu$. Since the least fixed point of $\mathbb{O}$ is also its least pre-fixed point, an equivalent formulation of the characterisation principle above reads: for all vertices u we have that Eve wins from u if and only if there exists a progress measure μ such that $\mu(u) \neq \top$.

To summarise this discussion, we are looking for a lattice $(Y, \leq)$ and a monotonic function $\delta : Y \times [1,d] \to Y$ such that for all parity games $\mathscr{G}$ with n vertices and priorities in $[1,d]$, for all vertices u we have that Eve wins from u if and only if there exists a progress measure μ such that $\mu(u) \neq \top$. Our next step is to show how the notion of universal trees provides a class of solutions to this problem.

Universal trees

The trees we consider have three properties: they are rooted, every leaf has the same depth, and the children of a node are totally ordered. Formally, a tree of height 0 is a leaf, and a tree t of height $h+1$ is an ordered list $[t_1,\dots,t_k]$ of subtrees each of height h.

We consider two parameters for trees: the height, and the size which is defined to be the number of leaves.

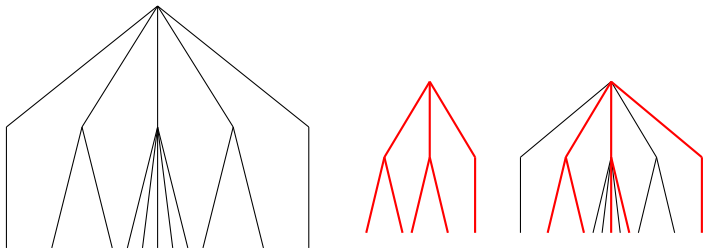


Figure 3.6: On the left, a tree of height $h=2$, which is the smallest $(5,2)$-universal tree: it has size 11 (meaning it has 11 leaves). On the right, a tree of size 5 and one possible embedding into the universal tree.

We say that a tree t embeds into another tree T if:

- either both are leaves,
- or let $t=[t_1,\dots,t_k]$ and $T=[T_1,\dots,T_{k'}]$, there exist $i_1<\dots<i_k$ such that for all $j\in[1,k]$ we have that t_j embeds into T_{i_j}.

Definition 6 (Universal trees)**.** *A tree is (n,h)-universal if it embeds all trees of size n and height h.*

We refer to Figure 3.6 for an example of a $(5,2)$-universal tree. A first example of an (n,h)-universal tree is the tree where each node has degree n: formally we define it recursively by $T_{n,0}$ is a leaf, and $T_{n,h+1}=[\underbrace{T_{n,h},\dots,T_{n,h}}_{n\text{ copies}}]$. It has size n^h.

A quasi-polynomial universal tree

We present an inductive construction of a quasi-polynomial universal tree.

Theorem 32 (Construction of a quasi-polynomial universal tree)**.** *There exists an (n,h)-universal tree with size $f(n,h)$ satisfying the following:*

$$\begin{aligned} f(n,h) &= f(n,h-1)+f(\lfloor n/2\rfloor,h)+f(n-1-\lfloor n/2\rfloor,h),\\ f(n,0) &= 1,\\ f(0,h) &= 0. \end{aligned}$$

An upper bound is given by

$$f(n,h)\le n\cdot\binom{h-1+\lfloor\log_2(n)\rfloor}{\lfloor\log_2(n)\rfloor}.$$

This is also bounded from above by $n^{2.45+\log_2\left(1+\frac{h-1}{\log_2(n)}\right)}$*, which is quasi-polynomial in* n *and* h *in general, and polynomial if* $h = O(\log_2(n))$.

Proof. To construct the (n,h)-universal tree T, let:

- T_{left} be a $(\lfloor n/2 \rfloor, h)$-universal tree,
- T_{middle} be a $(n, h-1)$-universal tree,
- T_{right} be a $(n-1-\lfloor n/2 \rfloor, h)$-universal tree.

The intuitive construction of T is as follows: we merge the roots of T_{left} and T_{right} and insert in-between them a child of the root to which is attached T_{middle}. Formally, let $T_{\text{left}} = [T^1_{\text{left}}, \ldots, T^k_{\text{left}}]$ and $T_{\text{right}} = [T^1_{\text{right}}, \ldots, T^{k'}_{\text{right}}]$, we define T as

$$[T^1_{\text{left}}, \ldots, T^k_{\text{left}},\ T_{\text{middle}},\ T^1_{\text{right}}, \ldots, T^{k'}_{\text{right}}].$$

The construction is illustrated in Figure 3.7.

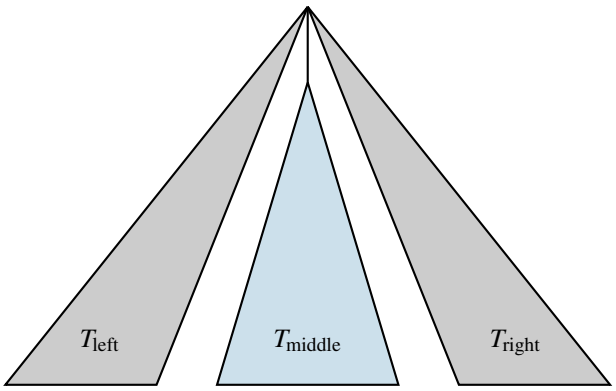


Figure 3.7: The inductive construction.

We argue that T is (n,h)-universal. Consider a tree $t = [t_1, \ldots, t_k]$ with n leaves. The question is where to cut, *i.e.* which subtree of t gets mapped to T_{middle}. Let $n(t_i)$ be the number of leaves in t_i. Since t has n leaves, we have $n(t_1) + \cdots + n(t_k) = n$. There exists a unique $p \in [1,k]$ such that $n(t_1) + \cdots + n(t_{p-1}) \leq \lfloor n/2 \rfloor$ and $n(t_1) + \cdots + n(t_p) > \lfloor n/2 \rfloor$. The choice of p implies that $n(t_{p+1}) + \cdots + n(t_k) \leq n - 1 - \lfloor n/2 \rfloor$. To embed t into T, we proceed as follows:

- the tree $[t_1, \ldots, t_{p-1}]$ has at most $\lfloor n/2 \rfloor$ leaves, so it embeds into T_{left} by induction hypothesis;
- the tree t_p has height $h-1$ and at most n leaves, so in embeds into T_{middle} by induction hypothesis;
- the tree $[t_{p+1}, \ldots, t_k]$ has at most $n - 1 - \lfloor n/2 \rfloor$ leaves, so it embeds into T_{right} by induction hypothesis.

□

The construction given in the proof yields the smallest $(5,2)$-universal tree illustrated in Figure 3.6.

Ordering the leaves

Let us consider a tree t of height h, and write $d = 2h$. A leaf is given by a list of directions that we define now. For technical convenience that will manifest itself later, the list of directions is indexed by odd numbers $p \in [1,d]$ downwards: for example for $d = 10$ a leaf is $(D_9, D_7, D_5, D_3, D_1)$.

We write Y_t for the set of leaves of t and $\leq$ for the lexicographic order on Y_t. Note that its interpretation on the tree is: for two leaves ℓ, ℓ', we have $\ell \leq \ell'$ if and only if ℓ is to the left of ℓ'. The strict version of $\leq$ is $<$.

We introduce a set of relations $\lhd_p$ over Y_t for each $p \in [1,d]$. For a leaf $\ell = (D_{d-1}, \dots, D_1)$ we write $\ell_{\geq p}$ for the tuple $(D_{d-1}, \dots, D_p)$, which we call the p-truncated branch of ℓ.

- For p odd, we say that $\ell \lhd_p \ell'$ if $\ell_{\geq p} < \ell'_{\geq p}$.
- For p even, we say that $\ell \lhd_p \ell'$ if $\ell_{\geq p} \leq \ell'_{\geq p}$.

To interpret $\lhd_p$ on the tree, we label the levels by priorities from bottom to top as in Figure 3.6. Then $\ell \lhd_p \ell'$ if and only if the p-truncated branch of ℓ is to the left of the p-truncated branch of ℓ', strictly if p is odd, and non-strictly if p is even.

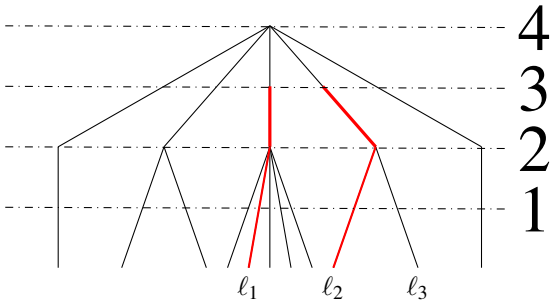


Figure 3.8: Illustration of the relations $\lhd_p$.

We refer to Figure 3.8 for some examples. In red, we see that $\ell_1 \lhd_3 \ell_2$.

$$\ell_2 \lhd_1 \ell_3 \quad ; \quad \ell_2 \lhd_2 \ell_3 \quad ; \quad \ell_3 \lhd_2 \ell_2 \quad ; \quad \ell_1 \lhd_3 \ell_2 \quad ; \quad \ell_1 \lhd_2 \ell_2.$$

Lemma 30 (Properties of the tree orders)**.** *The relations $\lhd_p$ for $p \in [1,d]$ induced by a tree t satisfy the following properties:*

- *$\lhd_d$ is the full relation: for all b, b' we have $b \lhd_d b'$;*
- *if $\ell \lhd_p \ell'$ and $\ell' \lhd_q \ell''$ then $\ell \lhd_{\max(p,q)} \ell''$;*

- *the relation $\lhd_p$ is non-reflexive if p is odd;*
- *the relation $\lhd_1$ is total;*
- *for $p < d$ even we have $\ell \lhd_p \ell'$ if and only if $\neg(\ell' \lhd_{p+1} \ell)$.*

The following observation rephrases the notion of embeddings between trees using the ordering on leaves.

Fact 14 (Equivalence between embedding and orders)**.** *Let t, T be two trees of height h and $d = 2h$. Then t embeds into T if and only if there exists a function $\mu : Y_t \to Y_T$ such that for all leaves ℓ, ℓ', for all $p \in [1,d]$:*

$$\ell \lhd_p^t \ell' \implies \mu(\ell) \lhd_p^T \mu(\ell').$$

Progress measures

We explain how a tree t induces both a lattice $(Y_t, \leq)$ and a monotonic function $\delta_t : Y_t \times [1,d] \to Y_t$. The set Y_t is the set of leaves of t augmented with a new element $\top$, and $\leq$ is the lexicographic order on leaves with $\top$ as greatest element. For each $p \in [1,d]$ and $\ell \in Y_t$ we extend $\lhd_p$ with $\ell \lhd_p \top$. We then define $\delta_t : Y_t \times [1,d] \to Y_t$ by

$$\delta_t(\ell, p) = \min_{\leq} \left\{ \ell' : \ell \lhd_p \ell' \right\}.$$

This in turn induces a monotonic operator $\mathbb{O}_t : F_V \to F_V$ defined by:

$$\mathbb{O}_t(\mu)(u) = \begin{cases} \min \left\{ \delta_t(\mu(v), p) : u \xrightarrow{p} v \in E \right\} & \text{if } u \in V_{\text{Eve}}, \\ \max \left\{ \delta_t(\mu(v), p) : u \xrightarrow{p} v \in E \right\} & \text{if } u \in V_{\text{Adam}}, \end{cases}$$

Let $\mathcal{G}$ be a parity game, a progress measure is a function $\mu : V \to Y_t$ which is a pre-fixed point: $\mathbb{O}_t(\mu) \leq \mu$. Unfolding the definitions, this means that for all vertices u, we have

$$\begin{array}{ll} \exists u \xrightarrow{p} v \in E, & \delta_t(\mu(v), p) \leq \mu(u) \quad \text{if } u \in V_{\text{Eve}}, \\ \forall u \xrightarrow{p} v \in E, & \delta_t(\mu(v), p) \leq \mu(u) \quad \text{if } u \in V_{\text{Adam}}. \end{array}$$

The definition of δ_t further simplifies it to: for all vertices u, we have

$$\begin{array}{ll} \exists u \xrightarrow{p} v \in E, & \mu(v) \lhd_p \mu(u) \quad \text{if } u \in V_{\text{Eve}}, \\ \forall u \xrightarrow{p} v \in E, & \mu(v) \lhd_p \mu(u) \quad \text{if } u \in V_{\text{Adam}}. \end{array}$$

The following theorem is our first and main step towards proving the characterisation principle.

Theorem 33 (Fundamental theorem for progress measures)**.** *Let $\mathcal{G}$ be a parity game and v a vertex. Then Eve wins from v if and only if there exists a tree t and a progress measure $\mu : V \to Y_t$ such that $\mu(v) \neq \top$.*

In order to prove Theorem 33, we first consider the case of parity graphs. A progress measure in a parity graph is a function $\mu : V \to Y_t$ such that for all edges $u \xrightarrow{p} v \in E$ we have $\mu(v) \lhd_p \mu(u)$.

Recall that a graph satisfies parity from v if all infinite paths from v satisfy parity. This is equivalent to asking whether all cycles reachable from v are even, meaning the maximal priority appearing in the cycle is even.

Lemma 31 (Fundamental lemma for progress measures over graphs)**.** *Let G be a parity graph and v a vertex. Then G satisfies parity from v if and only if there exists a tree t and a progress measure $\mu : V \to Y_t$ such that $\mu(v) \neq \top$.*

Proof. Let us assume that there exists a tree t and a progress measure $\mu : V \to Y_t$ such that $\mu(v) \neq \top$ and for all edges $u \xrightarrow{p} v \in E$ we have $\mu(v) \lhd_p \mu(u)$. To show that G satisfies parity from v we show that any cycle reachable from v is even. Let us consider such a cycle:

$$v_1 \xrightarrow{p_1} v_2 \xrightarrow{p_2} v_3 \cdots v_k \xrightarrow{p_k} v_1.$$

Since the cycle is reachable from v and $\mu(v) \neq \top$, this implies that $\mu(v_i) \neq \bot$ for $i \in [1,k]$. Let us assume towards contradiction that its maximal priority is odd, and without loss of generality it is p_1. Applying our hypothesis to each edge of the cycle we have

$$\mu(v_1) \lhd_{p_k} \mu(v_k) \lhd_{p_{k-1}} \cdots \lhd_{p_2} \mu(v_k) \lhd_{p_1} \mu(v_1).$$

The second item of Lemma 30 implies that $\mu(v_1) \lhd_{p_1} \mu(v_1)$, which contradicts the third item since $\lhd_{p_1}$ is non-reflexive given that p_1 is odd.

Let us now prove the converse implication. We prove the following property by induction on the number of priorities: for all graphs satisfying parity (without the usual assumption that every vertex has an outgoing edge), there exists a tree t and a progress measure $\mu : V \to Y_t$ such that $\mu(v) \neq \top$ for all vertices $v \in V$.

Let G a graph satisfying parity. Without loss of generality the largest priority d in the graph is even. Let us define G' the graph obtained from G by removing all edges with priority d. We consider its decomposition in strongly connected components: let $G_1, \ldots, G_k$ denote the strongly connected components of G', numbered so that for any edge $u \to v \in E(G')$, if $u \in V(G_i)$ then $v \in V(G_j)$ for $j \geq i$. A stronger property holds for edges $u \xrightarrow{d-1} v \in E(G')$: if $u \in V(G_i)$ then $v \in V(G_j)$ for $j > i$. Indeed, if $v \in V(G_i)$ then we could form a cycle in G' whose largest priority is $d-1$, a contradiction. Hence each G_i has priorities in $[1, d-2]$, and being a subgraph of G it satisfies parity. By induction hypothesis, there exists a tree t_i and a progress measure $\mu_i : V(G_i) \to Y_{t_i}$ such that $\mu_i(v) \neq \top$ for all vertices $v \in V(G_i)$.

Let us define $t = [t_1, \ldots, t_k]$, and the function $\mu : V(G) \to Y_t$ by $\mu(v) = \mu_i(v)$ if $v \in V(G_i)$. We claim that μ is a progress measure: let us consider an edge $u \xrightarrow{p} v \in E(G)$, then

- if $p = d$, then $\mu(v) \lhd_d \mu(u)$ because $\lhd_d$ is the full relation;
- if $p = d-1$, then $\mu(v) \lhd_{d-1} \mu(u)$ because $u \in V(G_i)$ and $v \in V(G_j)$ for $j > i$;

- if $p < d-1$, then $\mu(v) \lhd_p \mu(u)$. Indeed $u \in V(G_i)$ and $v \in V(G_j)$ for $j \geq i$, so either $j = i$ and this follows from the fact that μ_i is a progress measure, or $j > i$ and we have $\mu(v) \lhd_{d-1} \mu(u)$ so a fortiori $\mu(v) \lhd_p \mu(u)$.

□

We can now prove Theorem 33.

Proof. Assume that Eve wins from v and let σ be a positional strategy. The parity graph $\mathscr{G}[\sigma]$ satisfies parity from v, so thanks to Lemma 31 there exists a tree t and a function $\mu : V \to Y_t$ such that $\mu(v) \neq \top$ and for all edges $u \xrightarrow{p} v \in E$ we have $\mu(v) \lhd_p \mu(u)$. We remark that $\mu : V \to Y_t$ is actually a progress measure: the condition for $u \in V_{\text{Eve}}$ is ensured by the edge $\sigma(u)$, and the condition for $v \in V_{\text{Adam}}$ by assumption on μ.

Conversely, assume that there exists a tree t and a progress measure $\mu : V \to Y_t$. It induces a positional strategy defined by $\sigma(u) = u \xrightarrow{p} v$ such that $\mu(v) \lhd_p \mu(u)$. We argue that σ is a winning strategy from any vertex v such that $\mu(v) \neq \top$. This is a consequence of Lemma 31 for the parity graph $\mathscr{G}[\sigma]$. □

Theorem 33 is very close to the characterisation principle we are after, the only difference being that the lattice $(Y_t, \leq)$ depends on an existentially quantified tree t. This is where we use universal trees:

Corollary 5 (Fundamental corollary for progress measures)**.** *Let $\mathscr{G}$ be a parity game with n vertices and priorities in $[1,d]$, and v a vertex. Let T be a $(n,d/2)$-universal tree. Then Eve wins from v if and only if there exists a progress measure $\mu : V \to Y_T$ such that $\mu(v) \neq \top$.*

Proof. Assume that Eve wins from v, thanks to Theorem 33 there exists a tree t and a progress measure $\mu : V \to Y_t$ such that $\mu(v) \neq \top$. Since T is $(n,d/2)$-universal and t has at most n leaves, t embeds into T, which thanks to Fact 14 implies that there exists $\mu' : Y_t \to Y_T$ respecting the relations $\lhd$. We extend it to $\mu' : Y_t \to Y_T$ by $\mu'(\bot) = \bot$. Then the composition $\mu' \circ \mu : V \to Y_T$ is a progress measure such that $(\mu' \circ \mu)(v) \neq \bot$.

The converse implication is a direct consequence of Theorem 33. □

We have thus proved that the characterisation principle holds for any $(n,d/2)$-universal tree.

The algorithm

Let us fix T an $(n,d/2)$-universal tree. It induces both a lattice $(Y_T, \leq)$ and a monotonic function $\delta_T : Y_T \times [1,d] \to Y_T$, which in turn induces a monotonic operator $\mathbb{O}_T : F_V \to F_V$. Since T is fixed we do not specify the subscript T for all these objects.

The last step is to construct an algorithm returning the minimal progress measure relying on Kleene's fixed point theorem (stated as Theorem 4). The generic algorithm is explained in Section 1.9, let us instantiate it here.

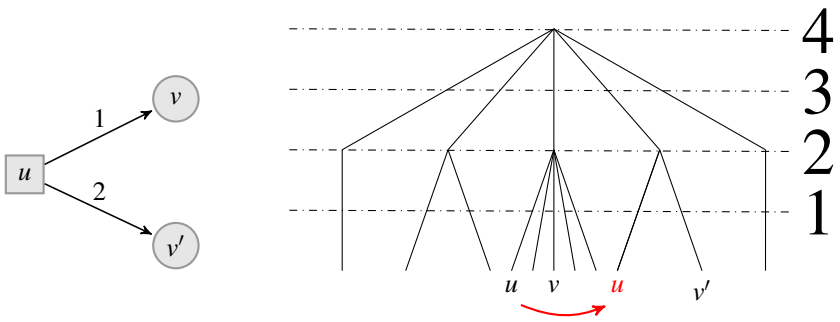


Figure 3.9: The operator $\mathbb{O}$ in action: the updated value for u is the minimal leaf (meaning the leftmost leaf) which satisfies $\mu(v) \lhd_1 \ell$ and $\mu(v') \lhd_2 \ell$.

We say that an edge $u \xrightarrow{p} v$ is *incorrect* if $\neg(\mu(v) \lhd_p \mu(u))$, and a vertex u is *incorrect* if either $u \in V_{\text{Eve}}$ and all outgoing edges are incorrect or $u \in V_{\text{Adam}}$ and there exists an incorrect outgoing edge.

The pseudocode for the algorithm is given in Algorithm 3.4, where we let $\ell_{\min}$ denote the minimal leaf in T.

Algorithm 3.4: The value iteration algorithm.

Data: A parity game with n vertices priorities in $[1,d]$ and a $(n,d/2)$-universal tree T.
for $v \in V$ **do**
  $\mu(v) \leftarrow \ell_{\min}$
repeat
  $\mu \leftarrow \mathbb{O}(\mu)$
until $\mu = \mathbb{O}(\mu)$
return μ

Theorem 34 (Generic value iteration algorithm)**.** *For all $(n,d/2)$-universal tree T, for all parity games $\mathcal{G}$ with n vertices and priorities in $[1,d]$, the value iteration algorithm over the tree T returns the minimal progress measure μ for $\mathcal{G}$ over T.*

Thanks to Corollary 5, the minimal progress measure yields a solution for parity games: Eve wins from v if and only if $\mu(v) \neq \top$.

Complexity analysis

The number of times the operator $\mathbb{O}$ is used is bounded by the number of leaves of T, which we write $|T|$, implying that the total number of iterations is bounded by $n \cdot |T|$. To determine the overall complexity we need to discuss two aspects of the algorithm:

- the data structure and in particular the choice of the vertex u in the loop;

- the computation of $\mathbb{O}$ and in particular the encoding of leaves of T.

Recall that a vertex $u \in V_{\text{Eve}}$ is incorrect if and only if all its outgoing edges are incorrect, and a vertex $u \in V_{\text{Adam}}$ is incorrect if and only if it has a incorrect outgoing edge. Hence checking whether a vertex u is incorrect requires considering all of its outgoing edges $u \xrightarrow{p} v$ and checking whether $\mu(v) \lhd_p \mu(u)$. Let us write Δ for the complexity of checking whether $\mu(v) \lhd_p \mu(u)$. Hence checking whether u is incorrect costs $O(|\text{In}^{-1}(u)| \cdot \Delta)$, where $|\text{In}^{-1}(u)|$ is the number of outgoing edges of u. A naive implementation of Algorithm 3.4 would in each repeat loop go through every vertex u to check whether it is incorrect. This would incur the following cost for a single iteration

$$\sum_{u \in V} O(|\text{In}^{-1}(u)| \cdot \Delta) = O(n \cdot m \cdot \Delta).$$

Thus the overall running time of the algorithm would be

$$O((n \cdot m \cdot \Delta) \cdot (n \cdot |T|)) = O(n^2 \cdot m \cdot \Delta \cdot |T|).$$

Typically Δ is small (we will see that for a well chosen universal tree T it is polylogarithmic in n and d), and T is the dominating factor (quasi-polynomial in n and d thanks to Theorem 32).

A less naive implementation maintains the list of incorrect vertices using a better data structure, saving a linear factor in the complexity. We first explain this, and then discuss the cost Δ by choosing an appropriate encoding of the quasi-polynomial universal tree constructed in Theorem 32.

Data structure

We use a data structure similar to the attractor computation presented in Section 2.1, also presented in general terms in Section 1.9. The pseudocode is given in Algorithm 3.5.

The data structure consists of the following objects:

- a leaf of T for each vertex, representing the current function $\mu : V \to Y$;
- a set `Incorrect` of vertices (the order in which vertices are stored and retrieved from the set does not matter);
- a table `Count` storing for each vertex of Eve a number of edges.

For our complexity analysis we use the unit cost RAM model, see Section 1.2 for details. In the case at hand let us choose for the machine word size $w = \log_2(m) + \log_2(d)$, so that an edge together with its priority can be stored in one machine word. The space complexity of this data structure depends on the encoding of T, which we will discuss later.

The invariant of the algorithm satisfied before each iteration of the repeat loop is the following:

- for $u \in V_{\text{Eve}}$, the value of $\texttt{Count}(u)$ is the number of incorrect edges of u;

- `Incorrect` is the set of incorrect vertices.

The invariant is satisfied initially thanks to the function `Init`. Let us assume that we choose and remove u from `Incorrect`. Since we modify only $\mu(u)$ the only potentially incorrect vertices are in `Incorrect` (minus u) and the incoming edges of u; for the latter each of them is checked and added to `Incorrect`$'$ when required. By monotonicity, incorrect vertices remain incorrect so all vertices in `Incorrect` (minus u) are still incorrect. Hence the invariant is satisfied.

The invariant implies that the algorithm indeed implements Algorithm 3.4 hence returns the minimal progress measure, but it also has implications on the complexity. Indeed one iteration of the repeat loop over some vertex u involves

$$O\left((|\mathrm{In}^{-1}(u)| + |\mathrm{Out}^{-1}(u)|) \cdot \Delta\right)$$

operations, the first term corresponds to updating $\mu(u)$ and `Incorrect`, which requires for each outgoing edge of u to compute δ, and the second term corresponds to considering all incoming edges of u. Thus the overall complexity for a single iteration is

$$O\left(\sum_{u \in V} (|\mathrm{In}^{-1}(u)| + |\mathrm{Out}^{-1}(u)|) \cdot \Delta\right) = O(m \cdot \Delta).$$

Since there are at most $n \cdot |T|$ iterations, the running time of the whole algorithm follows as announced.

Encoding leaves

Let us fix T to be the quasi-polynomial universal tree constructed in Theorem 32.

In our definition of trees we say that a tree is an ordered list of subtrees $[t_1, \ldots, t_k]$, so we use $[1,k]$ with the natural order for ordering the subtrees. Any other total order can be used to that effect, and a more appropriate order may lead to smaller encoding. Indeed, using $[1,k]$ for ordering subtrees, if a tree has height h and n leaves then a leaf is a sequence of h numbers in $[1,n]$, so it uses $O(h \log_2(n))$ bits.

Let us consider an order well suited for encoding T. We use $\{0,1\}^*$ the set of binary words and order them using the following three rules that apply for any $u, v \in \{0,1\}^*$:

$$0u < \varepsilon < 1u \quad ; \quad (0u < 0v \iff u < v) \quad ; \quad (1u < 1v \iff u < v).$$

For words of length at most 2 the order is $00 < 0 < 01 < \varepsilon < 10 < 1 < 11$.

We can now revisit the construction of the universal tree by defining directly the set of leaves. Recall that T is obtained from $T_{\text{left}}, T_{\text{middle}}$, and T_{right}. By induction hypothesis leaves in T_{left} and T_{right} are tuples of length $h-1$ and leaves in T_{middle} tuples of length h. The leaves of T are:

- leaves of T_{left} where the first component is prefixed with a 0;
- leaves of T_{middle} augmented with a new component ε;
- leaves of T_{right} where the first component is prefixed with a 1.

Algorithm 3.5: The value iteration algorithm with explicit data structure.

Data: A parity game with n vertices priorities in $[1,d]$ and a $(n,d/2)$-universal tree T.

Function `Init()`**:**
- **for** $u \in V$ **do**
  - $\mu(u) \leftarrow \ell_{\min}$
- **for** $u \in V_{\text{Eve}}$ **do**
  - **for** $u \xrightarrow{p} v \in E$ **do**
    - **if** *incorrect:* $\neg(\mu(v) \lhd_p \mu(u))$ **then**
      - $\texttt{Count}(u) \leftarrow \texttt{Count}(u)+1$
  - **if** $\texttt{Count}(u) = \texttt{Degree}(u)$ **then**
    - Add u to `Incorrect`
- **for** $u \in V_{\text{Adam}}$ **do**
  - **for** $u \xrightarrow{p} v \in E$ **do**
    - **if** *incorrect:* $\neg(\mu(v) \lhd_p \mu(u))$ **then**
      - Add v to `Incorrect`

Function `Treat(`u`)`**:**
- **if** $u \in V_{\text{Eve}}$ **then**
  - $\mu(u) \leftarrow \min\left\{\delta(\mu(v),p) : u \xrightarrow{p} v \in E\right\}$
- **if** $u \in V_{\text{Adam}}$ **then**
  - $\mu(u) \leftarrow \max\left\{\delta(\mu(v),p) : u \xrightarrow{p} v \in E\right\}$

Function `Update(`u`)`**:**
- **if** $u \in V_{\text{Eve}}$ **then**
  - $\texttt{Count}(u) \leftarrow 0$
- **for** $v \xrightarrow{p} u \in E$ **do**
  - **if** $v \xrightarrow{p} u$ *is incorrect* **then**
    - **if** $v \in V_{\text{Eve}}$ **then**
      - $\texttt{Count}(v) \leftarrow \texttt{Count}(v)+1$
      - **if** $\texttt{Count}(v) = \texttt{Degree}(v)$ **then**
        - Add v to $\texttt{Incorrect}'$
    - **if** $v \in V_{\text{Adam}}$ **then**
      - Add v to $\texttt{Incorrect}'$

Function `Main()`**:**
- `Init` ()
- **for** $i = 0,1,2,\ldots$ **do**
  - $\texttt{Incorrect}' \leftarrow \emptyset$
  - **for** $u \in \texttt{Incorrect}$ **do**
    - `Treat` (u)
    - `Update` (u)
  - **if** $\texttt{Incorrect}' = \emptyset$ **then**
    - **return** μ
  - **else**
    - $\texttt{Incorrect} \leftarrow \texttt{Incorrect}'$

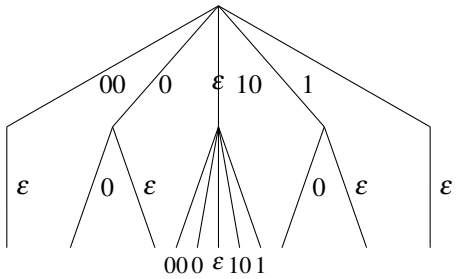


Figure 3.10: The succinct encoding on the $(5,2)$-universal tree.

We call this encoding the *succinct encoding*, it is illustrated in Figure 3.10 for the $(5,2)$-universal tree. The leftmost leaf is $(00,\varepsilon)$, and the middle leaf $(\varepsilon,\varepsilon)$. In general, the inductive construction implies that every leaf is a tuple $(D_{d-1},\ldots,D_1)$ such that the sum of the lengths of the directions D_i is at most $\log_2(n)$. Thus a leaf is encoded using $O(\log_2(h)\log_2(n))$ bits: for each of the $\log_2(n)$ bits we need $\log_2(h)$ bits to specify its component.

In terms of machine words of size $w = \log_2(n)+\log_2(d)$, this means that a leaf can be stored using $\log_2(d)$ machine words. Hence the data structure uses $O(n\log_2(d))$ machine words, with together with the input size $O(m)$ means that the space complexity of the algorithm is $O(m+n\log_2(d))$.

Using the succinct encoding and a tedious but simple case analysis we can compute $\delta(\ell,p)$ in time $O(\log_2(n)\log_2(d))$. Putting everything together we obtain the overall complexity stated in Theorem 31.

3.5 Comparing the three families of algorithms

At the beginning of the chapter we described three families of algorithms: strategy improvement, attractor decomposition, and value iterations.

Let us first clarify the relationship between the separation framework discussed in Section 3.3 and the value iteration paradigm presented in Section 3.4. Both are families of algorithms:

- An (n,d)-separating automaton $\mathbf{A}$ induces an algorithm for solving parity games in time $O(m\cdot|\mathbf{A}|)$ where $|\mathbf{A}|$ is the size of $\mathbf{A}$, meaning the number of states.
- An $(n,d/2)$-universal tree T induces a value iteration algorithm for solving parity games in time proportional to $|T|$ where $|T|$ is the size of T, meaning the number of leaves (the exact complexity depends on the cost of computing δ in T, which is typically small).

These two families are in a strong sense equivalent:

Theorem 35 (Equivalence between separating automata and universal trees)**.**

- *An (n,d)-separating automaton induces an $(n,d/2)$-universal tree of the same size;*
- *An $(n,d/2)$-universal tree induces an (n,d)-separating automaton of the same size.*

We do not prove this theorem here but note that it can be stated more generally for any half-positionally determined objective, replacing universal trees by the notion of universal graphs. The main advantage of the value iteration presentation is the space complexity, which for a good choice of the universal tree can be made very small (quasi-linear).

In terms of complexity, the strategy improvement has exponential complexity, while both attractor decompositions and value iterations algorithms have quasi-polynomial complexity. Let us make a finer comparison: the complexity of the attractor decomposition algorithm is a polynomial multiplied by the (non polynomial) term

$$\binom{d-1+\lfloor \log(n) \rfloor}{\lfloor \log(n) \rfloor},$$

while for the value iteration algorithm the complexity is a polynomial multiplied by the (also non polynomial) term

$$\binom{d/2-1+\lfloor \log(n) \rfloor}{\lfloor \log(n) \rfloor}.$$

The key difference is that the former performs an induction using all priorities, while the latter considers only odd priorities hence the dependence in $d/2$. Although our presentation of the attractor decomposition algorithm does not make it explicit, this class of algorithms is also related to the notion of universal trees; however an algorithm is induced not by one $(n,d/2)$-universal tree, but by two: one for each player, which are then interleaved to organise the recursive calls of the algorithm.

Since both value iteration and attractor decomposition algorithms are connected to the combinatorial notion of universal trees, the next question is whether the construction given in Section 3.4 is optimal. The answer is unfortunately yes, there exists a lower bound on the size of universal trees which matches this construction up to polynomial factors.

The last question we discuss here is whether there exists a quasi-polynomial strategy improvement algorithm. In particular a natural attempt would be to use universal trees for this endeavour. Unfortunately, the natural approach fails, as we explain now. As we have seen, universal trees can be used to represent progress measures for Eve: given a graph, a vertex is assigned $\top$ if all paths satisfy the complement of parity, and some left of the tree otherwise. Hence they can be used for a strategy improvement algorithm iterating over strategies of Adam. Let us consider the exponential universal tree, we present in Figure 3.11 a game where the strategy improvement algorithm alternates between two strategies, hence does not terminate. The minimal progress measures are illustrated on the right hand side, for both strategies. One can verify that in both cases, the missing edge is an improving edge.

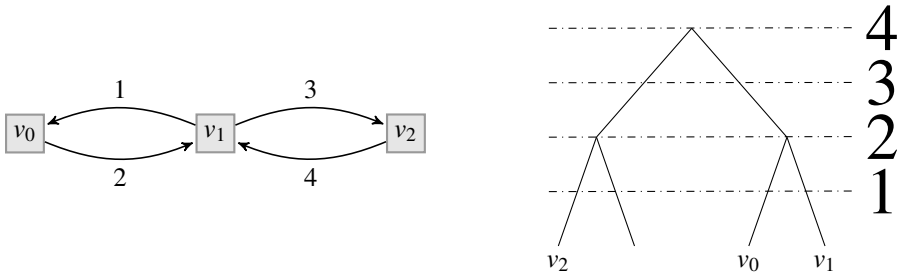


Figure 3.11: A game where the strategy improvement over the exponential universal tree does not terminate.

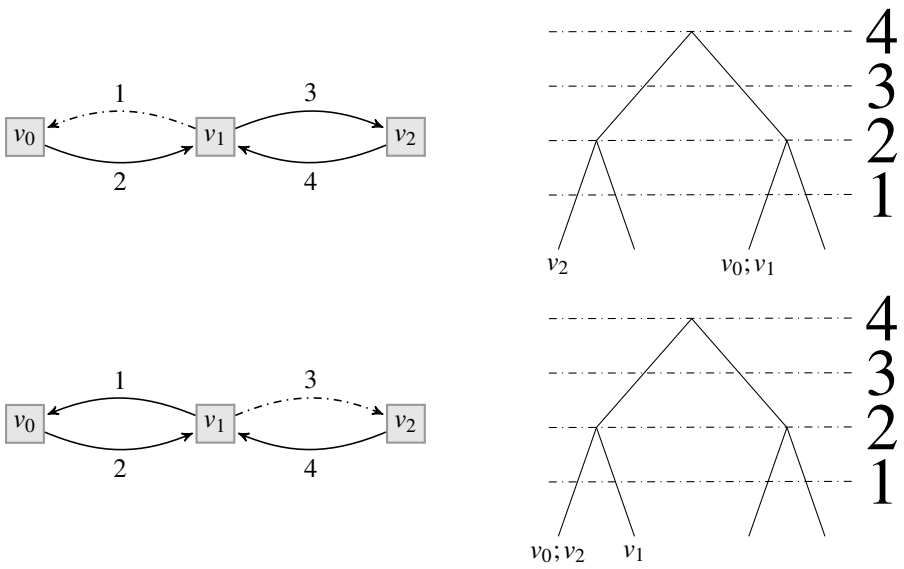


Figure 3.12: The two strategies have opposing improving edges (dashed).

Bibliographic references

We refer to Section 2.6 for the role of parity objectives and how they emerged in automata theory as a subclass of Muller objectives. Another related motivation comes from the works of Emerson, Jutla, and Sistla [EJS93], who showed that solving parity games is linear-time equivalent to the model-checking problem for modal μ-calculus. This logical formalism is an established tool in program verification, and a common denominator to a wide range of modal, temporal and fixpoint logics used in various fields.

Let us discuss the progress obtained over the years for each of the three families of algorithms.

Value iteration algorithms and separating automata. The heart of value iteration algorithms is the value function, which in the context of parity games and related developments for automata have been studied under the name progress measures or signatures. They appear naturally in the context of fixed point computations so it is hard to determine who first introduced them. Streett and Emerson [SE84, SE89] defined signatures for the study of the modal μ-calculus, and Stirling and Walker [SW89] later developed the notion. Both the proofs of Emerson and Jutla [EJ91] and of Walukiewicz [Wal96] use signatures to show the positionality of parity games over infinite games.

Jurdziński [Jur00] used this notion to give the first value iteration algorithm for parity games, with running time $O(mn^{d/2})$. The algorithm is called 'small progress measure lifting' and is an instance of the class of value iteration algorithms we construct in Section 3.4 by considering the universal tree of size n^h. Bernet, Janin, and Walukiewicz [BJW02] investigated the notion of permissive strategies and obtained reductions from parity games to safety games. In essence, they constructed the separating automaton corresponding to the universal tree of size n^h, although the general framework of separating automata came later, introduced by Bojańczyk and Czerwiński [BC18].

The new era for parity games started in 2017 when Calude, Jain, Khoussainov, Li, and Stephan [CJK$^+$17] constructed a quasi-polynomial time algorithm. Our presentation follows the technical developments of the subsequent paper by Fearnley, Jain, Schewe, Stephan, and Wojtczak [FJS$^+$17] which recasts the algorithm as a value iteration algorithm. Bojańczyk and Czerwiński [BC18] introduce the separation framework to better understand the original algorithm.

Soon after two other quasi-polynomial time algorithms emerged. Jurdziński and Lazić [JL17] showed that the small progress measure algorithm can be adapted to a 'succinct progress measure' algorithm, matching (and slightly improving) the quasi-polynomial time complexity. The presentation using universal tree that we follow in Section 3.4 and an almost matching lower bound on their sizes is due to Fijalkow [Fij18]. The connection between separating automata and universal trees was shown by Czerwiński, Daviaud, Fijalkow, Jurdziński, Lazić, and Parys [CDF$^+$19].

The third quasi-polynomial time algorithm is due to Lehtinen [Leh18]. The original algorithm has a slightly worse complexity ($n^{O(\log(n))}$ instead of $n^{O(\log(d))}$), but it was later improved to (essentially) match the complexity of the previous two algo-

rithms [Par20]. Although not explicitly, the algorithm constructs an automaton with similar properties as a separating automaton, but the automaton is non-deterministic. Colcombet and Fijalkow [CF19] revisited the link between separating automata and universal trees and proposed the notion of good for small games automata, capturing the automaton defined by Lehtinen's algorithm. The equivalence result between separating automata, good for small games automata, and universal graphs, holds for any half-positionally determined objective, giving a strong theoretical foundation for the family of value iteration algorithms.

Attractor decomposition algorithms. The McNaughton Zielonka's algorithm has complexity $O(mn^d)$. Parys [Par19] constructed the fourth quasi-polynomial time algorithm as an improved take over McNaughton Zielonka's algorithm. As for Lehtinen's algorithm, the original algorithm has a slightly worse complexity ($n^{O(\log(n))}$ instead of $n^{O(\log(d))}$). The construction was later improved [LPSW22]. As discussed in Section 3.5 the complexity of this algorithm is quasi-polynomial and of the form $n^{O(\log(d))}$, but a bit worse than the three previous algorithms since the algorithm is symmetric and has a recursion depth of d, while the value iteration algorithms only consider odd priorities hence replace d by $d/2$.

Jurdziński, Morvan, and Thejaswini [JMT22] constructed a generic McNaughton Zielonka's algorithm parameterised by the choice of two universal trees, one for each player. How the value iteration algorithm can simulate the attractor decomposition algorithm was explained by Ohlmann [Ohl21].

Strategy improvement algorithms. As we will see in Chapter 4, parity games can be reduced to mean payoff games or energy games, as well as discounted payoff games, so any algorithm for solving them can be used for solving parity games. In particular, the existing strategy improvement algorithms for these games can be run on parity games. Vöge and Jurdziński [VJ00] introduced the first discrete strategy improvement for parity games, running in exponential time. The algorithm we present is due to Björklund, Sandberg, and Vorobyov [BSV03], which was shown to run in sub-exponential time using a randomisation procedure. Our proof of correctness is original. The complexity was reduced to sub-exponential with deterministic algorithms by Jurdziński, Paterson, and Zwick [JPZ08]. For some time there was some hope that the strategy improvement algorithm, for some well chosen policy on switching edges, solves parity games in polynomial time. Friedmann [Fri11] cast some serious doubts by constructing numerous exponential lower bounds applying to different variants of the algorithm. Fearnley [Fea17] investigated efficient implementations of the algorithm, focussing on the cost of computing and updating the value function for a given strategy.

A natural question is whether there exists a quasi-polynomial strategy improvement algorithm. Koh and Loho [KL22] constructed a quasi-polynomial time algorithm and presented it as a strategy improvement algorithm: the subtlety is that in their algorithm, an iteration considers a strategy σ together with its valuation val^σ and builds a new strategy σ' with the additional constraint that $\mathrm{val}^\sigma < \mathrm{val}^{\sigma'}$. In other words, it does not improve only based on the strategy σ. Hence in some sense it is closer to a value iteration algorithm. As discussed in Section 3.5 the notion of universal trees cannot be used to achieve this: the example is due to Ohlmann [Ohl21]. Whether there exists

a quasi-polynomial time strategy improvement algorithm for parity games remains to this day open.

GAMES WITH PAYOFFS

Chapter 4

Games with Payoffs

NATHANAËL FIJALKOW, BENJAMIN MONMEGE

This chapter considers quantitative objectives defined using payoffs. Adding quantities can serve two goals: the first is for refining qualitative objectives by quantifying how well, how fast, or at what cost a qualitative objective is satisfied, and the second is to define richer specifications and preferences over outcomes.

- We start in Section 4.2 by studying extensions of the classical qualitative objectives. Among two strategies in a reachability game that guarantee to reach a target in ten steps or in a billion steps, we would certainly prefer the first one from a pragmatic point of view.

- We continue in Section 4.1 by introducing two approaches for proving positional determinacy, that imply the results we need in this chapter.

- We study *mean payoff games* in Section 4.3, and *energy games* along the way. We present three algorithms for solving them, the first two based on *value iteration* and the third on *strategy improvement*. Along the way we show that parity games reduce to mean payoff games.

- We study *discounted payoff games* in Section 4.4. We construct a strategy improvement algorithm for computing the value function. We also show that mean payoff games reduce to discounted payoff games, so the previous algorithm yields an algorithm for computing the value function of a mean payoff game.

- We study *shortest path games* in Section 4.5. They extend reachability games by requiring that Min reaches her target with minimal cost, which if the weights are all equal means *as soon as possible*.

- We study *total payoff games* in Section 4.6.

Notations

In this chapter all objectives we consider use the set of colours $C = \mathbb{Z}$ the set of integers (or $C = \mathbb{Z} \cup \{\text{Win}\}$ for the shortest path objective), so a colour is called a *weight* interpreted as a payoff for Max. We will study different quantitative objectives corresponding to different ways of aggregating the weights. We let $\text{val}^{\mathscr{G}}_{\text{Max}}(v)$ and $\text{val}^{\mathscr{G}}_{\text{Min}}(v)$ be the values for Max and Min in the game $\mathscr{G}$: this is the best each player can unilaterally guarantee from the vertex v, no matter which strategy the other player uses. All the quantitative objectives we will study in this chapter are Borel, hence determined thanks to Corollary 1: $\text{val}^{\mathscr{G}}_{\text{Max}}(v) = \text{val}^{\mathscr{G}}_{\text{Min}}(v)$ for all vertices v. We thus let $\text{val}^{\mathscr{G}}(v)$ denote this value (and $\text{val}(v)$ if $\mathscr{G}$ is clear from the context).

For complexity statements we use the unit cost RAM model as defined in Section 1.2. Let us make some preliminary remarks, and consider a game $\mathscr{G}$ with an objective using the set of colours $C = \mathbb{Z}$. We let W denote the largest weight appearing in $\mathscr{G}$ in absolute value. Choosing the machine word size $w = \log(m) + \log(W)$ implies that an edge together with its weight can be stored in one machine word and that we can perform arithmetic operations on weights, hence for most objectives we use the machine word size $w = \log(m) + \log(W)$. The only exception will be the discounted payoff objectives, which additionally have a discount factor $\lambda \in (0,1)$ that needs to be given in the input.

4.1 Proving positional determinacy over finite arenas

In this section we present two approaches for proving positional determinacy of quantitative objectives over finite arenas.

Positional determinacy via first cycle games

Theorem 36 (First cycle games)**.** *Let $\Phi : C^\omega \to \mathsf{R} \cup \{\pm\infty\}$ a quantitative objective, and $f : C^* \to \mathsf{R} \cup \{\pm\infty\}$. We assume that the following properties are satisfied. For all games $\mathscr{G}$ with objective Φ:*

- *for all $x \in \mathsf{R} \cup \{\pm\infty\}$, for all plays π,*

$$(\forall c \in \text{Cycles}(\pi),\ f(c) \geq x) \implies \Phi(\pi) \geq x,$$

- *for all $x \in \mathsf{R} \cup \{\pm\infty\}$, for all plays π,*

$$(\forall c \in \text{Cycles}(\pi),\ f(c) \leq x) \implies \Phi(\pi) \leq x,$$

- *Φ is prefix independent.*

Then Φ is uniformly positionally determined over finite arenas.

To build some intuition, let us consider the case of mean payoff: let $\Phi = \texttt{MeanPayoff}^+$. Then the properties above hold when considering for f the function Mean:

$$\text{Mean}(w_0 \dots w_{k-1}) = \frac{1}{k} \cdot \sum_{i=0}^{k-1} w_i.$$

Let us argue that the first property holds. To estimate $\texttt{MeanPayoff}^+(\pi)$ let us look at the prefix $\pi_{<k}$ of π of length k. Indeed $\texttt{MeanPayoff}^+(\pi) = \limsup_k \text{Mean}(\pi_{<k})$. To calculate $\text{Mean}(\pi_{<k})$ we use the fact that every vertex in $\pi_{<k}$ either belongs to exactly one cycle in $\text{Cycles}(\pi_{<k})$ or to $\widehat{\pi_{<k}}$, and the linearity of the arithmetic mean:

$$\text{Mean}(\pi_{<k}) = \text{Mean}\left\{\text{Mean}(\widehat{\pi_{<k}}), \{\text{Mean}(c) : c \in \text{Cycles}(\pi_{<k})\}\right\}.$$

By assumption each $\text{Mean}(c)$ is greater than or equal to x. Note that since $\widehat{\pi_{<k}}$ does not contain any cycle, it has length at most n, independently of k. On the other hand $\text{Cycles}(\pi_{<k})$ has size at least k/n. Hence when k tends to infinity the term $\text{Mean}(\widehat{\pi_{<k}})$ vanishes and we have $\texttt{MeanPayoff}^+(\pi) = \limsup_k \text{Mean}(\pi_{<k}) \geq x$.

The second property follows as above from the linearity of the arithmetic mean and the cycle decomposition. The third property will be proved in Lemma 32.

Corollary 6 (Positional determinacy of mean payoff)**.** *Mean payoff objectives are uniformly positionally determined over finite arenas.*

By duality this is equivalent to saying that both limit superior and limit inferior mean payoff objectives are uniformly positionally determined over finite arenas.

Proof. Let $\mathscr{A}$ an arena. A cycle is a sequence $c = v_0 \xrightarrow{w_0} v_1 \xrightarrow{w_1} v_2 \dots v_{k-1} \xrightarrow{w_{k-1}} v_0$. For technical convenience we do not indicate the returning vertex v_0 when introducing a cycle.

Let us consider a finite play $\pi = v_0 \xrightarrow{w_0} v_1 \xrightarrow{w_1} \dots$, we define two objects by induction: the cycle decomposition $\text{Cycles}(\pi)$ and the folded play $\widehat{\pi}$. To guide the intuition: $\text{Cycles}(\pi)$ is a list of *some* cycles in π, and $\widehat{\pi}$ is obtained from π by removing the cycles from $\text{Cycles}(\pi)$ and does not contain any cycle.

If π is a single vertex then the cycle decomposition is empty and the folded play is equal to π. Otherwise let $\pi = \pi' \xrightarrow{w} v$, by induction we have already defined $\text{Cycles}(\pi')$ and $\widehat{\pi'}$. There are two cases:

- Either v appears in $\widehat{\pi'}$ and then $\widehat{\pi}$ is obtained from $\widehat{\pi'} \xrightarrow{w} v$ by replacing that cycle by v and $\text{Cycles}(\pi)$ by adding the cycle to $\text{Cycles}(\pi')$.
- Or v does not appear in $\widehat{\pi'}$ and then $\widehat{\pi} = \widehat{\pi'} \xrightarrow{w} v$ and $\text{Cycles}(\pi) = \text{Cycles}(\pi')$.

The cycle decomposition breaks down π into (possibly interleaved) cycles and the folded play: every vertex in π either belongs to exactly one cycle in $\text{Cycles}(\pi)$ or to $\widehat{\pi}$. For instance for

$$\pi = v_0 \xrightarrow{w_0} v_1 \xrightarrow{w_1} v_2 \xrightarrow{w_2} v_3 \xrightarrow{w_3} v_2 \xrightarrow{w_4} v_4 \xrightarrow{w_5} v_1 \xrightarrow{w_6} v_5$$

we have $\text{Cycles}(\pi) = (c_1 = v_2 \xrightarrow{w_2} v_3 \xrightarrow{w_3}; c_2 = v_1 \xrightarrow{w_1} v_2 \xrightarrow{w_4} v_4 \xrightarrow{w_5})$ and $\widehat{\pi} = v_0 \xrightarrow{w_0} v_1 \xrightarrow{w_6} v_5$, as illustrated in Figure 4.1.

The definition of $\text{Cycles}(\pi)$ is extended for infinite plays. We write $\text{FC}(\pi)$ for the first cycle in π.

Let $\mathscr{G} = (\mathscr{A}, \Phi(\mathfrak{c}))$. The outline of the proof is as follows.

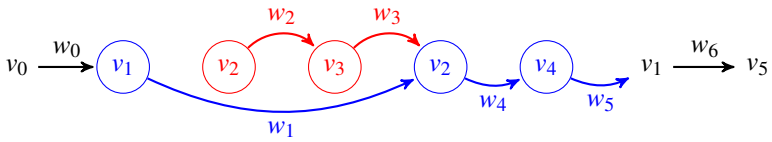


Figure 4.1: An example for the cycle decomposition. The first cycle is c_1 and is in red, the cycle c_2 is in blue, and $\widehat{\pi} = v_0 \xrightarrow{w_0} v_1 \xrightarrow{w_6} v_5$.

1. We define the condition FirstCycle and $\mathcal{G}_{\text{FC}} = (\mathcal{A}, \text{FirstCycle})$, and show that $\text{val}^{\mathcal{G}} = \text{val}^{\mathcal{G}_{\text{FC}}}$ and that if $\mathcal{G}_{\text{FC}}$ is positionally determined then $\mathcal{G}$ is positionally determined.
2. We define the condition FirstCycleReset_v (parameterised by a vertex v) and $\mathcal{G}_{\text{FCR}(v)} = (\mathcal{A}, \text{FirstCycleReset}_v)$, and show that $\text{val}^{\mathcal{G}} = \text{val}^{\mathcal{G}_{\text{FCR}(v)}}$.
3. We show that $\mathcal{G}_{\text{FC}}$ is positionally determined.

Step 1. The condition FirstCycle computes the value of f on the first cycle. Formally:

$$\text{FirstCycle}(\pi) = f(\text{FC}(\pi)).$$

Let us define $\mathcal{G}_{\text{FC}} = (\mathcal{A}, \text{FirstCycle})$. We make two claims:

- let σ_{FC} a strategy in $\mathcal{G}_{\text{FC}}$, it induces a strategy σ in $\mathcal{G}$ such that $\text{val}^{\sigma_{\text{FC}}} \leq \text{val}^{\sigma}$, and σ is positional if σ_{FC} is, and
- let τ_{FC} a strategy in $\mathcal{G}_{\text{FC}}$, it induces a strategy τ in $\mathcal{G}$ such that $\text{val}^{\tau_{\text{FC}}} \geq \text{val}^{\tau}$, and τ is positional if τ_{FC} is.

Let us prove the first claim. We let $\sigma(\pi) = \sigma_{\text{FC}}(\widehat{\pi})$: by definition if π is a play consistent with σ then $\widehat{\pi}$ is a play consistent with σ_{FC}. Note that indeed if σ_{FC} is positional then so is σ. We show that $\text{val}^{\sigma_{\text{FC}}} \leq \text{val}^{\sigma}$.

Let π be a play consistent with σ from v_0 and $x = \text{val}^{\sigma_{\text{FC}}}(v_0)$. We first argue that all cycles in $\text{Cycles}(\pi)$ have value for f greater than or equal to x. Indeed consider a cycle c in $\text{Cycles}(\pi)$ and let π' the prefix of π ending with the cycle c. The play $\widehat{\pi'}$ is consistent with σ_{FC}, implying that $f(c) \geq x$. Thanks to the first property, this implies that $\Phi(\pi) \geq x$.

We turn to the second claim. The construction is identical for Min: we let $\tau(\pi) = \tau_{\text{FC}}(\widehat{\pi})$. We show that $\text{val}^{\tau_{\text{FC}}} \geq \text{val}^{\tau}$ following the same arguments. Let π be a play consistent with τ from v_0 and $x = \text{val}^{\tau_{\text{FC}}}(v_0)$. Thanks to the second property, this implies that $\Phi(\pi) \leq x$.

Let us now prove that $\text{val}^{\mathcal{G}} = \text{val}^{\mathcal{G}_{\text{FC}}}$ using the two claims. We first need to establish that $\mathcal{G}_{\text{FC}}$ is determined: one argument is that FirstCycle is a Borel condition, hence determinacy follows from the general Borel determinacy result (Theorem 1). Another

simpler argument is that FirstCycle is a finite duration condition, meaning that the outcome of the play is determined with a finite number of steps (at most n), hence determinacy follows from an easier direct argument for finite duration games.

It follows from the two claims that

$$\begin{array}{llll} \mathrm{val}^{\mathscr{G}_{\mathrm{FC}}} & = \sup_{\sigma_{\mathrm{FC}}} \mathrm{val}^{\sigma_{\mathrm{FC}}} & \leq \sup_{\sigma} \mathrm{val}^{\sigma} & = \mathrm{val}^{\mathscr{G}} \\ \mathrm{val}^{\mathscr{G}_{\mathrm{FC}}} & = \inf_{\tau_{\mathrm{FC}}} \mathrm{val}^{\tau_{\mathrm{FC}}} & \geq \inf_{\tau} \mathrm{val}^{\tau} & = \mathrm{val}^{\mathscr{G}}, \end{array}$$

where in both lines: the first equalities is by determinacy of $\mathscr{G}_{\mathrm{FC}}$, the inequalities thanks to the two claims, and the last equalities by determinacy of $\mathscr{G}$. The two obtained inequalities imply that $\mathrm{val}^{\mathscr{G}} = \mathrm{val}^{\mathscr{G}_{\mathrm{FC}}}$, and positional optimal strategies for either player in $\mathscr{G}_{\mathrm{FC}}$ induce positional optimal strategies in $\mathscr{G}$.

Step 2. We define a third condition on $\mathscr{A}$:

$$\mathrm{FirstCycleReset}_v(\pi) = \begin{cases} \mathrm{FirstCycle}(\pi) & \text{if } \pi \text{ does not visit } v \text{ before } \mathrm{FC}(\pi), \\ \mathrm{FirstCycle}(\pi_{\geq k}) & \text{for } k \text{ the first index such that } \mathrm{In}(\pi_k) = v. \end{cases}$$

In words: if a cycle is closed before visiting v, then the condition is FirstCycle and otherwise when reaching v the game is 'reset' and the condition is FirstCycle from this point onwards.

This second step is similar to the first step; the reason why we separated them is because this step relies on the fact that Φ is prefix independent.

Let us define $\mathscr{G}_{\mathrm{FCR}(v)} = (\mathscr{A}, \mathrm{FirstCycleReset}_v)$. We make two claims:

- let $\sigma_{\mathrm{FCR}(v)}$ an optimal strategy in $\mathscr{G}_{\mathrm{FCR}(v)}$, it induces a strategy σ in $\mathscr{G}$ such that $\mathrm{val}^{\sigma_{\mathrm{FCR}(v)}} \leq \mathrm{val}^{\sigma}$, and
- let $\tau_{\mathrm{FCR}(v)}$ an optimal strategy in $\mathscr{G}_{\mathrm{FCR}(v)}$, it induces a strategy τ in $\mathscr{G}$ such that $\mathrm{val}^{\tau_{\mathrm{FCR}(v)}} \geq \mathrm{val}^{\tau}$.

Let us prove the first claim. We let

$$\sigma(\pi) = \begin{cases} \sigma_{\mathrm{FCR}(v)}(\widehat{\pi}) & \text{if } \pi \text{ does not contain } v, \\ \sigma_{\mathrm{FCR}(v)}(\widehat{\pi_{\geq k}}) & \text{for } k \text{ the first index such that } \mathrm{In}(\pi_k) = v. \end{cases}$$

We show that $\mathrm{val}^{\sigma_{\mathrm{FCR}(v)}} \leq \mathrm{val}^{\sigma}$. Let π a play consistent with σ from v_0 and $x = \mathrm{val}^{\sigma_{\mathrm{FCR}(v)}}(v_0)$. There are two cases.

Either π does not contain v, and then as in the first step this implies that all cycles in π have an arithmetic mean greater than or equal to x, and we conclude as in the first step that $\mathrm{FirstCycleReset}_v(\pi) \geq x$.

Or π contains v. We first argue that $x = \mathrm{val}^{\sigma_{\mathrm{FCR}(v)}}(v_0) \leq \mathrm{val}^{\sigma_{\mathrm{FCR}(v)}}(v)$. Indeed, let π_0 be a play without cycles from v_0 to v consistent with $\sigma_{\mathrm{FCR}(v)}$, we let $\sigma'(\pi) = \sigma_{\mathrm{FCR}(v)}(\pi_0\pi)$. Then $\mathrm{val}^{\sigma_{\mathrm{FCR}(v)}}(v_0) \leq \mathrm{val}^{\sigma'}(v) \leq \mathrm{val}^{\sigma_{\mathrm{FCR}(v)}}(v)$, the first inequality is because a play π consistent with σ' from v correspond to the play $\pi_0\pi$ consistent with $\sigma_{\mathrm{FCR}(v)}$ from v_0, and the second inequality by optimality of $\sigma_{\mathrm{FCR}(v)}$.

Let $y = \mathrm{val}^{\sigma_{\mathrm{FCR}(v)}}(v)$, the inequality above reads $x \leq y$. Let π' the suffix of π starting from the first occurrence of v, then π' is consistent with $\sigma_{\mathrm{FCR}(v)}$ from v, so as in the first

step this implies that all cycles in π' have an arithmetic mean greater than or equal to y. We conclude as in the first step that $\Phi(\pi') \geq y \geq x$, The last but important argument is that Φ is prefix independent, implying that $\Phi(\pi) \geq x$.

We prove that $\mathrm{val}^{\mathscr{G}} = \mathrm{val}^{\mathscr{G}_{\mathrm{FCR}(v)}}$ using the two claims following the same arguments as in the first step. In particular we need to establish that $\mathscr{G}_{\mathrm{FCR}(v)}$ is determined, and again it either follows from the general Borel determinacy result (Theorem 1) or by determinacy for finite duration games.

Step 3. We (finally!) prove that $\mathscr{G}_{\mathrm{FC}}$ is positionally determined. Let us consider the case of Max, the case of Min being symmetric. We show that for all games there exists an optimal positional strategy, by induction on the number of vertices of Max with more than one outgoing edge. The base case is clear since in that case there is a unique strategy and it is positional. Let $v \in V_{\mathrm{Max}}$ with more than one outgoing edge.

Let $\sigma_{\mathrm{FCR}(v)}$ an optimal strategy in $\mathscr{G}_{\mathrm{FCR}(v)}$. Intuitively, since the game is reset at the first visit of v and that the second visit to v would close a loop hence determine the outcome, we can use any optimal strategy from v. We define σ' as follows:

$$\sigma'(\pi) = \begin{cases} \sigma(\pi) & \text{if } \pi \text{ does not contain } v, \\ \sigma(\pi_{\geq k}) & \text{for } k \text{ the last index such that } \mathrm{In}(\pi_k) = v. \end{cases}$$

Note that indeed σ' uses only one outgoing edge of v. We show that $\mathrm{val}^{\mathscr{G}_{\mathrm{FCR}(v)},\sigma}(v_0) \leq \mathrm{val}^{\mathscr{G}_{\mathrm{FCR}(v)},\sigma'}(v_0)$, implying that σ' is optimal in $\mathscr{G}_{\mathrm{FCR}(v)}$ and the inequality is an equality.

Let π be a play consistent with σ' from v_0. If it does not contain v it is consistent with σ, so $\mathrm{FirstCycleReset}_v(\pi) \geq \mathrm{val}^{\mathscr{G}_{\mathrm{FCR}(v)},\sigma}(v_0)$. If it contains v, let π' the suffix of π starting from the first occurrence of v, then $\mathrm{FirstCycleReset}_v(\pi) = \mathrm{FirstCycle}(\pi')$. Since π' is consistent with σ (until a cycle is formed) we have $\mathrm{FirstCycle}(\pi') \geq \mathrm{val}^{\mathscr{G}_{\mathrm{FCR}(v)},\sigma}(v_0)$, implying that $\mathrm{FirstCycleReset}_v(\pi) \geq \mathrm{val}^{\mathscr{G}_{\mathrm{FCR}(v)},\sigma}(v_0)$. We conclude: $\mathrm{val}^{\mathscr{G}_{\mathrm{FCR}(v)},\sigma'}(v_0) \leq \mathrm{val}^{\mathscr{G}_{\mathrm{FCR}(v)},\sigma}(v_0)$.

Let $\mathscr{B}$ the arena obtained from $\mathscr{A}$ by removing all outgoing edges of v not used by σ'. Let $\mathscr{G}'_{\mathrm{FCR}(v)} = (\mathscr{B}, \mathrm{FirstCycleReset}_v)$ and $\mathscr{G}'_{\mathrm{FC}} = (\mathscr{B}, \mathrm{FirstCycle})$. By definition we have $\mathrm{val}^{\mathscr{G}_{\mathrm{FCR}(v)},\sigma'} = \mathrm{val}^{\mathscr{G}'_{\mathrm{FCR}(v)},\sigma'} = \mathrm{val}^{\mathscr{G}'_{\mathrm{FCR}(v)}}$. In the previous step we proved that $\mathrm{val}^{\mathscr{G}'_{\mathrm{FC}}} = \mathrm{val}^{\mathscr{G}'_{\mathrm{FCR}(v)}}$, so $\mathrm{val}^{\mathscr{G}'_{\mathrm{FC}}} = \mathrm{val}^{\mathscr{G}_{\mathrm{FC}}}$. The arena $\mathscr{B}$ contains one less vertex of Max with more than one outgoing edge, so the induction hypothesis applies and implies that there exists an optimal strategy in $\mathscr{G}'_{\mathrm{FC}}$, which is also optimal in $\mathscr{G}_{\mathrm{FC}}$ thanks to the equality $\mathrm{val}^{\mathscr{G}'_{\mathrm{FC}}} = \mathrm{val}^{\mathscr{G}_{\mathrm{FC}}}$. □

Positional determinacy via fairly mixing property

Definition 7 (Fairly mixing)**.** *A quantitative objective* $\Phi : C^\omega \to \mathsf{R} \cup \{\pm\infty\}$ *is* fairly mixing *if:*

1. *for all* $x \in C^+$, $y_0, y_1 \in C^\omega$, *if* $\Phi(y_0) \leq \Phi(y_1)$ *then* $\Phi(xy_0) \leq \Phi(xy_1)$*;*
2. *for all* $x \in C^+$, $y \in C^\omega$, $\min(\Phi(x^\omega), \Phi(y)) \leq \Phi(xy) \leq \max(\Phi(x^\omega), \Phi(y))$*;*

3. *if $(x_i)_{i\in\mathbb{N}}$ is a sequence of non-empty words $x_i \in C^+$ such that $x_0x_1x_2\cdots$ contains only a finite number of colours, and $I \uplus J = \mathbb{N}$ is a partition of $\mathbb{N}$ into two infinite sets, then*
$$\inf(\mathscr{P}_I \cup \mathscr{P}_J) \leq \Phi(x_0x_1x_2\cdots) \leq \sup(\mathscr{P}_I \cup \mathscr{P}_J)$$
with $\mathscr{P}_I = \{\Phi(x_{i_0}x_{i_1}x_{i_2}\cdots) \mid i_k \in I\}$ and $\mathscr{P}_J = \{\Phi(x_{j_0}x_{j_1}x_{j_2}\cdots) \mid j_k \in J\}$.

It is not difficult to convince oneself that the following quantitative objectives: `Inf`, `Sup`, `LimInf`, `LimSup`, `MeanPayoff`$^+$, `MeanPayoff`$^-$, `Energy`, `DiscountedPayoff`, `TotalPayoff` are all are fairly mixing.

Theorem 37 (Fairly mixing induces positional determinacy)**.** *If Φ is a fairly mixing quantitative objective, then Φ is uniformly positionally determined over finite arenas.*

The (rather technical) proof is by induction on the number of vertices.

Corollary 7. *The quantative objectives* `Inf`*,* `Sup`*,* `LimInf`*,* `LimSup`*,* `MeanPayoff`$^+$*,* `MeanPayoff`$^-$*,* `Energy`*,* `DiscountedPayoff`*,* `TotalPayoff` *are all uniformly positionally determined over finite arenas.*

4.2 Refining qualitative objectives with quantities

In this section we define quantitative objectives extending the qualitative objectives `Safe`, `Reach`, `Buchi`, and `CoBuchi`. The four quantitative objectives we will define in this section return as outcome some weight in the sequence (for instance, the maximum weight). This is in contrast with the `MeanPayoff` and `DiscountedPayoff` objectives that we will study later, which perform *arithmetic operations* on the sequence of weights.

A first way to compute a payoff from a sequence of weights $\rho \in \mathbb{Z}^\omega$ is to consider the maximum weight in the sequence:

$$\texttt{Sup}(\rho) = \sup_i \rho_i.$$

This extends the qualitative objective `Reach`(Win) in the following sense: the objective `Reach`(Win) corresponds to the quantitative objective `Sup` using two weights: 0 for Lose and 1 for Win. The outcome of a sequence is 1 if and only if the sequence contains Win. It refines `Reach`(Win) by specifying (numerical) preferences.

The dual objective is to consider the smallest weight:

$$\texttt{Inf}(\rho) = \inf_i \rho_i.$$

The qualitative objective `Safe`(Win) corresponds to the quantitative objective `Inf` using two weights: 0 for Win and 1 for Lose. The outcome of a sequence is 0 if and only if the sequence does not contain Lose.

Similarly the following quantitative objectives refine `Buchi` and `CoBuchi`:

$$\texttt{LimSup}(\rho) = \limsup_i \rho_i, \qquad \texttt{LimInf}(\rho) = \liminf_i \rho_i.$$

The analyses and algorithms for solving games with `Reach`, `Safe`, `Buchi`, and `CoBuchi` objectives extend to these four quantitative objectives.

Theorem 38 (Sup, Inf, LimSup, LimInf objectives)**.** *Games with objectives* `Sup`*,* `Inf`*,* `LimSup`*, and* `LimInf` *are uniformly positionally determined. There exists an algorithm for computing the value function of those games in polynomial time and space. More precisely, let k be the number of different weights in the game, the time complexity is $O(m)$ for objectives* `Sup` *and* `Inf`*, and $O(knm)$ for objectives* `LimSup` *and* `LimInf`*, and for all algorithms the space complexity is $O(m)$.*

Proof. We sketch the algorithm for the objective `Sup`, the other cases are similar. Let $c_1 < \cdots < c_k$ be the ordered enumeration of all weights in the game. The set of vertices of value c_k is $\mathrm{Attr}_{\mathrm{Max}}(c_k)$, which can be computed in linear time. We then construct the subgame $\mathscr{G}'$ of $\mathscr{G}$ induced by $V \setminus \mathrm{Attr}_{\mathrm{Max}}(c_k)$, and continue recursively: $\mathscr{G}'$ has one less weight.

A naive complexity analysis yields a time complexity $O(km)$, but it is easily refined to $O(m)$ by revisiting the attractor computation and showing that each edge in the whole game is treated at most once throughout the recursive attractor computations. This complexity analysis does not extend to `LimSup` and `LimInf` objectives, where the complexity is multiplied by k. □

4.3 Mean payoff games

A natural approach for aggregating an infinite sequence of weights is to consider the arithmetic mean. Since the sequence $(\frac{1}{k}\sum_{i=0}^{k-1}\rho_i)_{k\in\mathbb{N}}$ may not converge, we can either consider the limit superior or the limit inferior, leading to the two definitions:

$$\texttt{MeanPayoff}^{+}(\rho) = \limsup_k \frac{1}{k}\sum_{i=0}^{k-1}\rho_i \quad ; \quad \texttt{MeanPayoff}^{-}(\rho) = \liminf_k \frac{1}{k}\sum_{i=0}^{k-1}\rho_i.$$

Note that $\texttt{MeanPayoff}^{+}(-\rho) = -\texttt{MeanPayoff}^{-}(\rho)$, where in $-\rho$ we take the opposite of each weight. In other words, $\texttt{MeanPayoff}^{+}$ and $\texttt{MeanPayoff}^{-}$ are dual objectives.

In this section we study mean payoff games. We will first prove that they are prefix independent, then that they are positionally determined, and then construct algorithms for solving them and compute the value function. The best known time complexity for both problems is pseudo-polynomial, meaning polynomial when the numerical inputs are given in unary.

Lemma 32 (Prefix independence)**.** $\texttt{MeanPayoff}^{+}$ *is prefix independent.*

Proof. We show that $\texttt{MeanPayoff}^{+}(\rho_0\rho_1\dots) = \texttt{MeanPayoff}^{+}(\rho_p\rho_{p+1}\dots)$ for a fixed

$p \in \mathbb{N}$:

$$\begin{aligned}\limsup_k \frac{1}{k}\sum_{i=0}^{k-1}\rho_i &= \limsup_k \left(\underbrace{\frac{1}{k}\sum_{i=0}^{p-1}\rho_i}_{\to 0 \text{ for } k\to\infty} + \underbrace{\frac{k-p}{k}}_{\to 1 \text{ for } k\to\infty} \cdot \frac{1}{k-p}\sum_{i=p}^{k-1}\rho_i \right) \\ &= \liminf_k \frac{1}{k-p}\sum_{i=0}^{p-1}\rho_{p+i}.\end{aligned}$$

□

Note that by duality this implies that $\texttt{MeanPayoff}^-$ is also prefix independent.

In the setting we consider in this chapter, meaning two player zero sum games of perfect information, the two objectives are equivalent, which will be a corollary of Corollary 6.

As an example, consider the mean payoff game represented in Figure 4.2. As we will show later, the positional strategies defined below are optimal strategies for Max and Min:

$$\begin{gathered}\sigma^*(v_0) = v_0 \xrightarrow{4} v_1 \quad ; \quad \sigma^*(v_2) = v_2 \xrightarrow{4} v_3 \\ \tau^*(v_1) = v_1 \xrightarrow{2} v_2 \quad ; \quad \tau^*(v_3) = v_3 \xrightarrow{-1} v_1 \quad ; \quad \tau^*(v_4) = v_4 \xrightarrow{-2} v_0.\end{gathered}$$

The play consistent with σ^* and τ^* starting from v_0 is the lasso word

$$v_0 \xrightarrow{4} v_1 \left(v_1 \xrightarrow{2} v_2 \xrightarrow{4} v_3 \xrightarrow{-1} v_1\right)^\omega,$$

which has as mean payoff the average weight of the cycle $v_1 \xrightarrow{2} v_2 \xrightarrow{4} v_3 \xrightarrow{-1} v_1$, i.e. $5/3$. If we restrict ourselves to memoryless strategies for both players, we may easily convince ourselves that this is the best that both players may aim for. Indeed,

- the loop around v_4 has average weight 2;
- the simple cycle alternating between v_0 and v_4 has average weight 1.5;
- the simple cycle alternating between v_0 and v_1 has average weight 2;
- the loop around v_2 has average weight 1, and
- the simple cycle alternating between v_0, v_1, v_2 and v_3 has average weight 2.

Therefore, we may check that no player can obtain a better mean payoff than $5/3$, from its own point of view: for instance, if Max switches her decision in v_0, hoping to get value 2, she will get a lower value 1.5 since Min still prefers cycle alternating between v_0 and v_4 to his self-loop around v_4. This is the case for all starting vertices, which proves that all vertices have the same value $5/3$.

Let us draw some corollaries from positional determinacy of mean payoff games.

Corollary 8 (Limit superior and limit inferior mean payoff games)**.**

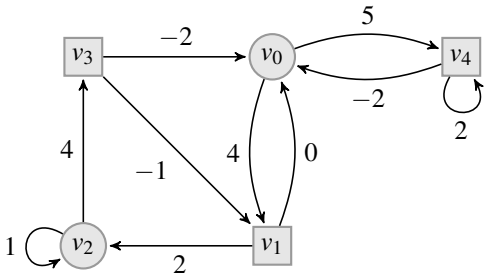


Figure 4.2: A mean payoff game.

- *Limit superior and limit inferior mean payoff games are equivalent: for every arena $\mathscr{A}$ and colouring function $\mathfrak{c} : E \to \mathbb{Z}$, let $\mathscr{G}_+ = (\mathscr{A}, \texttt{MeanPayoff}^+(\mathfrak{c}))$ and $\mathscr{G}_- = (\mathscr{A}, \texttt{MeanPayoff}^-(\mathfrak{c}))$, then $\text{val}^{\mathscr{G}_+} = \text{val}^{\mathscr{G}_-}$. This implies that a positional strategy is optimal in $\mathscr{G}_+$ if and only if it is optimal in $\mathscr{G}_-$.*

  Since they are equivalent we speak of mean payoff games without specifying whether the objective is $\texttt{MeanPayoff}^+$ *or* $\texttt{MeanPayoff}^-$*, and write* MeanPayoff *instead.*

- *For a mean payoff game with n vertices and weights in $[-W, W]$, the mean payoff values are rational numbers in $[-W, W]$ whose denominators are at most n.*

Proof. Thanks to Corollary 6, there exist σ_+ and τ_+ optimal positional strategies in $\mathscr{G}_+$ and σ_- and τ_- optimal positional strategies in $\mathscr{G}_-$ (for the latter by duality). By definition $\text{val}^{\mathscr{G}_+}(v) = \texttt{MeanPayoff}^+(\pi^v_{\sigma_+,\tau_+})$ and $\text{val}^{\mathscr{G}_-}(v) = \texttt{MeanPayoff}^-(\pi^v_{\sigma_-,\tau_-})$. Since $\texttt{MeanPayoff}^- \leq \texttt{MeanPayoff}^+$ we already have $\text{val}^{\mathscr{G}_-} \leq \text{val}^{\mathscr{G}_+}$.

For two positional strategies σ and τ, the play $\pi^v_{\sigma,\tau}$ is a lasso, meaning of the form πc^ω with π a simple path and c a simple cycle, implying that $\texttt{MeanPayoff}^+(\pi^v_{\sigma,\tau}) = \texttt{MeanPayoff}^-(\pi^v_{\sigma,\tau})$, let us write $\texttt{MeanPayoff}(\pi^v_{\sigma,\tau})$ for this value.

We have:

$$\texttt{MeanPayoff}(\pi^v_{\sigma_+,\tau_+}) \leq \texttt{MeanPayoff}(\pi^v_{\sigma_+,\tau_-}) \leq \texttt{MeanPayoff}(\pi^v_{\sigma_-,\tau_-}),$$

where the first inequality is by optimality of τ_- and the second inequality by optimality of σ_-. Hence $\text{val}^{\mathscr{G}_+} \leq \text{val}^{\mathscr{G}_-}$, and finally $\text{val}^{\mathscr{G}_+} = \text{val}^{\mathscr{G}_-}$.

For the second item, recall that $\text{val}^{\mathscr{G}}(v) = \texttt{MeanPayoff}(\pi^v_{\sigma,\tau})$ with σ and τ optimal positional strategies. Let us write $\pi^v_{\sigma,\tau} = \pi c^\omega$ with π a simple path and c a simple cycle, then by prefix independence $\texttt{MeanPayoff}(\pi^v_{\sigma,\tau}) = \text{Mean}(c)$, thus $\text{val}^{\mathscr{G}}(v)$ is the mean of at most n weights from $\mathscr{G}$. □

Solving mean payoff games in NP ∩ coNP

The positional determinacy of mean payoff games easily gives an upper bound on the complexity of *solving* these games.

Theorem 39 (Complexity of mean payoff)**.** *Solving mean payoff games is in* NP ∩ coNP.

Proof. The first ingredient for this proof is a polynomial time algorithm for solving the one player variants of mean payoff games. Indeed, they correspond to the minimum cycle mean problem in a weighted graph, which can be solved in polynomial time by a dynamic programming algorithm. The second ingredient is the positional determinacy result proved in Corollary 6.

Let us show the NP membership. Consider a mean payoff game $\mathscr{G}$, a vertex v and a threshold $x \in \mathbb{Q} \cup \{\pm\infty\}$. Thanks to Corollary 6, we know that there exist an optimal positional strategy for Max. With a non-deterministic Turing machine, we may guess a positional strategy for Max, and check that it ensures x in $\mathscr{G}$ from v.

Let us now show the coNP membership. By determinacy of mean payoff games, whether Max *cannot* ensure x in $\mathscr{G}$ from v is equivalent to whether Min can ensure x in $\mathscr{G}$ from v. Again thanks to Corollary 6, we know that there exist an optimal positional strategy for Min. With a non-deterministic Turing machine, we may guess a positional strategy for Min, and check that it ensures x in $\mathscr{G}$ from v. □

We can turn the non-deterministic algorithm given in Theorem 39 into a deterministic algorithm with exponential complexity since there are exponentially many positional strategies.

Reducing parity games to mean payoff games

We now show that solving mean payoff games is at least as hard as solving parity games.

Theorem 40 (Reducing parity games to mean payoff games)**.** *Solving parity games reduce in polynomial time to solving mean payoff games with threshold* 0*.*

Proof. Let $\mathscr{G} = (\mathscr{A}, \mathtt{Parity}(\mathfrak{c}))$ a parity game with n vertices and priorities in $[1,d]$. We construct a mean payoff game $\mathscr{G}' = (\mathscr{A}, \mathtt{MeanPayoff}[\mathfrak{c}'])$ using the same arena and the colouring function:

$$\mathfrak{c}'(e) = (-n)^{\mathfrak{c}(e)}.$$

Note that $\mathfrak{c}'(e)$ is of polynomial size since $\log(|\mathfrak{c}'(e)|) = \mathfrak{c}(e)\log(n) \leq d\log(n)$.

The key property relating $\mathfrak{c}'$ and $\mathfrak{c}$ is the following: let c a simple cycle

$$v_0 \xrightarrow{p_0} p_1 \xrightarrow{p_2} v_2 \cdots v_{k-1} \xrightarrow{p_{k-1}} v_0,$$

then the largest priority in c is even if and only if the mean value with respect to $\mathfrak{c}'$ is non-negative. Indeed, if the largest priority in c is p even, then it contributes n^p and all other values are greater than or equal to $-n^{p-1}$, and since there are at most n in total, the largest priority dominates the others.

We claim that for all vertices v, $v \in W_{\mathrm{Eve}}(\mathscr{G})$ if and only if $\mathrm{val}^{\mathscr{G}'} \geq 0$. Let $v \in W_{\mathrm{Eve}}(\mathscr{G})$, and σ a positional strategy winning from v in $\mathscr{G}$, we show that σ ensures mean payoff at least 0 in $\mathscr{G}'$ from v. Indeed, in $\mathscr{G}[\sigma, v]$ all cycles are even, which

thanks to the above implies that all cycles in $\mathscr{G}'[\sigma, v]$ are non-negative. The converse implication is similar: a positional strategy σ' in $\mathscr{G}'$ ensuring mean payoff at least 0 from v has the property that all cycles in $\mathscr{G}'[\sigma, v]$ are non-negative, hence that all cycles in $\mathscr{G}[\sigma, v]$ are even, therefore that it is winning in $\mathscr{G}$ from v. □

Note that we did not construct a reduction between objectives as defined in Section 1.6: indeed it is not true that $\texttt{Parity}$ reduces to $\texttt{MeanPayoff}_{\geq 0}$, the reduction depends on the number n of vertices. As a corollary of Theorem 39, this polynomial reduction gives an alternative proof of the fact that solving parity games is in $\mathsf{NP} \cap \mathsf{coNP}$.

A first value iteration algorithm using finite horizon payoffs

Let us give a first algorithm for solving mean payoff games using the value iteration paradigm. The idea is to consider the game where we only play for a fixed number of steps k, and the payoff is the sum of the weights. We compute iteratively the optimal values for increasing values of k, and show that for k large enough it allows us to obtain the values of mean payoff up by a simple rounding procedure.

Let $\mathscr{G} = (\mathscr{A}, \texttt{MeanPayoff}[\mathfrak{c}])$ a mean payoff game with n vertices and weights in $[-W, W]$. Recall that for a vertex u, the mean payoff value is defined as

$$\mathrm{val}^{\mathscr{G}}(u) = \sup_{\sigma} \inf_{\tau} \texttt{MeanPayoff}[\mathfrak{c}](\pi_u^{\sigma,\tau}).$$

Our goal is to compute these values, or compare them to a threshold.

Let us fix a number k and define the following objective summing the first k weights:

$$\texttt{Payoff}_k(\rho) = \sum_{i=0}^{k-1} \rho_i.$$

Let us define $\mathrm{val}^{\mathscr{G},k} : V \to \mathbb{Z}$ the value function for the $\texttt{Payoff}_k$ objective:

$$\mathrm{val}^{\mathscr{G},k}(u) = \sup_{\sigma} \inf_{\tau} \texttt{Payoff}_k[\mathfrak{c}](\pi_u^{\sigma,\tau}).$$

We let F_V be the set of functions $V \to \mathbb{Z}$, we define the operator $\mathbb{O}^{\mathscr{G}} : F_V \to F_V$ by:

$$\mathbb{O}^{\mathscr{G}}(\mu)(u) = \begin{cases} \max\left\{\mu(v) + w : u \xrightarrow{w} v \in E\right\} & \text{if } u \in V_{\mathrm{Max}}, \\ \min\left\{\mu(v) + w : u \xrightarrow{w} v \in E\right\} & \text{if } u \in V_{\mathrm{Min}}. \end{cases}$$

Let μ_0 defined by $\mu_0(u) = 0$ for all u and $\mu_{k+1} = \mathbb{O}^{\mathscr{G}}(\mu_k)$.

Lemma 33. *The following holds for all k.*

- *We have $\mu_k = val^{\mathscr{G},k}$.*
- *For all $u \in V$ we have $k \cdot val^{\mathscr{G}}(u) - 2nW \leq val^{\mathscr{G},k} \leq k \cdot val^{\mathscr{G}}(u) + 2nW$.*

Proof. The proof of the first item is an easy induction on k. We turn to the second item. Let us consider σ a positional optimal strategy for Max. We claim that for all k, the strategy σ ensures from u that $\mathtt{Payoff}_k[\mathfrak{c}]$ is at least $(k-n)\cdot \mathrm{val}^{\mathscr{G}}(u) - nW$. We first note that all cycles reachable from u in $\mathscr{G}[\sigma]$ have values (meaning, average value of the weights) at least $\mathrm{val}^{\mathscr{G}}(u)$. Now, consider a path of length k, and iteratively remove cycles from it. What remains is a most n edges, which contribute in the worst case to $-nW$. The remaining $k-n$ edges are included in some cycle, hence contribute $(k-n)\cdot \mathrm{val}^{\mathscr{G}}(u)$. Since $\mathrm{val}^{\mathscr{G}}(u) \leq W$, we obtain a lower bound of $k\cdot \mathrm{val}^{\mathscr{G}}(u) - 2nW$.

The argument is symmetrical for Adam. □

A direct corollary is as follows.

Corollary 9. *We have* $\lim_k \frac{1}{k}\cdot val^{\mathscr{G},k} = val^{\mathscr{G}}$.

However, we can make this effective using Corollary 8, which places bounds on the mean payoff values. This yields two very simple algorithms, one for computing the mean payoff values, and the other one for solving mean payoff games.

Theorem 41. *The following holds.*

- *There exists an algorithm running in time $O(n^3mW)$ for computing the mean payoff values.*
- *There exists an algorithm running in time $O(n^2mW)$ for solving the mean payoff games, meaning determining whether $val^{\mathscr{G}}(u) \geq c$ for some threshold c.*

Proof. Both arguments rely on Corollary 8.

- Let us fix $k = 4n^3W$. We can compute the values $\mathrm{val}^{\mathscr{G},k}$ in time $O(n^3mW)$ as explained above. The value of k was chosen in such a way that

$$\mathrm{val}^{\mathscr{G},k}(u) - \frac{1}{2n(n-1)} < \mathrm{val}^{\mathscr{G}}(u) < \mathrm{val}^{\mathscr{G},k}(u) + \frac{1}{2n(n-1)}.$$

  Indeed, since $\mathrm{val}^{\mathscr{G}}(u)$ is a rational number whose denominator is at most n, the minimum distance between two possible values of $\mathrm{val}^{\mathscr{G}}(u)$ is at most $\frac{1}{n(n-1)}$. Hence the exact value of $\mathrm{val}^{\mathscr{G}}(u)$ is the unique rational number with a denominator at most n that lies in the interval

$$\left[\ \mathrm{val}^{\mathscr{G},k}(u) - \frac{1}{2n(n-1)}\ ,\ \mathrm{val}^{\mathscr{G},k}(u) + \frac{1}{2n(n-1)}\ \right],$$

  which is easily computed.

- The distance between c and the closest rational number with a denominator at most n is $\frac{1}{n}$, so to determine whether $\mathrm{val}^{\mathscr{G}}(u) \geq c$ it is enough to compute the values $\mathrm{val}^{\mathscr{G},k}$ for $k = 4n^2W$.

□

A second value iteration algorithm using energy

Let us give a second algorithm, based on energy games. The construction of the algorithm will follow closely the high-level presentation of value iteration algorithms given in Section 1.9. Let $\mathscr{G} = (\mathscr{A}, \texttt{MeanPayoff}[\mathfrak{c}])$ a mean payoff game with n vertices and weights in $[-W, W]$.

The key insight is to use the energy objective. Recall that the energy quantitative objective is defined over the set of colours $C = \mathbb{Z}$:

$$\texttt{Energy}(\rho) = \inf \left\{ \ell \in \mathbb{N} : \forall k \in \mathbb{N},\ \ell + \sum_{i=0}^{k-1} \rho_i \geq 0 \right\}.$$

The interpretation is the following: weights are energy consumptions (negative values) and recharges (positive values), and $\texttt{Energy}(\rho)$ is the smallest initial budget ℓ such that Min can ensure that the energy level remains non-negative forever. Let us be careful here: maximising the mean payoff objective corresponds to minimising the energy objective. We formalise the relationship between mean payoff and energy in the following lemma.

Lemma 34 (Relating mean payoff and energy objectives)**.** *Let G be a mean payoff graph. Then G satisfies $\texttt{MeanPayoff}^- \geq 0$ if and only if it satisfies $\texttt{Energy} < \infty$.*

Proof. Let us say that a cycle in G is non-negative if the sum of its weights is non-negative. We consider the following properties:

(i) G satisfies $\texttt{MeanPayoff}^- \geq 0$.

(ii) All cycles in G are non-negative.

(iii) G satisfies $\texttt{Energy} < \infty$.

We prove the implications $(i) \Rightarrow (ii)$, then $(ii) \Rightarrow (iii)$, and finally $(iii) \Rightarrow (i)$.

$(i) \Rightarrow (ii)$ is clear.

$(ii) \Rightarrow (iii)$. Let us consider an infinite path, and strike out all edges involved in a cycle in it. At most n edges are not stricken out, incurring at most $-nW$ in energy drop. Since cycles are non-negative, the lowest level in a cycle is also lower bounded by $-nW$. Hence the energy level of the infinite path is at most $2nW$.

$(iii) \Rightarrow (i)$. Assume that G satisfies $\texttt{Energy} < \infty$, this implies that all partial sums are greater than or equal to a constant ℓ. This implies that the means of the partial sums are lower bounded by $\frac{\ell}{k}$, which converges to 0 when k goes to infinity. Therefore G satisfies $\texttt{MeanPayoff}^- \geq 0$. □

It is tempting to claim that a stronger property hold, namely $\texttt{MeanPayoff}^-(\rho) \geq 0$ if and only if $\texttt{Energy}(\rho) < \infty$. This is not the case.

Corollary 10. *Let $\mathscr{G}$ a mean payoff game. We define $\mathscr{G}'$ the energy game induced by $\mathscr{G}$. Then for all vertices u, we have $val^{\mathscr{G}}(u) \geq 0$ if and only if $val^{\mathscr{G}'}(u) < \infty$. Consequently:*

- *An algorithm computing the energy values induces an algorithm for solving mean payoff games with the same complexity.*

- *An algorithm computing the energy values in time $T(n,m,W)$ induces an algorithm for computing the mean payoff values in time $O(\log(n^2W)\cdot T(n,m,W))$.*

Proof. We consider σ a positional optimal strategy for Max in $\mathscr{G}$, we have $\mathrm{val}^{\mathscr{G}} = \mathrm{val}^{\mathscr{G},\sigma}$. Let u such that $\mathrm{val}^{\mathscr{G}}(u) \geq 0$, we consider the subgraph of $\mathscr{G}[\sigma,u]$ of vertices reachable from u. Thanks to Lemma 34 applied to this graph, since $\mathrm{val}^{\mathscr{G},\sigma}(u) \geq 0$ we have $\mathrm{val}^{\mathscr{G}',\sigma}(u) < \infty$. This implies that σ (seen as a strategy for Min in $\mathscr{G}'$) ensures $\mathtt{Energy} < \infty$, hence $\mathrm{val}^{\mathscr{G}'}(u) < \infty$.

Conversely, we consider τ a positional optimal strategy for Min in $\mathscr{G}'$, we have $\mathrm{val}^{\mathscr{G}'} = \mathrm{val}^{\mathscr{G}',\tau}$. Let u such that $\mathrm{val}^{\mathscr{G}'}(u) < \infty$, we consider the subgraph of $\mathscr{G}[\tau,u]$ of vertices reachable from u. Thanks to Lemma 34 applied to this graph, since $\mathrm{val}^{\mathscr{G}',\tau}(u) < \infty$ we have $\mathrm{val}^{\mathscr{G},\tau}(u) \geq 0$. This implies that τ (seen as a strategy for Max in $\mathscr{G}$) ensures $\mathtt{MeanPayoff} \geq 0$, hence $\mathrm{val}^{\mathscr{G}}(u) \geq 0$.

The first consequence is immediate. For the second, we use a simply binary search. Indeed, to determine whether $\mathrm{val}^{\mathscr{G}}(u) \geq c$, we can shift all weights by c, thus reducing to the question whether $\mathrm{val}^{\mathscr{G}}(u) \geq 0$, and compute the energy values. Thanks to Corollary 8 the mean payoff values are rational numbers in $[-W,W]$ whose denominators are at most n, so this requires $O(\log(n^2W))$ calls. □

Therefore, we set as a goal to compute the energy values.

Theorem 42 (Value iteration algorithm)**.** *There exists a value iteration algorithm which computes the energy values in time $O(mnW)$.*

We define $Y = \mathbb{N} \cup \{\infty\}$, equipped with the natural total order on integers. We let F_V be the lattice of functions $V \to Y$ equipped with the componentwise order induced by Y. We define a function $\delta : Y \times [-W,W] \to Y$ by $\delta(\ell,w) = \max(\ell - w, 0)$. This induces an operator $\mathbb{O}^{\mathscr{G}} : F_V \to F_V$:

$$\mathbb{O}^{\mathscr{G}}(\mu)(u) = \begin{cases} \min\left\{\delta(\mu(v),w) : u \xrightarrow{w} v \in E\right\} & \text{if } u \in V_{\mathrm{Min}}, \\ \max\left\{\delta(\mu(v),w) : u \xrightarrow{w} v \in E\right\} & \text{if } u \in V_{\mathrm{Max}}. \end{cases}$$

We note that $\mathbb{O}^{\mathscr{G}}$ is a monotonic operator, therefore it has a least fixed point.

Lemma 35 (Least fixed point for energy games)**.** *Let $\mathscr{G}$ an energy game with n vertices and weights in $[-W,W]$. Then the least fixed point of $\mathbb{O}^{\mathscr{G}}$ computes the energy values of $\mathscr{G}$.*

Proof. We first argue that the energy values, meaning the function $\mathrm{val}^{\mathscr{G}} : V \to \mathbb{Z} \cup \{\infty\}$, form a fixed point of the operator $\mathbb{O}^{\mathscr{G}}$. The fact that $\mathrm{val}^{\mathscr{G}} = \mathbb{O}^{\mathscr{G}}(\mathrm{val}^{\mathscr{G}})$ is a routine verification, which follows from two properties (see Lemma 8).

- For all ρ sequences of weights, we have $\mathtt{Energy}(w\cdot\rho) = \delta(\mathtt{Energy}(\rho),w)$.
- The function δ is monotonic and continuous.

It already yields one inequality: $\mathrm{val}^{\mathscr{G}}$ is larger than or equal to the least fixed point of $\mathbb{O}^{\mathscr{G}}$. Let us give another proof of the same inequality. We recall that thanks to Kleene fixed point theorem (Theorem 4), the least fixed point of $\mathbb{O}^{\mathscr{G}}$ is computed as follows:

$$\forall u \in V,\ \mu_0(u) = 0 \quad ; \quad \mu_{b+1} = \mathbb{O}^{\mathscr{G}}(\mu_b).$$

We have $\mu_0 \leq \mu_1 \leq \ldots$, and since F_V is a finite lattice, for some k we have that μ_k is the least fixed point of $\mathbb{O}^{\mathscr{G}}$. The crux here is to understand what are the values μ_b for $b = 0, 1, \ldots$. Let us define the truncated energy objective:

$$\mathtt{Energy}_b(\rho) = \inf \left\{ \ell \in \mathbb{N} : \forall k \in [0,b],\ \ell + \sum_{i=0}^{k-1} \rho_i \geq 0 \right\}.$$

The interpretation is the following: $\mathtt{Energy}_b(\rho)$ is the smallest initial budget ℓ such that Min can ensure that the energy level remain non-negative for the first b steps. A simple induction on b shows that μ_b is the values for $\mathtt{Energy}_b$ in $\mathscr{G}$. Note that $\mathtt{Energy}_0 \leq \mathtt{Energy}_1 \leq \cdots \leq \mathtt{Energy}$, hence $\mu_b = \mathrm{val}^{\mathtt{Energy}_b} \leq \mathrm{val}^{\mathscr{G}}$, the desired inequality.

Let us now prove the converse inequality: $\mathrm{val}^{\mathscr{G}}$ is smaller than or equal to the least fixed point of $\mathbb{O}^{\mathscr{G}}$. For this, we consider a fixed point μ of $\mathbb{O}^{\mathscr{G}}$, and argue that $\mathrm{val}^{\mathscr{G}} \leq \mu$. To this end, we extract from μ a strategy τ for Min, and show that $\mathrm{val}^{\tau} \leq \mu$; since we know that $\mathrm{val}^{\mathscr{G}} \leq \mathrm{val}^{\tau}$, this implies $\mathrm{val}^{\mathscr{G}} \leq \mu$. We define τ as an argmin strategy:

$$u \in V_{\mathrm{Min}} :\ \tau(u) \in \mathrm{argmin} \left\{ \delta(\mu(v), w) : u \xrightarrow{w} v \in E \right\}.$$

Let us define the graph $G = \mathscr{G}[\tau]$, by definition of τ for all edges $u \xrightarrow{w} v$ in G we have $\mu(u) \geq \delta(\mu(v), w)$. We claim that this implies that for all paths ρ from u in G, we have $\mathtt{Energy}(\rho) \leq \mu(u)$. To this end, we show that for all $k \in \mathbb{N}$ we have $\mu(u) + \sum_{i=0}^{k-1} \rho_i \geq 0$. We proceed by induction on k. This is clear for $k = 0$, let us assume that it holds for k and show that it also does for $k+1$. Let us write $\rho_0 = u \xrightarrow{w} v$. By the property above, we have $\mu(u) \geq \delta(\mu(v), w) = \max(\mu(v) - w, 0) \geq \mu(v) - w$. Hence

$$\mu(u) + \sum_{i=0}^{k} \rho_i = \mu(u) + w + \sum_{i=1}^{k} \rho_i \geq \mu(v) + \sum_{i=0}^{k-1} \rho_i \geq 0,$$

where the last inequality is by induction hypothesis for the path $\rho_1 \ldots \rho_{k-1}$ from v. This concludes the induction. □

Lemma 35 does not immediately yield a value iteration algorithm because the lattice Y is infinite. However, let us note that by positional determinacy, the value of a vertex is the value of a path consisting of a prefix of length at most n and a simple cycle, which is ∞ if the cycle is negative and in $[0, nW]$ otherwise. Hence we can equivalently define $Y = [0, nW] \cup \{\infty\}$ and $\delta(\ell, w) = \max(\ell - w, 0)$ if $\ell - w \leq nW$, and ∞ otherwise. It is clear that the least fixed points for both operators coincide thanks to the remark above, and now that we have a finite lattice we obtain the value iteration algorithm presented in Algorithm 4.1.

Algorithm 4.1: The value iteration algorithm for energy games.

Data: An energy game
for $u \in V$ **do**
  $\mu(u) \leftarrow 0$
repeat
  $\mu \leftarrow \mathbb{O}^{\mathscr{G}}(\mu)$
until $\mu = \mathbb{O}^{\mathscr{G}}(\mu)$;
return μ

The value iteration algorithm for energy games is conceptually very simple, its pseudocode is given in Algorithm 4.1. Since for each vertex its value can only increase, and at each iteration at least one vertex increases, the total number of iterations before reaching the fixed point is at most $O(nW)$. However to obtain the announced complexity of $O(nmW)$, one needs to be a little bit more subtle: in the naive version the computational cost of an iteration of $\mathbb{O}^{\mathscr{G}}$ is $O(nm)$, since we need to look at each vertex and each outgoing edge. This can be improved using a more involved data structure keeping track of vertices to be updated. We detail this below.

Refined value iteration algorithm for energy values

The pseudocode is given in Algorithm 4.2. For an edge $u \xrightarrow{w} v$ we say that it is incorrect if $\mu(u) < \delta(\mu(v), w)$. A vertex $u \in V_{\text{Max}}$ is incorrect if it has an outgoing edge which is incorrect, and a vertex $u \in V_{\text{Min}}$ is incorrect if all of its outgoing edges are incorrect. The key idea of the data structure we are building is not to keep track of all incorrect edges for vertices in V_{Min}, but rather to count them.

The data structure consists of the following objects:

- a value of Y for each vertex, representing the current function $\mu : V \to Y$;
- a set `Incorrect` of vertices (the order in which vertices are stored and retrieved from the set does not matter);
- a table `Count` storing for each vertex of Min a number of edges.

For our complexity analysis we use the unit cost RAM model, see Section 1.2 for details. In the case at hand let us choose for the machine word size $w = \log_2(m) + \log_2(W)$, so that an edge together with its weight can be stored in one machine word.

The invariant of the algorithm satisfied before each iteration of the repeat loop is the following:

- for $u \in V_{\text{Min}}$, the value of `Count`(u) is the number of incorrect edges of u;
- `Incorrect` is the set of incorrect vertices.

The invariant is satisfied initially thanks to the function `Init`. Let us assume that we choose and remove u from `Incorrect`. Since we modify only $\mu(u)$ the only potentially incorrect vertices are in `Incorrect` (minus u) and the incoming edges of

u; for the latter each of them is checked and added to `Incorrect`$'$ when required. By monotonicity, incorrect vertices remain incorrect so all vertices in `Incorrect` (minus u) are still incorrect. Hence the invariant is satisfied.

The invariant implies that the algorithm indeed implements Algorithm 4.1 hence returns the minimal fixed point, but it also has implications on the complexity. Indeed one iteration of the repeat loop over some vertex u involves

$$O\left(\left(|\mathrm{In}^{-1}(u)| + |\mathrm{Out}^{-1}(u)|\right)\right)$$

operations: the first term corresponds to updating $\mu(u)$ and `Incorrect`, which requires for each outgoing edge of u to compute δ, and the second term corresponds to considering all incoming edges of u. Thus the running time for a single iteration is

$$O\left(\sum_{u\in V}\left(|\mathrm{In}^{-1}(u)| + |\mathrm{Out}^{-1}(u)|\right)\right) = O(m).$$

Since there are at most $n \cdot W$ iterations, we obtain the running time of $O(n \cdot m \cdot W)$.

A strategy improvement algorithm for energy games

As explained above, solving energy games yields algorithms for solving mean payoff games. So, we continue our investigation of energy games, and now construct a strategy improvement algorithm for them.

Theorem 43 (Strategy improvement algorithm for energy games)**.** *There exists a strategy improvement algorithm for solving energy games in exponential time.*

We rely on the high-level presentation of strategy improvement algorithms given in Section 1.10, although it is not necessary to have read that part. We have already proved in Lemma 35 that the energy values correspond to the least fixed point of the operator $\mathbb{O}^{\mathscr{G}}$ defined by:

$$\mathbb{O}^{\mathscr{G}}(\mu)(u) = \begin{cases} \min\left\{\delta(\mu(v), w) : u \xrightarrow{w} v \in E\right\} & \text{if } u \in V_{\mathrm{Min}}, \\ \max\left\{\delta(\mu(v), w) : u \xrightarrow{w} v \in E\right\} & \text{if } u \in V_{\mathrm{Max}}. \end{cases}$$

Recall that $Y = [0, nW] \cup \{\infty\}$, and F_V is the lattice of functions $V \to Y$ equipped with the componentwise order induced by Y. The function $\delta : Y \times [-W, W] \to Y$ is defined by $\delta(\ell, w) = \max(\ell - w, 0)$ if $\ell - w \leq nW$, and ∞ otherwise.

Improving a strategy. Let σ a (positional) strategy for Max, and a vertex $u \in V_{\mathrm{Max}}$, we say that $e : u \xrightarrow{w} v$ is an *improving edge* if

$$\delta(\mathrm{val}^{\sigma}(v), w) > \mathrm{val}^{\sigma}(u).$$

Intuitively: according to val^{σ}, playing e is better than playing $\sigma(u)$.

Algorithm 4.2: The refined value iteration algorithm for energy games.

Function `Init()`**:**
- **for** $u \in V$ **do**
  - $\mu(u) \leftarrow 0$
- **for** $u \in V_{\text{Min}}$ **do**
  - **for** $u \xrightarrow{w} v \in E$ **do**
    - **if** *incorrect:* $\mu(u) < \delta(\mu(v), w)$ **then**
      - `Count`$(u) \leftarrow$ `Count`$(u) + 1$
  - **if** $\mathit{Count}(u) = \mathit{Degree}(u)$ **then**
    - Add u to `Incorrect`
- **for** $u \in V_{\text{Max}}$ **do**
  - **for** $u \xrightarrow{w} v \in E$ **do**
    - **if** *incorrect:* $\mu(u) < \delta(\mu(v), w)$ **then**
      - Add u to `Incorrect`

Function `Treat(`u`)`**:**
- **if** $u \in V_{\text{Max}}$ **then**
  - $\mu(u) \leftarrow \max\left\{\delta(\mu(v), w) : u \xrightarrow{w} v \in E\right\}$
- **if** $u \in V_{\text{Min}}$ **then**
  - $\mu(u) \leftarrow \min\left\{\delta(\mu(v), w) : u \xrightarrow{w} v \in E\right\}$

Function `Update(`u`)`**:**
- **if** $u \in V_{\text{Min}}$ **then**
  - `Count`$(u) \leftarrow 0$
- **for** $v \xrightarrow{w} u \in E$ *which is incorrect* **do**
  - **if** $v \in V_{\text{Min}}$ **then**
    - `Count`$(v) \leftarrow$ `Count`$(v) + 1$
    - **if** $\mathit{Count}(v) = \mathit{Degree}(v)$ **then**
      - Add v to `Incorrect`$'$
  - **if** $v \in V_{\text{Max}}$ **then**
    - Add v to `Incorrect`$'$

Function `Main()`**:**
- `Init` ()
- **for** $i = 0, 1, 2, \ldots$ **do**
  - `Incorrect`$' \leftarrow \emptyset$
  - **for** $u \in \mathit{Incorrect}$ **do**
    - `Treat` (u)
    - `Update` (u)
  - **if** $\mathit{Incorrect}' = \emptyset$ **then**
    - **return** μ
  - **else**
    - `Incorrect` $\leftarrow$ `Incorrect`$'$

Given a strategy σ and a set of improving edges S (for each $u \in V_{\text{Max}}$, S contains at most one outgoing edge of u), we write $\sigma[S]$ for the strategy

$$\sigma[S](u) = \begin{cases} e & \text{if there exists } e = u \xrightarrow{w} v \in S, \\ \sigma(v) & \text{otherwise.} \end{cases}$$

The difficulty is that an edge being improving does not mean that it is a better move than the current one in any context, but only according to the value function val^σ, so it is not clear that $\sigma[S]$ is better than σ. Strategy improvement algorithms depend on the following two principles:

- **Progress**: updating a strategy using improving edges is a strict improvement,
- **Optimality**: a strategy which does not have any improving edges is optimal.

Let us write $\sigma \leq \sigma'$ if for all vertices v we have $\text{val}^\sigma(v) \leq \text{val}^{\sigma'}(v)$, and $\sigma < \sigma'$ if additionally $\neg(\sigma' \leq \sigma)$.

The algorithm. The pseudocode of the algorithm is given in Algorithm 4.3.

Algorithm 4.3: The strategy improvement algorithm for energy games.

```
Data: An energy game 𝒢
Choose an initial strategy σ_0 for Max
for i = 0, 1, 2, ... do
    Compute val^{σ_i} and the set of improving edges
    if σ_i does not have improving edges then
        return σ_i
    Choose a non-empty set S_i of improving edges
    σ_{i+1} ← σ_i[S_i]
```

The potential reduction point of view. A first important insight into the algorithm is through so-called potential reductions. From a game $\mathcal{G}$ and a strategy σ with value val^σ, we define $\mathcal{G}_\sigma$ as follows. The two games are identical, except for the weights: if $u \xrightarrow{w} v$ in $\mathcal{G}$, then $u \xrightarrow{w+\text{val}^\sigma(v)-\text{val}^\sigma(u)} v$ in $\mathcal{G}_\sigma$.

Fact 15. *We have* $val^{\mathcal{G}} = val^\sigma + val^{\mathcal{G}_\sigma}$.

Proof. This follows from the observation that given any finite play

$$\pi = v_0 \xrightarrow{w_0} v_1 \cdots v_{k-1} \xrightarrow{w_{k-1}} v_k$$

in $\mathcal{G}$, writing the corresponding play $\pi' = v_0 \xrightarrow{w'_0} v_1 \cdots v_{k-1} \xrightarrow{w'_{k-1}} v_k$ in $\mathcal{G}_\sigma$, we have a telescoping sum:

$$\sum_{i=0}^{k-1} w'_i = \text{val}^\sigma(v_k) - \text{val}^\sigma(v_0) + \sum_{i=0}^{k-1} w_i.$$

□

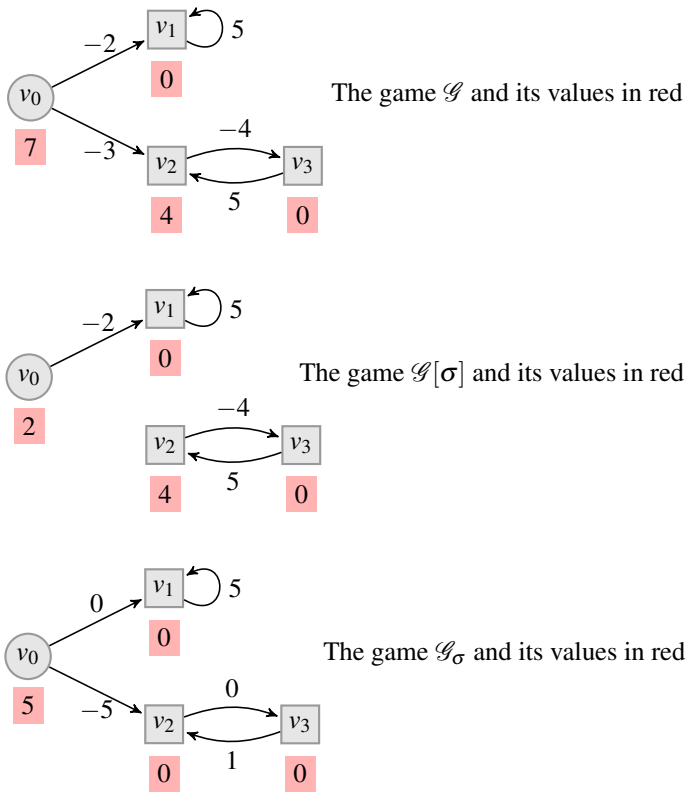


Figure 4.3: An example of potential reduction.

The benefit of this point of view is to interpret the notion of improving edges in $\mathscr{G}_\sigma$.

Fact 16. *Let $e = u \xrightarrow{w} v$ an edge in $\mathscr{G}$.*

- *if $u \in V_{\text{Max}}$, then e is an improving edge if and only if its weight in $\mathscr{G}_\sigma$ is negative.*
- *if $u \in V_{\text{Max}}$ and $\sigma(u) = e$ then the weight of e in $\mathscr{G}_\sigma$ is zero.*
- *if $u \in V_{\text{Min}}$, then the weight of e in $\mathscr{G}_\sigma$ is non-negative.*

The second and third properties directly follow from the fact that val^σ is a fixed point of $\mathbb{O}^{\mathscr{G}[\sigma]}$.

Proof of correctness. We now rely on Lemma 35 to prove the two principles: progress and optimality.

Lemma 36 (Progress for the strategy improvement algorithm for energy games)**.** *Let σ a strategy and S a set of improving edges. We let σ' denote $\sigma[S]$. Then $\sigma < \sigma'$.*

Proof. We use the potential reduction point of view. Let us apply Fact 15 to the game $\mathscr{G}[\sigma']$: we have $\mathrm{val}^{\sigma'} = \mathrm{val}^{\sigma} + \mathrm{val}^{G'}$. Thanks to Fact 16 in G' all weights are non-positive, implying that $\mathrm{val}^{G'} \geq 0$, and even $\mathrm{val}^{G'} > 0$ because the weights corresponding to improving edges are negative. □

Lemma 37 (Optimality for the strategy improvement algorithm for energy games)**.** *Let σ be a strategy that has no improving edges, then σ is optimal.*

Proof. We prove the contrapositive: assume that σ is not optimal, we show that it must have some improving edge. The fact that σ is not optimal means that $\mathrm{val}^{\sigma} < \mathrm{val}^{\mathscr{G}}$. Since $\mathrm{val}^{\mathscr{G}}$ is the least fixed point of $\mathbb{O}^{\mathscr{G}}$, it is also its least pre-fixed point. Therefore val^{σ} is not a pre-fixed point: $\neg(\mathrm{val}^{\sigma} \geq \mathbb{O}^{\mathscr{G}}(\mathrm{val}^{\sigma}))$. Hence there exists $u \in V$ such that $\mathrm{val}^{\sigma}(u) < \mathbb{O}^{\mathscr{G}}(\mathrm{val}^{\sigma})(u)$.

We rule out the case that $u \in V_{\mathrm{Min}}$: since val^{σ} is a fixed point of $\mathbb{O}^{\mathscr{G}[\sigma]}$, this implies that for $u \in V_{\mathrm{Min}}$ we have $\mathrm{val}^{\sigma}(u) = \min\left\{\delta(\mathrm{val}^{\sigma}(v), w) : u \xrightarrow{w} v \in E\right\}$, equal to $\mathbb{O}^{\mathscr{G}}(\mathrm{val}^{\sigma})(u)$. Therefore $u \in V_{\mathrm{Max}}$, implying that there exists $u \xrightarrow{w} v$ such that $\mathrm{val}^{\sigma}(u) < \delta(\mathrm{val}^{\sigma}(v), w)$. This is the definition of $u \xrightarrow{w} v$ being an improving edge. □

Complexity analysis. The computation of val^{σ} for a strategy σ can be seen to be a shortest path problem. Thus, any algorithm for the shortest path problem can be applied, such as the Bellman-Ford algorithm. In particular computing val^{σ} can be done in polynomial time, and even more efficiently through a refined analysis.

An aspect of the algorithm we did not develop is choosing the set of improving edges. Many possible rules for choosing this set have been studied, as for instance the *greedy all-switches* rule.

The next question is the number of iterations, meaning the length of the sequence $\sigma_0, \sigma_1, \ldots$. It is at most exponential since it is bounded by the number of strategies (which is bounded aggressively by m^n). There are lower bounds showing that the sequence can be of exponential length, which apply to different rules for choosing improving edges. Hence the overall complexity is exponential; we do not elaborate further here. We refer to Section 4.6 for bibliographic references and a discussion on the family of strategy improvement algorithms.

4.4 Discounted payoff games

From a practical point of view, the modelling of a real-world situation via mean payoff games requires that only the long-term behaviour is important. Since mean payoff only depends on the limit of the play, it cannot be used to model the beginning of the execution: the mean payoff is *prefix independent*. In economical studies, there is a tendency to make the prefixes count more, since they represent short-term implications of the actions taken, even if long-term behaviours also matter. The common payoff used to model this preference to prefixes is the discounted payoff that associates to a

play ρ the value

$$\texttt{DiscountedPayoff}(\rho) = (1-\lambda)\cdot\sum_{i=0}^{\infty}\lambda^i\rho_i,$$

where λ is a parameter in the interval $(0,1)$, ensuring the convergence of the infinite series (since weights π_i are bounded). We assume that λ is a rational number, it is part of the input of a discounted payoff game. The coefficient $1-\lambda$ before the series is just to counterbalance the fact that if all weights in the game are 1, we would like the payoff to be 1 too, which then holds since $\sum_{i=0}^{\infty}\lambda^i = \frac{1}{1-\lambda}$. When λ tends to 0, only the prefixes (and even the first weight) matters. On the contrary, when λ tends to 1, the discounted payoff looks more and more like the mean payoff. To grasp an intuition why this holds, consider a play that results from positional strategies in a mean payoff game. The weights encountered during the play then ultimately follow a periodic sequence $w_0, w_1, \ldots, w_{r-1}, w_0, w_1, \ldots, w_{r-1}, w_0, \ldots$ with average payoff $\frac{1}{r}\sum_{i=0}^{r-1} w_i$. Grouping the terms of the series $(1-\lambda)\sum_{i=0}^{\infty}\lambda^i w_i$ by batches of r terms, we then obtain

$$(1-\lambda)\sum_{i=0}^{\infty}\lambda^{ri}\sum_{j=0}^{r-1}\lambda^j w_j = \frac{1-\lambda}{1-\lambda^r}\sum_{j=0}^{r-1}\lambda^j w_j = \frac{1}{1+\lambda+\cdots+\lambda^{r-1}}\sum_{j=0}^{r-1}\lambda^j w_j$$

that tends towards the average-payoff $\frac{1}{r}\sum_{j=0}^{r-1} w_j$ when λ tends to 1.

The game of Figure 4.2 can also be equipped with a discounted payoff condition. If λ is close to 1, for instance $\lambda = 0.9$, then optimal strategies are the same as for the mean payoff objective:

$$\begin{array}{c}\sigma^*(v_0) = v_0 \xrightarrow{4} v_1 \quad ; \quad \sigma^*(v_2) = v_2 \xrightarrow{4} v_3 \\ \tau^*(v_1) = v_1 \xrightarrow{2} v_2 \quad ; \quad \tau^*(v_3) = v_3 \xrightarrow{-1} v_1 \quad ; \quad \tau^*(v_4) = v_4 \xrightarrow{-2} v_0.\end{array}$$

The play consistent with σ^* and τ^* starting from v_0 is the lasso word

$$v_0 \xrightarrow{4} v_1 \cdot \left(v_1 \xrightarrow{2} v_2 \xrightarrow{4} v_3 \xrightarrow{-1} v_1\right)^{\omega},$$

which has as discounted payoff $(1-\lambda)\left(4+\frac{2\lambda+4\lambda^2-\lambda^3}{1-\lambda^3}\right)$, it is approximately 1.7 when $\lambda = 0.99$. Recall that the mean payoff optimal value of vertex v_0 was $5/3 \approx 1.67$. However, the situation completely changes when λ decreases. When $\lambda = 0.5$ for instance, Min changes his decision in vertex v_1 and his optimal move is $v_1 \xrightarrow{0} v_0$. For a really low value of λ, for instance $\lambda = 0.1$, the decisions again change drastically for both players: now the optimal (positional) strategies become

$$\begin{array}{c}\sigma^*(v_0) = v_0 \xrightarrow{4} v_1 \quad ; \quad \sigma^*(v_2) = v_2 \xrightarrow{4} v_3 \\ \tau^*(v_1) = v_1 \xrightarrow{0} v_0 \quad ; \quad \tau^*(v_3) = v_3 \xrightarrow{-2} v_0 \quad ; \quad \tau^*(v_4) = v_4 \xrightarrow{-2} v_0.\end{array}$$

Computing the values using a contracting fixed point

Let us consider a discounted payoff game $\mathscr{G}$ with n vertices and weights in $[-W,W]$. We define Y as $\mathsf{R}\cup\{\pm\infty\}$, equipped with the natural total order on the reals. We let

F_V be the lattice of functions $V \to Y$ equipped with the componentwise order induced by Y. We equip F_V with the infinity norm: $\|\mu\| = \max_{u \in V} |\mu(u)|$.

We define a function $\delta : Y \times [-W, W] \to Y$ by $\delta(x, w) = \lambda \cdot x + (1-\lambda) \cdot w$. To understand the definition of δ, the key observation is that the discounted payoff can be computed recursively:

$$\begin{array}{rcl} \texttt{DiscountedPayoff}(\rho) & = & (1-\lambda) \cdot \sum_{i=0}^{\infty} \lambda^i \rho_i \\ & = & (1-\lambda) \cdot \rho_0 + \lambda \cdot \texttt{DiscountedPayoff}(\rho_{\geq 1}). \end{array}$$

The function δ induces an operator $\mathbb{O}^{\mathscr{G}} : F_V \to F_V$:

$$\mathbb{O}^{\mathscr{G}}(\mu)(u) = \begin{cases} \max \left\{ \delta(\mu(v), w) : u \xrightarrow{w} v \in E \right\} & \text{if } u \in V_{\text{Max}}, \\ \min \left\{ \delta(\mu(v), w) : u \xrightarrow{w} v \in E \right\} & \text{if } u \in V_{\text{Min}}. \end{cases}$$

We note that $\mathbb{O}^{\mathscr{G}}$ is a monotonic operator. Even more interesting, it is contracting:

Fact 17. *The operator $\mathbb{O}^{\mathscr{G}}$ is contracting with contraction factor λ, meaning that for all $\mu, \mu' \in F_V$:*

$$\|\mathbb{O}^{\mathscr{G}}(\mu) - \mathbb{O}^{\mathscr{G}}(\mu')\| \leq \lambda \cdot \|\mu - \mu'\|.$$

By Banach fixed point theorem, see Theorem 5, a direct consequence of this fact is that $\mathbb{O}^{\mathscr{G}}$ has a unique fixed point. The following theorem is the cornerstone of the study of discounted payoff games.

Lemma 38 (Values as unique fixed point)**.** *The discounted payoff values are the unique fixed point of the operator $\mathbb{O}^{\mathscr{G}}$. Moreover, the unique fixed point induces a pair of optimal positional strategies:*

$$\begin{array}{l} u \in V_{\text{Max}} : \ \sigma(u) \in \mathit{argmax} \left\{ \delta(\mathit{val}^{\mathscr{G}}(v), w) : u \xrightarrow{w} v \in E \right\} \\ u \in V_{\text{Min}} : \ \tau(u) \in \mathit{argmin} \left\{ \delta(\mathit{val}^{\mathscr{G}}(v), w) : u \xrightarrow{w} v \in E \right\}. \end{array}$$

Proof. Since we already know that $\mathbb{O}^{\mathscr{G}}$ has a unique fixed point, it is enough to show that $\text{val}^{\mathscr{G}}$ is a fixed point of $\mathbb{O}^{\mathscr{G}}$. This follows from two properties, see Lemma 8:

- For all ρ sequences of weights, we have

$$\texttt{DiscountedPayoff}(w \cdot \rho) = \delta(\texttt{DiscountedPayoff}(\rho), w).$$

- The function δ is monotonic and continuous.

We now move to the second point: the values imply optimal positional strategies. Let us define σ, τ two positional strategies:

$$\begin{array}{l} u \in V_{\text{Max}} : \ \sigma(u) \in \text{argmax} \left\{ \delta(\text{val}^{\mathscr{G}}(v), w) : u \xrightarrow{w} v \in E \right\} \\ u \in V_{\text{Min}} : \ \tau(u) \in \text{argmin} \left\{ \delta(\text{val}^{\mathscr{G}}(v), w) : u \xrightarrow{w} v \in E \right\}. \end{array}$$

To show that (σ, τ) is a pair of optimal strategies, we claim that $\text{val}^{\sigma} = \text{val}^{\tau} = \text{val}^{\mathscr{G}}$. Indeed, both val^{σ} and val^{τ} are fixed points of $\mathbb{O}^{\mathscr{G}}$, and since there exists a unique fixed point the claim follows. □

Let us illustrate the operator $\mathbb{O}^{\mathscr{G}}$ on the discounted payoff game of Figure 4.2. It is convenient to the contracting operator is:

$$\begin{array}{rcl}
\mathbb{O}^{\mathscr{G}}(\mu)(v_0) & = & \max\big((1-\lambda)\cdot 4+\lambda\cdot\mu(v_1),\ (1-\lambda)\cdot 5+\lambda\cdot\mu(v_4)\big)\\
\mathbb{O}^{\mathscr{G}}(\mu)(v_1) & = & \min\big((1-\lambda)\cdot 0+\lambda\cdot\mu(v_0),\ (1-\lambda)\cdot 2+\lambda\cdot\mu(v_2)\big)\\
\mathbb{O}^{\mathscr{G}}(\mu)(v_2) & = & \max\big((1-\lambda)\cdot 1+\lambda\cdot\mu(v_2),\ (1-\lambda)\cdot 4+\lambda\cdot\mu(v_3)\big)\\
\mathbb{O}^{\mathscr{G}}(\mu)(v_3) & = & \min\big((1-\lambda)\cdot(-2)+\lambda\cdot\mu(v_0),\ (1-\lambda)\cdot(-1)+\lambda\cdot\mu(v_1)\big)\\
\mathbb{O}^{\mathscr{G}}(\mu)(v_4) & = & \min\big((1-\lambda)\cdot(-2)+\lambda\cdot\mu(v_0),\ (1-\lambda)\cdot 2+\lambda\cdot\mu(v_4)\big)
\end{array}$$

A careful analysis gives the fixed points for all values of $\lambda \in (0,1)$, which in turn allows us to find the associated optimal positional strategies σ^* and τ^* on the various intervals of values for λ, summarised in the following table:

λ	$(0,\lambda_1]$	$(\lambda_1,\lambda_2]$	$(\lambda_2,\lambda_3]$	$(\lambda_3,1)$
$\sigma^*(v_0)$	v_4	v_4	v_1	v_1
$\tau^*(v_1)$	v_0	v_0	v_0	v_2
$\sigma^*(v_2)$	v_3	v_3	v_3	v_3
$\tau^*(v_3)$	v_0	v_1	v_1	v_1
$\tau^*(v_4)$	v_0	v_0	v_0	v_0

The frontiers are at $\lambda_1 = 1-\sqrt{2}/2 \approx 0.293$, $\lambda_2 = 1/2$, and $\lambda_3 \approx 0.841$. For instance, on interval $(0,\lambda_1]$, Min gets discounted payoff $\frac{5\lambda-2}{1+\lambda}$ when starting in vertex v_3, while switching his decision in interval $(\lambda_1,\lambda_2]$ allows him to secure $\frac{-2\lambda^3+6\lambda^2-1}{1+\lambda}$: this gives the explanation for the value of λ_1 which allows one to equal the two values. A similar reasoning provides the values of λ_2 and λ_3.

Solving discounted payoff games is in NP ∩ coNP

The first step in proving NP ∩ coNP upper bounds is to solve the one-player variants in polynomial time.

Lemma 39 (One player discounted payoff games)**.** *There exists a polynomial time algorithm for computing the optimal values of one-player discounted payoff games.*

Proof. Let us consider a discounted payoff game where only Max has moves. We claim that val^σ (seen as a vector $(x_u)_{u\in V}$) is the unique solution of the following linear program

$$\begin{array}{ll}
\text{maximise} & \sum_{u\in V} x_u\\
\text{subject to} & x_u \le (1-\lambda)\cdot w+\lambda x_v \quad \text{for } u \xrightarrow{w} c \in E,
\end{array}$$

Indeed, vectors satisfying the constraints are exactly pre-fixed points of $\mathbb{O}^\sigma$, and since $\mathbb{O}^\sigma$ is monotonic the least fixed point of $\mathbb{O}^\sigma$ coincide with its least pre-fixed point. Linear programming can be solved in polynomial time, see Theorem 2.

A dual argument solves the case where only Min has moves. □

As for mean payoff (or parity) games, the existence of positional optimal (or winning) strategies for both players, and the ability to solve in polynomial time the one-player version of these games, allows us to obtain easily an NP ∩ coNP complexity to

solve discounted payoff games. The use of the above contracting operator even ensures that the Turing machines guessing and checking the optimal strategies may indeed be designed as unambiguous (instead of just non-deterministic). Calling UP the class of problems that can be solved by an unambiguous Turing machine running in polynomial time, and coUP the class of problems whose complement are in UP, we then obtain the theorem:

Theorem 44 (Complexity)**.** *Solving discounted payoff games is in* $\mathsf{UP} \cap \mathsf{coUP}$.

Proof. Using the previous result, we know that the value of a discounted payoff game is the *unique* solution of the fixed point equation $\mu = \mathbb{O}^{\mathscr{G}}(\mu)$. Therefore, guessing μ and checking it is indeed a fixed point of $\mathbb{O}^{\mathscr{G}}$ can be done by an unambiguous Turing machine. To ensure that the machine runs in polynomial time, it only remains to show that the solution is of polynomial size. Let us fix σ, τ a pair of positional optimal strategies. Let us rewrite the equation in a matrix form:

- We write $\vec{x}$ for a vector of values, $\vec{x} \in \mathsf{R}^V$.
- We define the matrix $Q \in \{0,1\}^{V \times V}$: the entry $Q_{u,v}$ is 1 if $u \in V_{\text{Min}}$ and $\sigma(u) = u \xrightarrow{w} v$ or $u \in V_{\text{Max}}$ and $\tau(u) = u \xrightarrow{w} v$, and 0 otherwise.
- We define the vector $\vec{c} \in \mathbb{Z}^V$: the entry c_u is the weight of the edge $u \xrightarrow{w} v$ chosen by the strategies σ and τ.

With these notations, the equation rewrites

$$\vec{x} = (1-\lambda) \cdot \vec{c} + \lambda \cdot Q \cdot \vec{x}.$$

Letting $\lambda = a/b$ the rational discount factor, the above equation rewrites into

$$A \cdot \vec{x} = (b-a) \cdot \vec{c} \tag{4.1}$$

with $A = b \cdot I - a \cdot Q$ (I being the identity matrix). Therefore, A is a matrix that has at most two non-zero elements in each row: each of these non-zero elements can be written using at most $N = \max(\log_2 a, \log_2 b)$ bits (therefore polynomial in the representation of the game), and are therefore bounded in absolute value by 2^N. By induction on the size of the matrix, we can then show that the determinant of A is at most $4^{n \cdot N}$. The solution to Equation (4.1), using Cramer's formula, reads $x_v = \det(A_v)/\det(A)$ where A_v is the matrix obtained from A by replacing the v-th column with the vector $(b-a) \cdot \vec{c}$. Therefore, all components of $\vec{x}$ can be written with only a polynomial number of bits with respect to the size of the weights in the game and N.

The coUP membership follows, as in Theorem 39, from a dual reasoning for Min. □

A value iteration algorithm for discounted payoff games

Let us recall that thanks to Banach fixed point theorem, see Theorem 5, the fixed point is obtained as the limit of the following sequence:

$$\forall u \in V,\ \mu_0(u) = 0 \quad ; \quad \mu_{k+1} = \mathbb{O}^{\mathscr{G}}(\mu_k).$$

We have $\lim_k \mu_k = \mathrm{val}^{\mathscr{G}}$. To make sense of this sequence, let us define the following condition for each k:

$$\texttt{DiscountedPayoff}_k(\rho) = (1-\lambda)\cdot\sum_{i=0}^{k-1}\lambda^i\rho_i.$$

We define $\mathrm{val}^{\mathscr{G},k} : V \to \mathsf{R}\cup\{\pm\infty\}$ the value function by

$$\mathrm{val}^{\mathscr{G},k}(u) = \sup_{\sigma}\inf_{\tau}\texttt{DiscountedPayoff}_k[\mathfrak{c}](\pi_u^{\sigma,\tau})$$

Fact 18. *For all k, we have $val^{\mathscr{G},k} = \mu_k$.*

The remaining question is therefore to design a rounding procedure to compute the exact values. The following lemma places bounds on the discounted payoff values, in a similar manner as for Corollary 8 for mean payoff.

Lemma 40 (Upper bound on values in discounted payoff games)**.** *Let us write $\lambda = \frac{a}{b} \in (0,1)$. The discounted payoff values are rational numbers in $[-W,W]$ whose denominators are at most $D = b^{n-1}\prod_{j=1}^{n}(b^j - a^j)$.*

Proof. Let σ,τ a pair of optimal positional strategies. The corresponding play consists of a prefix of length at most n and a simple cycle, hence the sequence of weights is:

$$w_0, w_1, \ldots, w_{k-1}, (w_k, \ldots, w_\ell)^\omega,$$

with $k,\ell \le n$.

$$\begin{aligned}
\mathrm{val}^{\mathscr{G}}(v) &= (1-\lambda)\left[\sum_{i=0}^{k-1}\lambda^i w_i + \lambda^k\sum_{m=0}^{\infty}\lambda^{(\ell-k+1)m}\sum_{i=0}^{\ell-k}\lambda^i w_{k+i}\right]\\
&= \frac{b-a}{b}\left[\sum_{i=0}^{k-1}\frac{b^{k-1-i}a^i}{b^{k-1}}\cdot w_i + \frac{\lambda^k}{1-\lambda^{\ell-k+1}}\sum_{i=0}^{\ell-k}\frac{b^{\ell-k-i}a^i}{b^{\ell-k}}\cdot w_{k+i}\right]\\
&= \frac{N_1}{b^k} + \frac{a^k b^{\ell-k+1}}{b^{k+1}(b^{\ell-k+1}-a^{\ell-k+1})}\frac{N_2}{b^{\ell-k}} \qquad (\text{with } N_1, N_2 \in \mathbb{Z})\\
&= \frac{N_3}{b^k(b^{\ell-k+1}-a^{\ell-k+1})} \qquad (\text{with } N_3 \in \mathbb{Z})\\
&= \frac{N}{b^{n-1}\prod_{j=1}^{n}(b^j-a^j)} \qquad (\text{with } N \in \mathbb{Z})
\end{aligned}$$

□

It follows from Lemma 40 that if we have an approximation η of $\mathrm{val}^{\mathscr{G}}(v)$ such that $|\mathrm{val}^{\mathscr{G}}(v) - \eta| < \frac{1}{2D}$, we can recover the value as $\mathrm{val}^{\mathscr{G}}(v) = \frac{\lfloor D\eta + 1/2\rfloor}{D}$.

Let

$$K = \left\lceil \frac{1}{-\log_2\lambda}\left(\frac{n(n+3)}{2}\log_2 b + \log_2 W + 2\right)\right\rceil$$

Lemma 41 (Number of steps of value iteration)**.** *For the value K above, we have $\|F^K(\vec{0}) - val\| < \frac{1}{2D}$.*

Proof. First, we bound D by $b^{n+\frac{n(n+1)}{2}}$, so that $\frac{n(n+3)}{2}\log_2 b \geq \log_2 D$. Therefore $K \geq \frac{1}{-\log_2 \lambda}(\log_2 D + \log_2 W + \log_2 4) = \log_{1/\lambda}(4DW)$. This implies that $\lambda^K W \leq \frac{1}{4D} < \frac{1}{2D}$. Since $\mathbb{O}^{\mathscr{G}}$ is contracting with contraction factor λ, we have $\|F^K(\vec{0}) - \mathrm{val}^{\mathscr{G}}\|_\infty \leq \lambda^K \cdot \|\mathrm{val}^{\mathscr{G}}\|_\infty$. Since $|\mathrm{val}^{\mathscr{G}}(v)| \leq W$, we obtain the desired inequality. □

Algorithm 4.4: The value iteration algorithm for discounted payoff games.

Data: A discounted payoff game $\mathscr{G}$ with discount factor $\lambda = a/b \in (0,1)$, and F the contracting operator
for $u \in V$ **do**
  $\mu(u) \leftarrow 0$
$K \leftarrow \left\lceil \frac{1}{-\log_2 \lambda}\left(\frac{n(n+3)}{2}\log_2 b + \log_2 W + 2\right)\right\rceil$;
for $i = 1$ *to* K **do**
  $\mu \leftarrow \mathbb{O}^{\mathscr{G}}(\mu)$
return μ

Theorem 45 (Value iteration algorithm for discounted payoff games)**.** *There exists a value iteration algorithm computing in pseudo-polynomial time the discounted payoff values.*

One can check that K is polynomial in the size of the arena, but not in the discount factor λ. Indeed, consider that $\lambda = 1 - \frac{1}{b}$, with $b \in \mathbb{N} \setminus \{0\}$. Then, we may store λ with $\log_2 b$ bits, yet $\frac{1}{-\log_2 \lambda} \sim_{b\to\infty} b \ln 2$ is exponential in $\log_2 b$. As a consequence, the value iteration algorithm runs in pseudo-polynomial time.

Strategy improvement algorithm for discounted payoff games

Theorem 46 (Strategy improvement for discounted payoff games)**.** *There exists a strategy improvement algorithm computing the discounted payoff values in exponential time.*

We construct a strategy improvement algorithm, following the presentation in Section 1.10. Let us consider a (positional) strategy σ for Max. We compute val^σ (for instance by solving a linear program as in Lemma 39). Now there are two cases:

- Either $\mathrm{val}^\sigma = \mathbb{O}^{\mathscr{G}}(\mathrm{val}^\sigma)$, in which case we have found a fixed point of $\mathbb{O}^{\mathscr{G}}$. By uniqueness, this is the discounted payoff values, and σ is an optimal strategy for Max.
- Or $\mathrm{val}^\sigma \neq \mathbb{O}^{\mathscr{G}}(\mathrm{val}^\sigma)$, in which case we need to improve the strategy σ.

Let us consider the second case. Let $u \in V_{\text{Max}}$, we say that $u \xrightarrow{w} v \in E$ is an improving edge if $\text{val}^\sigma(u) < \delta(\text{val}^\sigma(v), w)$. Given a set of improving edges S (for each $u \in V_{\text{Max}}$, S contains at most one outgoing edge of u), we write $\sigma[S]$ for the strategy

$$\sigma[S](u) = \begin{cases} e & \text{if there exists } e = u \xrightarrow{w} v \in S, \\ \sigma(v) & \text{otherwise.} \end{cases}$$

Strategy improvement algorithms depend on the following two principles:

- **Progress**: updating a strategy using improving edges is a strict improvement,
- **Optimality**: a strategy which does not have any improving edges is optimal.

Let us write $\sigma \leq \sigma'$ if for all vertices v we have $\text{val}^\sigma(v) \leq \text{val}^{\sigma'}(v)$, and $\sigma < \sigma'$ if additionally $\neg(\sigma' \leq \sigma)$.

The algorithm. The pseudocode of the algorithm is given in Algorithm 4.5.

Algorithm 4.5: The strategy improvement algorithm for discounted payoff games.

```
Data: A discounted payoff game 𝒢
Choose an initial strategy σ_0 for Max
for i = 0, 1, 2, ... do
    Compute val^{σ_i} and the set of improving edges
    if σ_i does not have improving edges then
        return σ_i
    Choose a non-empty set S_i of improving edges
    σ_{i+1} ← σ_i[S_i]
```

An example Let us consider the discounted payoff game of Figure 4.2 with $\lambda = 0.5$, and start from the strategy $\sigma(v_0) = v_0 \xrightarrow{5} v_4$ and $\sigma(v_2) = v_2 \xrightarrow{1} v_2$.

We compute val^σ by solving the linear program:

$$\begin{array}{lrcl} \text{maximise} & & & x_0 + x_1 + x_2 + x_3 + x_4 \\ \text{subject to} & x_0 & = & (1-\lambda)\cdot 5 + \lambda \cdot x_4 \\ & x_1 & \leq & (1-\lambda)\cdot 0 + \lambda \cdot x_0 \\ & x_1 & \leq & (1-\lambda)\cdot 2 + \lambda \cdot x_2 \\ & x_2 & = & (1-\lambda)\cdot 1 + \lambda \cdot x_2 \\ & x_3 & \leq & (1-\lambda)\cdot (-2) + \lambda \cdot x_0 \\ & x_3 & \leq & (1-\lambda)\cdot (-1) + \lambda \cdot x_1 \\ & x_4 & \leq & (1-\lambda)\cdot (-2) + \lambda \cdot x_0 \\ & x_4 & \leq & (1-\lambda)\cdot 2 + \lambda \cdot x_4 \end{array}$$

Feeding this linear program to a solver we obtain the solution

$$\vec{x} = \left(\frac{8}{3}, \frac{4}{3}, 1, \frac{1}{6}, \frac{1}{3}\right).$$

The only improving edge is $v_2 \xrightarrow{4} v_3$, so we iterate with the strategy σ' defined by $\sigma'(v_0) = v_0 \xrightarrow{5} v_4$ and $\sigma'(v_2) = v_2 \xrightarrow{4} v_3$. The new vector of values found by solving the new linear program is

$$\vec{x'} = \left(\frac{8}{3}, \frac{4}{3}, \frac{25}{12}, \frac{1}{6}, \frac{1}{3}\right),$$

which is the (unique) fixed point of $\mathbb{O}^{\mathscr{G}}$.

Proof of correctness. We now prove the two principles: progress and optimality.

Lemma 42 (Progress for the strategy improvement algorithm for discounted payoff games)**.** *Let σ a strategy and S a set of improving edges. We let σ' denote $\sigma[S]$. Then $\sigma < \sigma'$.*

Proof. Let τ, τ' optimal positional strategies in $\mathscr{G}[\sigma]$ and $\mathscr{G}[\sigma']$. As in the proof of Theorem 44, letting Q, $\vec{c}$, Q', and $\vec{c'}$ the respective matrices and cost vectors described by the pairs of strategies (σ, τ) and (σ', τ'), we have

$$\mathrm{val}^{\sigma} = (1-\lambda)\cdot\vec{c} + \lambda\cdot Q\cdot\mathrm{val}^{\sigma} \qquad \text{and} \qquad \mathrm{val}^{\sigma'} = (1-\lambda)\cdot\vec{c'} + \lambda\cdot Q'\cdot\mathrm{val}^{\sigma'}.$$

Therefore:

$$\mathrm{val}^{\sigma'} - \mathrm{val}^{\sigma} = \lambda\cdot Q'\cdot(\mathrm{val}^{\sigma'} - \mathrm{val}^{\sigma}) + \underbrace{\lambda\cdot(Q'-Q)\cdot\mathrm{val}^{\sigma} + (1-\lambda)\cdot(\vec{c'}-\vec{c})}_{=\vec{\delta}}.$$

So:

$$(I - \lambda\cdot Q')\cdot(\mathrm{val}^{\sigma'} - \mathrm{val}^{\sigma}) = \vec{\delta}.$$

Since Q' is a positive matrix with coefficients in $\{0,1\}$, the series $\sum_i \lambda^i\cdot Q'^i$ converges, which shows that $I - \lambda\cdot Q'$ is invertible of inverse $\sum_{i=0}^{\infty}\lambda^i\cdot Q'^i$. In particular, the inverse $(I-\lambda Q')^{-1}$ has only non-negative coefficients, and its diagonal coefficients are positive. Therefore, to show that $\mathrm{val}^{\sigma'} - \mathrm{val}^{\sigma} = (I-\lambda\cdot Q')^{-1}\cdot\vec{\delta}$ is non-negative with at least one positive coefficient, it suffices to show that $\vec{\delta}$ is non-negative with at least one positive coefficient. Let $u \in V$:

- If $u \in V_{\mathrm{Max}}$, let $\sigma(u) = u \xrightarrow{w} v$ and $\sigma'(u) = u \xrightarrow{w'} v'$, we have

  $$\delta_u = \lambda\cdot(\mathrm{val}^{\sigma}(v') - \mathrm{val}^{\sigma}(v)) + (1-\lambda)(w'-w).$$

  If $\sigma(u) = \sigma'(u)$, then $\delta_u = 0$. Otherwise, $\delta_u > 0$ by definition of an improving edge.

- If $u \in V_{\mathrm{Min}}$, let $\tau(u) = u \xrightarrow{w} v$ and $\tau'(u) = u \xrightarrow{w'} v'$, we have

  $$\delta_u = \lambda\cdot(\mathrm{val}^{\sigma}(v') - \mathrm{val}^{\sigma}(v)) + (1-\lambda)(w'-w).$$

  By definition of τ we have $\mathrm{val}^{\sigma}(u) = \delta(\mathrm{val}^{\sigma}(v), w) \leq \delta(\mathrm{val}^{\sigma}(v'), w')$. This implies $\lambda\cdot\mathrm{val}^{\sigma}(v) + (1-\lambda)\cdot w \leq \lambda\cdot\mathrm{val}^{\sigma}(v') + (1-\lambda)\cdot w'$, so $\delta_u \geq 0$.

□

Lemma 43 (Optimality for the strategy improvement algorithm for discounted payoff games)**.** *Let σ be a strategy that has no improving edges, then σ is optimal.*

Proof. Let σ be a strategy that has no improving edges. We claim that val^σ is a fixed point of $\mathbb{O}^{\mathscr{G}}$, which implies that $\mathrm{val}^\sigma = \mathrm{val}^{\mathscr{G}}$, meaning that σ is optimal. Since val^σ is a fixed point of $\mathbb{O}^{\mathscr{G}[\sigma]}$, for $u \in V_{\mathrm{Min}}$ we have $\mathrm{val}^\sigma(u) = \min\left\{\delta(\mathrm{val}^\sigma(v), w) : u \xrightarrow{w} v \in E\right\}$. The fact that σ has no improving edges reads: for all $u \in V_{\mathrm{Max}}$, for all $u \xrightarrow{w'} v' \in E$, $\delta(\mathrm{val}^\sigma(v'), w') \leq \delta(\mathrm{val}^\sigma(v), w)$ where $\sigma(u) = u \xrightarrow{w} v$. Since $\mathrm{val}^\sigma(u) = \delta(\mathrm{val}^\sigma(v), w)$, this implies that $\mathrm{val}^\sigma(u) = \max\left\{\delta(\mathrm{val}^\sigma(v'), w) : v \xrightarrow{w} v' \in E\right\}$. The two equalities above witness that val^σ is the unique fixed point of $\mathbb{O}^{\mathscr{G}}$. □

The value iteration and strategy improvement algorithms are incomparable:

- the value iteration algorithm has a runtime pseudo-polynomial, and more precisely polynomial with respect to the number of vertices and the binary encoding of the weights of the arena, but exponential with respect to the binary encoding of λ,
- the strategy improvement algorithm has a runtime exponential with respect to the number of vertices, but polynomial with respect to the binary encoding of λ and the weights of the arena.

Reducing mean payoff games to discounted payoff games

Recall that Corollary 8 states that the mean payoff value $\mathrm{val}^{\mathscr{G}}(v)$ is a rational number with denominator at most n. The minimal distance between two such rational numbers is $\frac{1}{n-1} - \frac{1}{n} = \frac{1}{n(n-1)}$, implying that a $\frac{1}{2n(n-1)}$ approximation β of $\mathrm{val}^{\mathscr{G}}(v)$ is enough to apply a rounding procedure finding the only such rational in the interval $[\beta - \frac{1}{2n(n-1)}, \beta + \frac{1}{2n(n-1)}]$. By interpreting the mean payoff game as a discounted payoff game with a nicely chosen λ, we are able to find such a good approximation:

Theorem 47 (Discounted payoff approximation)**.** *Let $\mathscr{G}$ a mean payoff game. Let $\lambda \in (0,1)$, we define $\mathscr{G}_\lambda$ the discounted payoff game obtained from $\mathscr{G}$. We let val denote the mean payoff values and $val_\lambda(v)$ the discounted payoff values with λ as discount factor. Then*

$$\|val - val_\lambda\|_\infty \leq (1-\lambda) \cdot 2n \cdot W.$$

Proof. Let $u \in V$. We prove the inequality $\mathrm{val}_\lambda(u) - \mathrm{val}^{\mathscr{G}}(u) \geq -(1-\lambda) \cdot 2n \cdot W$ by reasoning on Max's strategies: a similar reasoning on Min's strategies allows one to obtain the other inequality $\mathrm{val}_\lambda(u) - \mathrm{val}^{\mathscr{G}}(u) \leq (1-\lambda) \cdot 2n \cdot W$.

Let σ, τ a pair of optimal positional strategies for the mean payoff game. The play starting from u consistent with σ and τ is a finite prefix and a simple cycle, hence the sequence of weights is:

$$\rho = w_0, w_1, \ldots, w_{k-1}, (w_k, \ldots, w_\ell)^\omega,$$

with $k, \ell \leq n$. Note that $\mathrm{val}^{\mathscr{G}}(u) = \frac{1}{\ell-k+1}\sum_{i=k}^{\ell} w_i$. Playing σ in the discounted payoff game, we obtain that $\mathrm{val}_\lambda(u) \geq \texttt{DiscountedPayoff}(\rho)$. We now compute precisely $\texttt{DiscountedPayoff}(\rho)$:

$$(1-\lambda)\cdot\sum_{i=0}^{k-1}\lambda^i\cdot\underbrace{w_i}_{\geq -W} + (1-\lambda)\cdot\lambda^k\cdot\sum_{m=0}^{\infty}\lambda^{(\ell-k+1)m}\sum_{i=0}^{\ell-k}\lambda^i w_{k+i}$$

The first term is greater than or equal to $-(1-\lambda^k)\cdot W$, and the second term is equal to

$$\frac{(1-\lambda)\cdot\lambda^k}{1-\lambda^{\ell-k+1}}\sum_{i=0}^{\ell-k}\lambda^i\cdot w_{k+i}.$$

By shifting all weights by W, we can rewrite the sum as:

$$\sum_{i=0}^{\ell-k}\lambda^i\cdot w_{k+i} = \sum_{i=0}^{\ell-k}\lambda^i\cdot(w_{k+i}+W) - W\cdot\sum_{i=0}^{\ell-k}\lambda^i$$

By using the fact that $w_{k+i}+W$ is non-negative and $\lambda^i \geq \lambda^{\ell-k}$, we obtain

$$\begin{aligned}\sum_{i=0}^{\ell-k}\lambda^i\cdot w_{k+i} &\geq \lambda^{\ell-k}\cdot\sum_{i=0}^{\ell-k}(w_{k+i}+W) - W\cdot\frac{1-\lambda^{\ell-k+1}}{1-\lambda}\\ &= \lambda^{\ell-k}\sum_{i=0}^{\ell-k}w_{k+i} + (\ell-k+1)\cdot\lambda^{\ell-k}\cdot W - W\cdot\frac{1-\lambda^{\ell-k+1}}{1-\lambda}\end{aligned}$$

Therefore, since $\mathrm{val}^{\mathscr{G}}(u) = \frac{1}{\ell-k+1}\cdot\sum_{i=k}^{\ell} w_i$, we obtain

$$\sum_{i=0}^{\ell-k}\lambda^i\cdot w_{k+i} \geq \lambda^{\ell-k}\cdot(\ell-k+1)\cdot(\mathrm{val}^{\mathscr{G}}(u)+W) - W\cdot\frac{1-\lambda^{\ell-k+1}}{1-\lambda}.$$

Together with the first term, we have

$$\texttt{DiscountedPayoff}(\pi) \geq -W + \frac{(1-\lambda)\cdot(\ell-k+1)}{1-\lambda^{\ell-k+1}}\cdot\lambda^\ell\cdot(\mathrm{val}^{\mathscr{G}}(u)+W)$$

Since $\frac{1-\lambda^{\ell-k+1}}{1-\lambda} = \sum_{i=0}^{\ell-k}\lambda^i < \ell-k+1$ and $\mathrm{val}^{\mathscr{G}}(u)+W \geq 0$, we have

$$\texttt{DiscountedPayoff}(\pi) \geq -W + \lambda^\ell\cdot(\mathrm{val}^{\mathscr{G}}(u)+W)$$

Finally, since $\ell \leq n$ we have $\lambda^\ell \geq \lambda^n > 1-n\cdot(1-\lambda)$, using the fact that $\frac{1-\lambda^n}{1-\lambda} = \sum_{i=0}^{n-1}\lambda^i < n$. Therefore, using again $\mathrm{val}^{\mathscr{G}}(u) \leq W$,

$$\begin{aligned}\texttt{DiscountedPayoff}(\pi) &\geq -W + (1-n\cdot(1-\lambda))\cdot(\mathrm{val}^{\mathscr{G}}(u)+W)\\ &= -n\cdot(1-\lambda)\cdot(W+\mathrm{val}^{\mathscr{G}}(u)) + \mathrm{val}^{\mathscr{G}}(u)\\ &\geq -2n\cdot(1-\lambda)\cdot W + \mathrm{val}^{\mathscr{G}}(u)\end{aligned}$$

We obtain

$$\mathrm{val}_\lambda(u) - \mathrm{val}^{\mathscr{G}}(u) \geq -2n(1-\lambda)\cdot W.$$

□

A direct corollary of Theorem 47 is an algorithm for computing the mean payoff values: picking $\lambda = 1 - \frac{1}{4n^2(n-1)W}$, we obtain a good enough approximation of the mean payoff values by solving the associated discounted payoff game. From a complexity point of view, this value iteration algorithm runs in polynomial time in the size of the arena, but exponential with respect to the representation of λ.

The previous reductions implies an improved theoretical complexity for mean payoff and parity games.

Corollary 11 (Complexity)**.** *Solving mean payoff games and parity games is in* $\mathsf{UP} \cap \mathsf{coUP}$.

Proof. The reduction from mean payoff to discounted payoff games allows to lift the $\mathsf{UP} \cap \mathsf{coUP}$ complexity of Theorem 44. Moreover, the reduction of Theorem 40 implies the same complexity for parity games. □

4.5 Shortest path games

The quantitative objective `Sup` generalises the qualitative objective `Reach` by stating numerical preferences on the target. Another quantitative extension of the reachability objective is to quantify the cost of a path towards the target: we define the quantitative objective `ShortestPath` over the set of colours $C = \mathbb{Z} \cup \{\text{Win}\}$ by

$$\texttt{ShortestPath}(\rho) = \begin{cases} \sum_{i=0}^{k-1} \rho_i & \text{for } k \text{ the first index such that } \rho_k = \text{Win}, \\ \infty & \text{if } \rho_k \neq \text{Win for all } k. \end{cases}$$

We interpret the weights as costs and Min is trying to reach the target with the smallest possible cost. Note that we use the same abusive terminology as for the shortest path graph problem: the *cost* of a path is the sum of the weights along it (until the first occurrence of Win) and we are looking for a path of minimal cost, hence not necessarily the shortest in number of edges.

We fix a shortest path game $\mathscr{G}$. Without loss of generality we assume that Win appears only in a sink. Recall that by definition:

$$\text{val}^{\mathscr{G}}(u) = \sup_{\sigma} \inf_{\tau} \texttt{ShortestPath}(\pi^{v}_{\sigma,\tau}).$$

Hence for a vertex u there are three possibilities:

- $\text{val}^{\mathscr{G}}(u) = \infty$, meaning that Min cannot ensure to reach Win,
- $\text{val}^{\mathscr{G}}(u) \in \mathbb{Z}$, meaning that Min can ensure to reach Win with a finite cost (bounded from below),
- $\text{val}^{\mathscr{G}}(u) = -\infty$, meaning that Min can ensure to reach Win with arbitrarily negative cost.

Note that if all weights are one, then $\text{val}^{\mathscr{G}}$ is the rank defined in the attractor computation, see Section 2.1.

Detecting whether $\text{val}^{\mathscr{G}}(u) = \infty$ is easy:

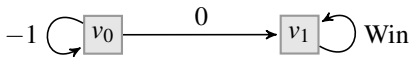


Figure 4.4: An example of a shortest path game with negative weights where Min does not have an optimal strategy. Indeed $\mathrm{val}^{\mathscr{G}}(v_0) = -\infty$ since for any k, Min has a strategy ensuring that `ShortestPath` is $-k$ by using k times the self loop -1 before Win. However, if Min never sees Win the outcome is ∞.

Lemma 44 (Detection of ∞ values)**.** *Let $\mathscr{G}$ a shortest path game and u a vertex. Then $\mathrm{val}^{\mathscr{G}}(u) < \infty$ if and only if $u \in W_{\mathrm{Min}}(\mathtt{Reach}(\mathit{Win}))$. Consequently, if $\mathrm{val}^{\mathscr{G}}(u) < \infty$, then $\mathrm{val}^{\mathscr{G}}(u) \leq nW$.*

Proof. The first equivalence is clear. For the second statement, consider a positional strategy ensuring $\mathtt{Reach}(\mathrm{Win})$, it ensures to reach Win within at most n steps, hence incurs a cost bounded by n times the largest weight. □

Before moving to algorithms, let us illustrate two difficulties:

- As illustrated in Figure 4.4, Min does not have optimal strategies in general.
- As illustrated in Figure 4.5, even if Min has an optimal strategy, it may not be positional.

We discuss Figure 4.5, which is a shortest path game where Min has an optimal strategy, but it cannot be made positional. First, Min has two positional strategies: $\tau_1(v_0) = v_0 \xrightarrow{-1} v_1$ and $\tau_2(v_0) = v_0 \xrightarrow{0} v_2$. Strategy τ_1 does not ensure to reach the target, since Max can enforce the cycle $v_0 \xrightarrow{0} v_1 \xrightarrow{0} v_0$ forever and obtain payoff ∞. Strategy τ_2 guarantees a payoff of 0. However, Min can be smarter by threatening Max. If Min plays once τ_1, and then switches to τ_2, he guarantees a payoff of -1. Doing so twice, he guarantees a payoff of -2. This reasoning is valid for playing up to 50 times τ_1, showing to Min can ensure a payoff of -50. However it is not valid beyond: against the strategy that plays 51 times τ_1, Max's optimal decision is to stop the game with $v_1 \xrightarrow{-50} v_2$, ensuring a payoff of -50. Indeed and more generally (as we will see), Max has a positional optimal strategy, which is to choose $v_1 \xrightarrow{-50} v_2$ right from the beginning.

Detection of $-\infty$ values using mean payoff games

We call 'detecting $-\infty$ values in shortest path games' the following decision problem: given a shortest path game $\mathscr{G}$ and a vertex u, do we have $\mathrm{val}^{\mathscr{G}}(u) = -\infty$?

Theorem 48 (Detection of $-\infty$ values using mean payoff games)**.** *Detecting $-\infty$ values in shortest path games is polynomial time equivalent to solving mean payoff games.*

We construct two simple reductions.

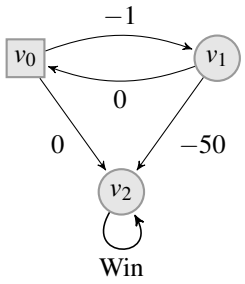


Figure 4.5: A shortest path game where Min needs memory to play optimally.

Lemma 45. *Let $\mathcal{G}$ a shortest path game, and $\mathcal{G}'$ the mean payoff game obtained from $\mathcal{G}$ by replacing the self loop with Win by a self loop with 0. Assume that for all vertices u we have $\mathrm{val}^{\mathcal{G}}(u) \neq \infty$. Then for all vertices u, we have $\mathrm{val}^{\mathcal{G}}(u) = -\infty$ if and only if $\mathrm{val}^{\mathcal{G}'}(u) < 0$. As a consequence, if $\mathrm{val}^{\mathcal{G}}(u) > -\infty$, then $\mathrm{val}^{\mathcal{G}}(u) \geq -nW$.*

Note that as stated in Lemma 44, detecting vertices with value ∞ can be done in polynomial time, hence it does not reduce the generality of the reduction.

Proof. Let us fix σ an optimal positional strategy for Max in the mean payoff game $\mathcal{G}'$. Since for all vertices u we have $\mathrm{val}^{\mathcal{G}}(u) \neq \infty$, thanks to Lemma 44 there exists a strategy σ_0 which ensures $\texttt{ShortestPath} \leq nW$ from all vertices.

Assume that $\mathrm{val}^{\mathcal{G}}(u) = -\infty$. Let τ_M a strategy ensuring $\texttt{ShortestPath} < -nW$, and look at a play consistent with σ and τ_M. It contains necessarily a negative cycle, implying that σ does not ensure $\texttt{MeanPayoff} \geq 0$. Since σ is optimal, this implies that $\mathrm{val}^{\mathcal{G}'} < 0$.

For the converse implication, assume now that $\mathrm{val}^{\mathcal{G}'}(u) < 0$. This implies that all cycles consistent with σ are negative. Consequently, for all paths consistent with σ, the sum of the weights diverges to $-\infty$. Let us fix M, and consider the strategy σ_M which plays like σ until the sum of the weights reaches below $M - nW$, and then switches to σ_0. This strategy ensures $\texttt{ShortestPath} \leq M$. Thus $\mathrm{val}^{\mathcal{G}}(u) = -\infty$.

We now explain how it follows that if $\mathrm{val}^{\mathcal{G}}(u) > -\infty$, then $\mathrm{val}^{\mathcal{G}}(u) \geq -nW$. Since the optimal positional strategy σ ensures $\texttt{MeanPayoff} \geq 0$, it also ensures that the sum of the weights remains larger than $-nW$ at all times. □

Lemma 46. *Let $\mathcal{G}$ a mean payoff game. We construct $\mathcal{G}'$ a shortest path game where after each edge, Min has a choice to stop the game with colour Win. Then for all vertices u in the original game $\mathcal{G}$, we have $\mathrm{val}^{\mathcal{G}}(u) < 0$ if and only if $\mathrm{val}^{\mathcal{G}'}(u) = -\infty$.*

Proof. Note that Min can ensure to reach Win from anywhere, so Min has a strategy to reach Win, implying that for all u we have $\mathrm{val}^{\mathcal{G}'}(u) \neq \infty$.

Let $\mathcal{G}''$ the mean payoff obtained from $\mathcal{G}'$ as in Lemma 45, we have that $\mathrm{val}^{\mathcal{G}'}(u) = -\infty$ if and only if $\mathrm{val}^{\mathcal{G}''}(v) < 0$. To conclude, it only remains to show that $\mathrm{val}^{\mathcal{G}}(u) < 0$ if and only if $\mathrm{val}^{\mathcal{G}''}(u) < 0$. The game $\mathcal{G}''$ is exactly as $\mathcal{G}$, except that anytime Min can

stop the game and settle for a mean payoff of 0. Hence when asking whether the mean payoff value is < 0, this option is not relevant. □

A value iteration algorithm for shortest path games

Theorem 49 (Half-positional determinacy in shortest path games)**.** *Shortest path games are half-positionally determined over finite arenas.*

This half-positional determinacy result will follow from the correctness of a value iteration algorithm, which works for shortest path games where no vertices has value $-\infty$. Thanks to the above, this can be computed and removed from the game. We say that such a game is normalised. Note that it is enough to prove half-positional determinacy for normalised games, since on the vertices having value $-\infty$ the strategy of Max is irrelevant.

Theorem 50 (Value iteration algorithm)**.** *There exists a value iteration algorithm for computing the value function of normalised shortest path games in pseudo-polynomial time and space.*

Our first lemma shows the existence of optimal strategies.

Lemma 47 (Optimal strategies)**.** *Let $\mathscr{G}$ be a normalised shortest path game, then there exists an optimal strategy for Min.*

Proof. Thanks to the assumption that the game is normalised and Lemma 45, the values are lower bounded by $-nW$, which implies that the infimum is indeed a minimum. □

Figure 4.4 shows that the assumption that the game is normalised in Lemma 47 is necessary.

We follow the high-level presentation of value iteration algorithms given in Section 1.9. Let us define $Y = \mathbb{Z} \cup \{-\infty, \infty\}$ equipped with the natural order and the function $\delta : Y \times (\mathbb{Z} \cup \{\text{Win}\}) \to Y$ by

$$\delta(x,w) = \begin{cases} 0 & \text{if } w = \text{Win}, \\ x+w & \text{if } w \in \mathbb{Z}. \end{cases}$$

We let F_V be the lattice of functions $\mu : V \to Y$ equipped with the componentwise order induced by Y. Note that δ is monotonic, it induces the monotonic operator $\mathbb{O}^{\mathscr{G}} : F_V \to F_V$ defined by:

$$\mathbb{O}^{\mathscr{G}}(\mu)(u) = \begin{cases} \max\left\{\delta(\mu(v),w) : u \xrightarrow{w} v \in E\right\} & \text{if } u \in V_{\text{Max}}, \\ \min\left\{\delta(\mu(v),w) : u \xrightarrow{w} v \in E\right\} & \text{if } u \in V_{\text{Min}}. \end{cases}$$

Thanks to Theorem 4, the operator $\mathbb{O}^{\mathscr{G}}$ has a greatest fixed point which is also the greatest post-fixed point of $\mathbb{O}^{\mathscr{G}}$. Recall that a post-fixed point of $\mathbb{O}^{\mathscr{G}}$ is a function $\mu \in F_V$ such that $\mathbb{O}^{\mathscr{G}}(\mu) \geq \mu$, it is also called a progress measure. Unfolding the definitions, for all vertices u, we have

$$\begin{array}{ll} \exists u \xrightarrow{w} v \in E, & \mu(u) \leq \delta(\mu(v),w) \quad \text{if } u \in V_{\text{Max}}, \\ \forall u \xrightarrow{w} v \in E, & \mu(u) \leq \delta(\mu(v),w) \quad \text{if } u \in V_{\text{Min}}. \end{array}$$

Lemma 48 (Shortest path values as greatest fixed point)**.** *Let $\mathscr{G}$ be a normalised shortest path game, then $\mathrm{val}^{\mathscr{G}}$ is the greatest fixed point of $\mathbb{O}^{\mathscr{G}}$.*

Proof. We show two properties:

- $\mathrm{val}^{\mathscr{G}}$ is a progress measure;
- $\mathrm{val}^{\mathscr{G}}$ is larger than the greatest fixed point of $\mathbb{O}^{\mathscr{G}}$.

Since the greatest fixed point of $\mathbb{O}^{\mathscr{G}}$ is also the greatest progress measure, the first item implies that it is larger than $\mathrm{val}^{\mathscr{G}}$, and the second item states the converse inequality.

The first item is a routine verification, which follows from two properties (see Lemma 8).

- For all ρ sequences of weights, we have

$$\texttt{ShortestPath}(w \cdot \rho) = \delta(\texttt{ShortestPath}(\rho), w).$$

- The function δ is monotonic and continuous.

We now show the second item: $\mathrm{val}^{\mathscr{G}}$ is larger than or equal to the greatest fixed point of $\mathbb{O}^{\mathscr{G}}$. For this, we consider a fixed point μ of $\mathbb{O}^{\mathscr{G}}$, and argue that $\mathrm{val}^{\mathscr{G}} \geq \mu$. To this end, we extract from μ a strategy σ for Max, and show that $\mathrm{val}^{\sigma} \geq \mu$; since we know that $\mathrm{val}^{\mathscr{G}} \geq \mathrm{val}^{\sigma}$, this implies $\mathrm{val}^{\mathscr{G}} \geq \mu$. We define σ as an argmax strategy:

$$u \in V_{\mathrm{Max}} : \ \sigma(u) \in \operatorname{argmax} \left\{ \delta(\mu(v), w) : u \xrightarrow{w} v \in E \right\}.$$

Let us define the graph $G = \mathscr{G}[\sigma]$, by definition of σ for all edges $u \xrightarrow{w} v$ in G we have $\mu(u) \leq \delta(\mu(v), w)$. We claim that this implies that for all paths ρ from u in G, we have $\texttt{ShortestPath}(\rho) \geq \mu(u)$. If $\texttt{ShortestPath}(\rho) = \infty$, this is clear, so let us consider the finite case, and proceed by induction on the length k of the path before reaching Win. This is clear for $k = 0$, since both values are 0. Let us assume that it holds for k and show that it also does for $k+1$. Let us write $\rho_0 = u \xrightarrow{w} v$. By the property above, we have $\mu(u) \leq \delta(\mu(v), w) = \mu(v) + w$. Hence

$$\sum_{i=0}^{k} \rho_i = w + \sum_{i=1}^{k} \rho_i \geq w + \mu(v) \geq \mu(u),$$

where the first inequality is by induction hypothesis for the path $\rho_1 \dots \rho_{k-1}$ from v. This concludes the induction. □

Lemma 48 does not immediately yield a value iteration algorithm because the lattice Y is infinite. However, let us note that by half-positional determinacy, the value of a vertex is either $\infty, -\infty$, or in $[-nW, nW]$. Hence we can equivalently define $Y = [-nW, nW] \cup \{-\infty, \infty\}$ and $\delta : Y \times (\mathbb{Z} \cup \{\mathrm{Win}\}) \to Y$ by

$$\delta(x, w) = \begin{cases} 0 & \text{if } w = \mathrm{Win}, \\ x + w & \text{if } w \in \mathbb{Z} \text{ and } x + w \in [-nW, nW], \\ \infty & \text{if } w \in \mathbb{Z} \text{ and } x + w > nW, \\ -\infty & \text{if } w \in \mathbb{Z} \text{ and } x + w < -nW. \end{cases}$$

It is clear that the greatest fixed points for both operators coincide thanks to the remark above, and now that we have a finite lattice we can use the generic monotonic fixed point computation (see Theorem 4) to compute $\mathrm{val}^{\mathscr{G}}$, which yields a pseudo-polynomial time and space algorithm.

Shortest path games with non-negative weights

Theorem 51 (Polynomial time algorithm for shortest path games with non-negative weights)**.** *There exists an algorithm for computing the values in shortest path games with non-negative weights running in time $O(m+n\log(n))$.*

In the case where all weights are non-negative, the value iteration algorithm can be made much more efficient, and in particular in polynomial time. To understand this, let us start by noting that the one-player case where only Min has moves is the classical shortest path problem (towards Win) for graphs. We extend Dijkstra's algorithm, see Algorithm 4.6 for the pseudocode.

Let us denote by $S_i, \mu_i(u)$, $\mu_i(u \xrightarrow{w} v)$ the values in iteration i. We define the invariants satisfied by the algorithm.

1. $\mu_i(u)$ is value of u in the shortest path game $\mathscr{G}_i$ obtained by replacing $u \xrightarrow{w} v$ with $u \xrightarrow{\infty} v$ if both u and v are still in S_i;
2. $\min\{\mu_i(u) : u \in S_i\} \geq \max\left\{\mathrm{val}^{\mathscr{G}}(u) : u \notin S_i\right\}$.

The second invariant generalises the greedy property of Dijkstra's algorithm.

Since the weight of every edge in $\mathscr{G}_i$ is non-increasing, the values $\mathrm{val}^{\mathscr{G}_i}(u)$ are also non-increasing. The invariants show imply that $\mu_i(u) = \mathrm{val}^{\mathscr{G}_i}(u)$ for all $u \in S_i$ and $\mu_i(u) = \mathrm{val}_{\mathscr{G}}(u)$ for all $u \notin S_i$. We refer to [KBB+08] for the detailed proofs of the invariants. A careful analysis, using (minimum) Fibonacci heaps, as in Dijkstra's algorithm, allows one to obtain an overall complexity $O(m+n\log(n))$.

4.6 Total payoff games

Recall that the total payoff objective is defined by

$$\texttt{TotalPayoff}(\rho) = \limsup_k \sum_{i=0}^{k-1} \rho_i.$$

Contrary to the shortest path objective, total payoff games do not include a reachability objective. In particular, all plays will be infinite and their payoff is the superior limit of the partial sums: we need this superior limit since partial sums might not have a limit (consider for instance the sequence of weights $1,-1,1,-1,1,\ldots$ whose partial sums alternate between 1 and 0).

It is easy to show that solving total payoff games is in $\mathsf{NP} \cap \mathsf{coNP}$: they are positionally determined, and the one-player games can be solved in polynomial time using

Algorithm 4.6: A polynomial time algorithm for shortest path games with non-negative weights.

Data: A shortest path game with non-negative weights.

Function `Init()`**:**

- **for** $u \in V$ **do**
  - $\mu(u) \leftarrow \infty$
- **for** $u \in V_{\text{Min}}$ **do**
  - **if** $\exists u \xrightarrow{Win} v \in E$ **then**
    - $\mu(u) \leftarrow 0$
    - Add u to S
- **for** $u \in V_{\text{Max}}$ **do**
  - **if** $\forall u \xrightarrow{Win} v \in E$ **then**
    - $\mu(u) \leftarrow 0$
    - Add u to S
  - **else**
    - **for** $u \xrightarrow{w} v \in E$ **do**
      - $\mu(u \xrightarrow{w} v) \leftarrow \infty$

Function `Main()`**:**

- `Init` ()
- **repeat**
  - Extract $v \in \operatorname{argmin}\{\mu(v) : v \in S\}$
  - **for** $u \xrightarrow{w} v \in E$ **do**
    - **if** $u \in V_{\text{Min}}$ *and* $\mu(u) > \delta(\mu(v), w)$ **then**
      - $\mu(u) \leftarrow \delta(\mu(v), w)$
      - Update $\mu(u)$ in S
    - **if** $u \in V_{\text{Max}}$ *and* $\mu(u \xrightarrow{w} v) > \mu(v)$ **then**
      - $\mu(u \xrightarrow{w} v) \leftarrow \mu(v)$
      - $x \leftarrow \max\left\{\mu(u \xrightarrow{w'} v') : u \xrightarrow{w'} v' \in E\right\}$
      - **if** $\mu(u) > x$ **then**
        - $\mu(u) \leftarrow x$
        - Update $\mu(u)$ in S
- **until** *S is empty*
- **return** μ

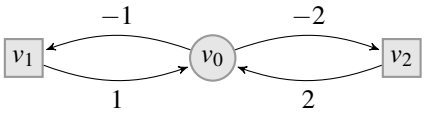


Figure 4.6: A total payoff game

shortest paths algorithms. However, constructing a value iteration is not easy. A natural attempt is to define the following operator:

$$\mathbb{O}^{\mathscr{G}}(\mu)(u) = \begin{cases} \max\left\{\mu(v)+w : u \xrightarrow{w} v \in E\right\} & \text{if } u \in V_{\text{Max}}, \\ \min\left\{\mu(v)+w : u \xrightarrow{w} v \in E\right\} & \text{if } u \in V_{\text{Min}}. \end{cases}$$

One can show that the value function is indeed a fixed point of $\mathbb{O}^{\mathscr{G}}$. But it is neither the greatest nor the least fixed point. Consider for example the total payoff game represented in Figure 4.6. The fixed points of $\mathbb{O}^{\mathscr{G}}$ are $(a, a+1, a+2)$ with $a \in \mathsf{R} \cup \{\pm\infty\}$: in particular, the greatest fixed point is $(+\infty, +\infty, +\infty)$ and the least fixed point is $(-\infty, -\infty, -\infty)$. However, the value function is $(0, 1, 2)$.

We obtain a pseudo-polynomial time algorithm by reduction to a shortest path game of pseudo-polynomial size.

Theorem 52 (A reduction from total payoff games to shortest path games)**.** *Let $\mathscr{G}$ a total payoff game, we can construct a shortest path game $\mathscr{G}'$ such that for all vertices u from $\mathscr{G}$, we have $val^{\mathscr{G}}(u) = val^{\mathscr{G}'}(u)$. The game $\mathscr{G}'$ has size pseudo-polynomial in the size of $\mathscr{G}$.*

Proof. Let us fix $K = n \cdot \big((2n-1) \cdot W + 1\big)$. The game $\mathscr{G}'$ consists in K consecutive copies of $\mathscr{G}$. In each copy, after each move, Min can offer Max the following alternative: either the game stops and reaches Win, or we move to the next copy. For the chosen K, one can show that the values of $\mathscr{G}$ coincide with the values of the first copy of $\mathscr{G}'$.

Instead of explicitly constructing $\mathscr{G}'$, we can run the value iteration algorithm for shortest path games in each copy, improving the space complexity of the algorithm. □

Bibliographic references

This chapter has been the occasion to start revealing a ladder of reductions going from parity games through mean payoff games and to discounted payoff games. In Chapter 6, the last reduction from discounted payoff games to simple stochastic games will complete this chain of reductions.

Mean payoff games. Mean payoff games have been first studied by Ehrenfeucht and Mycielski in [EM79] where positional determinacy is shown. The one-player variants had already been studied by Karp [Kar78]. It is much later that Zwick and Paterson [ZP96] first obtained the pseudo-polynomial value iteration algorithm to solve

them, while introducing discounted payoff games (that had been first studied in a probabilistic setting as will be studied in Chapter 5 and Chapter 6). The NP and coNP upper bound, together with strategy improvement algorithm, is due to Pure [Pur95].

The first-cycle games theorem is due to Aminof and Rubin [AR17], who fixed a flaw in the proof of Björklund, Sandberg and Vorobyov [BSV04] of positional determinacy for mean payoff games. The fairly mixing theorem is due to Gimbert and Zielonka [GZ04].

Strategy improvement algorithms. There are several strategy improvement algorithms for computing the mean payoff values. Björklund and Vorobyov [BV07] constructed one such algorithm for computing the energy values, presented via the notion of *longest shortest path problem*. A more direct approach was construct by Filar and Vrieze [FV96], involving a pair of values, called gain and bias. Strategy improvement methods (for parity games or payoff games) are very closely related to the simplex method for solving linear programs, see for instance [ABGJ14]. Lifshits and Pavlov [LP07] develop yet another strategy improvement algorithm for mean payoff games using potential reduction techniques, which were originally described by Gallai [Gal58] in the context of networks-related problems, and also developed by Gurvich, Karzanov, and Khachiyan [GKK88] in the context of mean payoff games. This has been further explored, see for instance [Ohl22].

Energy games.

Value iteration algorithm. The value iteration algorithm for energy games has been developed by Brim, Chaloupka, Doyen, Gentilini, and Raskin [BCD$^+$11]. Comin and Rizzi [CR17] have shown how to adapt the algorithm to compute optimal strategies as well with a running time $O(n^2mW)$. This removes the $\log(n)+\log(W)$ term due to binary search.

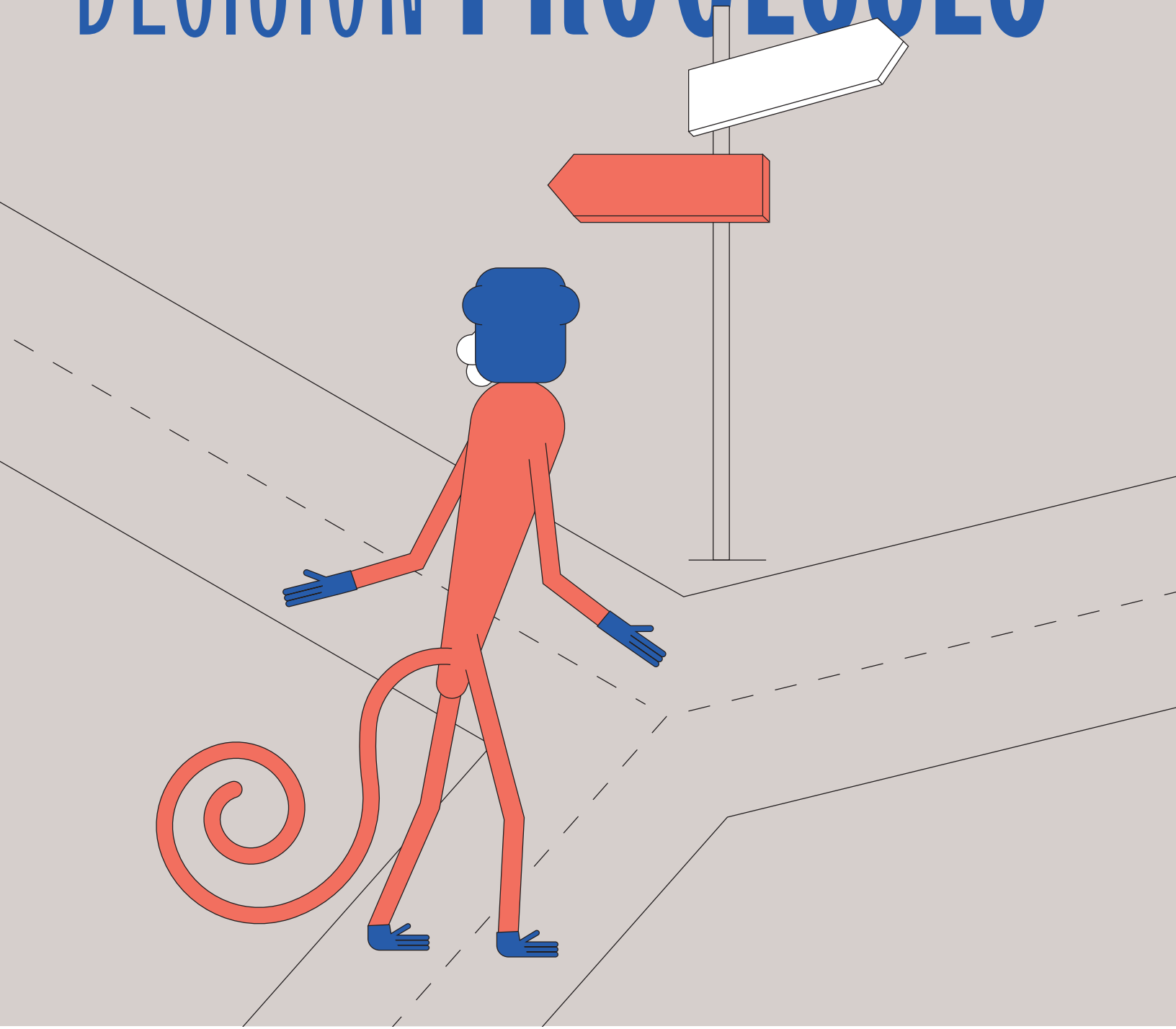
Shortest path games. As we have seen, computing the values for shortest path games can be done in polynomial time when weights are non-negative, this has been proved by Khachiyan and coauthors [KBB$^+$08]. No polynomial solution is known for the general case. The best known at the time this chapter is written is a polynomial time fragment consisting of *divergent shortest path games* [BGMR17] in which the arena does not contain any cycle with total weight 0: this a priori weak property indeed partitions the strongly connected components of the arena into the ones where all cycles are positive, and the ones where all cycles are negative; in each of these components, it is shown why the value iteration algorithm converges in polynomial time. Theorem 48 shows the equivalence between mean payoff games and detecting vertices of values $-\infty$ in shortest path games. It is open whether solving shortest path games where no vertices have value $-\infty$ can be done in polynomial time.

Total payoff games. The analysis of total payoff games and shortest path games is due to [BGHM17].

Part II

Stochastic

MARKOV DECISION PROCESSES



Chapter 5

Markov Decision Processes

PETR NOVOTNÝ

In this chapter we study Markov decision processes (MDPs), a standard model for decision making under uncertainty. MDPs are also called '$1\frac{1}{2}$-player games', since they can be viewed as a game in which only one player makes strategic choices, while the other player, which we call Nature, behaves according to some fixed probabilistic model. The chapter surveys the basic notions pertaining to MDPs, and algorithms for the following MDP-related problems:

- positive and almost-sure reachability and safety,
- discounted payoff,
- mean payoff in strongly connected MDPs,
- decomposition of MDPs into maximal end-components (MECs),
- reductions of general mean payoff MDPs, Büchi MDPs, and parity MDPs to general reachability MDPs (via the MEC decomposition),
- solving general reachability in MDPs.

Notations

We write vectors in boldface: $\vec{x}, \vec{y}$, etc. For a vector $\vec{x}$ indexed by a set I (i.e. $\vec{x} \in \mathbb{R}^I$) we denote by $\vec{x}_i$ the value of the component whose index is $i \in I$.

A (discrete) *probability distribution* over a finite or countably infinite set A is a function $f\colon A \to [0,1]$ such that $\sum_{a\in A} f(a) = 1$. The *support* of such a distribution f is the set of all $a \in A$ with $f(a) > 0$. A distribution f is called *Dirac* if its support has size 1. We denote by $\mathscr{D}(A)$ the set of all probability distributions over A.

We also deal with probabilities over uncountable sets of events. This is accomplished via the standard notion of a *probability space.*

Definition 8 (*Probability space*)**.** *A probability space is a triple* $(S, \mathscr{F}, \mathbb{P})$ *where*

- S *is a non-empty set of* events *(so called* sample space*).*
- $\mathscr{F}$ *is a* sigma-algebra *over S, i.e. a collection of subsets of S that contains the empty set* $\emptyset$ *and that is closed under complementation and countable unions. The members of* $\mathscr{F}$ *are called* $\mathscr{F}$-measurable sets*.*
- $\mathbb{P}$ *is a* probability measure *on* $\mathscr{F}$*, i.e. a function* $\mathbb{P}\colon \mathscr{F} \to [0,1]$ *such that:*
  1. $\mathbb{P}(\emptyset) = 0$*;*
  2. *for all* $A \in \mathscr{F}$ *it holds* $\mathbb{P}(S \setminus A) = 1 - \mathbb{P}(A)$*; and*
  3. *for all countable sequences of pairwise disjoint sets* $A_1, A_2, \cdots \in \mathscr{F}$ *(i.e.,* $A_i \cap A_j = \emptyset$ *for all* $i \neq j$*) we have* $\sum_{i=1}^{\infty} \mathbb{P}(A_i) = \mathbb{P}(\bigcup_{i=1}^{\infty} A_i)$*.*

A *random variable* in the probability space $(S, \mathscr{F}, \mathbb{P})$ is an $\mathscr{F}$-measurable function $X\colon \Omega \to \mathsf{R} \cup \{-\infty, \infty\}$, i.e., a function such that for every $a \in \mathsf{R} \cup \{-\infty, \infty\}$ the set $\{\omega \in \Omega \mid X(\omega) \leq a\}$ belongs to $\mathscr{F}$. We denote by $\mathbb{E}[X]$ the *expected value* of a random variable X (see Chapter 5 in [Bil95] for a formal definition).

We first give a syntactic notion of an MDP which is an analogue of the notion of an arena for games.

Definition 9 (*MDP*)**.** *A* Markov decision process *is a tuple* (V, E, Δ, c)*. The meaning of* V*,* E*, and* c *is the same as for games, i.e.* V *is a finite set of vertices,* $E \subseteq V \times V$ *is a set of edges and* $c\colon E \to C$ *a mapping of edges to a set of colours. However, the meaning of* Δ *is now different:* Δ *is a partial* probabilistic transition function *of type* $\Delta\colon V \times A \to \mathscr{D}(E)$*, such that the support of* $\Delta(v,a)$ *only contains edges outgoing from* v*. We usually write* $\Delta(v' \mid v, a)$ *as a shorthand for* $\Delta(v,a)((v,v'))$*, i.e. the probability of transitioning from* v *to* v' *under action* a*.*

We also stipulate that for each edge (v_1, v_2) there exists an action $a \in A$ such that $\Delta(v_2 \mid v_1, a) > 0$. Edges not satisfying this can be always removed without changing the semantics of the MDP, which is defined below. We denote by $p_{\min}$ the smallest non-zero edge probability in a given MDP, i.e. $p_{\min} = \min\{x > 0 \mid \exists u, v \in V, a \in A \text{ s.t. } x = \Delta(v \mid u, a)\}$.

We denote by E_{C} the set of edges coloured by C. Also, for MDPs where C is some set of numbers, we use $\max_c$ to denote the number $\max_{e \in E} |c(e)|$. In the setting of MDPs it is technically convenient to encode regular objectives (Reachability, Büchi,...) by colours on *vertices* as opposed to edges. Hence, when discussing these objectives, we assume that the colouring function c has the type $V \to C$.

Plays and strategies in MDPs The way in which a play is generated in an MDP is similar to games, but now encompasses a certain degree of randomness. There is a single player, say Max , who controls all the vertices. Max 's interaction with the 'world' described by an MDP is probabilistic. One reason is the stochasticity of the

transition function, the other is the fact that in MDP settings, it is usually permitted for Max to use *randomised strategies*. Formally, a randomised strategy is a function $\sigma : E^* \to \mathscr{D}(A)$, which to each finite play assigns a probability distribution over actions. We typically shorten $\sigma(\pi)(a)$ to $\sigma(a \mid \pi)$.

In this section, we will refer to randomised strategies simply as 'strategies'. The strategies known from the game setting will be called *deterministic strategies*. Formally, a deterministic strategy can be viewed as a special type of a randomised strategy which always selects a Dirac distribution over the edges. We shorten 'memoryless randomised/deterministic' to *MR* and *MD*, respectively.

Now a play in an MDP is produced as follows: in each step, when the finite play produced so far (i.e. the history of the game token's movement) is π, Max chooses an action a randomly according to the distribution $\sigma(\pi)$. Then, an edge outgoing from $\mathrm{last}(\pi)$ is chosen randomly according to $\Delta(\mathrm{last}(\pi), a)$ and the token is pushed along the selected edge. As shown below, this intuitive process can be formalized by constructing a special probability space whose sample space consists of infinite plays in the MDP.

Formal semantics of MDPs Formally, to each MDP $\mathscr{M}$, each (Max 's) strategy σ in $\mathscr{M}$, and each initial vertex v_0 we assign a probability space $(S_{\mathscr{M}}, \mathscr{F}_{\mathscr{M}}, \mathbb{P}^{\sigma}_{\mathscr{M},v_0})$. To explain the individual components, we need the notion of a cylinder set. A *basic cylinder* determined by a finite play π is the set of all infinite plays in $\mathscr{M}$ having π as a prefix. Now the above probability space consists of the following components:

- $S_{\mathscr{M}}$ is the set of all infinite plays in $\mathscr{M}$;
- $\mathscr{F}_{\mathscr{M}}$ is the *Borel* sigma-algebra over $\Omega_{\mathscr{M}}$; this is the smallest sigma-algebra containing all the basic cylinder sets determined by finite plays in $\mathscr{M}$. The sets in $\mathscr{F}_{\mathscr{M}}$ are called *events*. Note that the smallest sigma-algebra of the desired property is guaranteed to exist, since an intersection of an arbitrary number of sigma-algebras is again a sigma algebra.
- $\mathbb{P}^{\sigma}_{\mathscr{M},v_0}$ is the unique probability measure arising from the *cylinder construction* detailed below. We use $\mathbb{E}^{\sigma}_{\mathscr{M},v_0}$ to denote the *expectation* operator associated to the measure $\mathbb{P}^{\sigma}_{\mathscr{M},v_0}$.

Since the sample space $S_{\mathscr{M}}$ is uncountable, we construct the probability measure by first specifying a probability of certain simple sets of runs and then using an appropriate *measure-extension* theorem to extend the probability measure, in a unique way, to all sets in $\mathscr{F}_{\mathscr{M}}$. The standard cylinder construction proceeds as follows: for each finite play π we define the probability $p(\pi)$ such that

- for an empty play ε we put $p(\varepsilon) = 1$;
- for a non-empty play $\pi = \pi_0 \cdots \pi_k$ initiated in v_0 we put

$$p(\pi) = p(\pi_{<k}) \cdot \Big(\sum_{a \in A} \sigma(a \mid \pi_{<k}) \cdot \Delta(\mathrm{last}(\pi) \mid \mathrm{last}(\pi_{<k}), a) \Big),$$

  where we use the convention that $\mathrm{last}(\pi_{<0}) = v_0$;

- for all other π we have $p(\pi) = 0$.

Now using an appropriate measure-extension theorem (such as Hahn-Kolmogorov theorem [Ros06, Corollary 2.5.4 and Proposition 2.5.7], or Carathéodory theorem [ADD00, Theorem 1.3.10]) one can show that there is a unique probability measure $\mathbb{P}^{\sigma}_{\mathcal{M},v_0}$ on $\mathscr{F}_{\mathcal{M}}$ such that for every cylinder set $Cyl(\pi)$ determined by some finite play π we have $\mathbb{P}^{\sigma}_{v_0}(Cyl(\pi)) = p(\pi)$. (Abusing the notation, we write $\mathbb{P}^{\sigma}_{\mathcal{M},v_0}(\pi)$ for the probability of this cylinder set). There are some intermediate steps to be performed before an extension theorem can be applied, and we omit these due to space constraints. Full details on the cylinder construction can be found, e.g. in [ADD00, Nov15].

While the construction of the probability measure $\mathbb{P}^{\sigma}_{\mathcal{M},v_0}$ might seem a bit esoteric, in the context of MDP verification we do not usually need to be concerned with all the delicacies behind the associated probability space. The sets of plays that we work with typically arise from the basic cylinder sets by means of countable unions, intersections, and simple combinations thereof; such sets by definition belong to the sigma-algebra $\mathscr{F}_{\mathcal{M}}$, and their probabilities can be inferred using basic probabilistic reasoning. Nevertheless, one should keep in mind that all the probabilistic argumentation rests on solid formal grounds.

In the standard MDP literature [Put05], the plays are often defined as alternating sequence of vertices and actions. Here we stick to the edge-based definition inherited from deterministic games. Still, we would sometimes like to speak about quantities such as 'probability that action a is taken in step i'. To this end, we introduce, for each strategy σ, each action a, and each $i \geq 0$, a random variable $A^{\sigma}_{a,i}$ such that $A^{\sigma}_{a,i}(\pi) = \sigma(\pi_{<i})(a)$. It is easy to check that $\mathbb{E}^{\sigma}_{v}[A^{\sigma}_{a,i}]$ is the probability that action a is played in step i when using strategy σ.

Objectives in MDPs Similarly to plays, the notions of both qualitative and quantitative objectives are inherited from the non-stochastic world of games. However, since plays in MDPs are generated stochastically, even for a fixed strategy σ there is typically no single infinite play that would constitute the outcome of σ. A concrete σ might yield different outcomes, depending on the results of random events during the interaction with the MDP. Hence, we need a more general way of evaluating strategies in MDPs.

In the game setting, a qualitative objective was given as a set $\Omega \subseteq C^{\omega}$. In the MDP setting, we require that such Ω is measurable in the sense that the set $c^{-1}(\Omega) = \{\pi \in S_{\mathcal{M}} \mid c(\pi) \in \Omega\}$ belongs to $\mathscr{F}_{\mathcal{M}}$. We can then talk about a probability that the produced play satisfies Ω. For instance, for a colour $\mathtt{C}$ the objective $\mathtt{Reach}(\mathtt{C})$ is indeed measurable, since $c^{-1}(\Omega)$ can be written as a countable union of all basic cylinders that are determined by finite plays ending in a vertex coloured by $\mathtt{C}$. Indeed, all the qualitative objectives studied in previous chapters can be shown measurable in a similar way, and we encourage the reader to prove this as an exercise. Hence, the expression $\mathbb{P}^{\sigma}_{\mathcal{M},v_0}(\mathtt{Reach}(\mathtt{C}))$ denotes the probability that a vertex of colour $\mathtt{C}$ is reached when using strategy σ from vertex v_0. Naturally, Max aims at maximizing this probability. We refer to Max as "she", and in subsequent chapters when studying two-player stochastic games the opponent Min will be "he".

The situation is more complex for quantitative objectives. As shown in the previous chapter, when working with quantitative objectives, the set of colours C is typically the set of real numbers (or a subset thereof), and the quantitative objective is given by an 'aggregating function' $f\colon C^\omega \to \mathsf{R}$, which can be extended into a function $\bar{f}\colon E^\omega \to \mathsf{R}$ by putting $\bar{f}(\pi) = f(c(\pi_0)c(\pi_1)\cdots)$. In the MDP setting, we require that $\bar{f}$ is $\mathscr{F}_{\mathscr{M}}$-measurable, which means that for each $x \in \mathsf{R}$ the set $\{\pi \in E^\omega \mid f(c(\pi_0)c(\pi_1)\cdots) \leq x\}$ belongs to $\mathscr{F}_{\mathscr{M}}$ (again this holds for all the objectives studied in the previous chapters). Then there are two ways in which we can define the expected payoff achieved by strategy σ from a vertex v. First, we can treat $\bar{f}$ as a random variable in the probability space $(S_{\mathscr{M}}, \mathscr{F}_{\mathscr{M}}, \mathbb{P}^\sigma_{\mathscr{M},v})$. Then the *play-based payoff* of σ from v, which we denote by $\mathsf{p\text{-}Payoff}_f(v,\sigma)$, is the expected value of this random variable, i.e. $\mathsf{p\text{-}Payoff}_f(v,\sigma) = \mathbb{E}^\sigma_v[\bar{f}]$. That is, we compute the expected payoff over all plays. This approach subsumes also qualitative objectives: For such an objective Ω we can consider an *indicator* random variable $\mathbf{1}_\Omega$, such that $\mathbf{1}_\Omega(\pi) = 1$ of $\pi \in \Omega$ and $\mathbf{1}_\Omega(\pi) = 0$ otherwise. Then $\mathbb{P}^\sigma_{\mathscr{M},v}(\Omega) = \mathbb{E}^\sigma_{\mathscr{M},v}[\mathbf{1}_\Omega] = \mathsf{p\text{-}Payoff}_{\mathbf{1}_\Omega}(v,\sigma)$.

The second approach to quantitative objectives in MDPs, common e.g. in the operations research literature, is step-based: for each time step i we compute the expected one-step reward (i.e. colour) encountered in that step and then aggregate these one-step expectations. Formally, the *step-based payoff* of σ from v is $\mathsf{s\text{-}Payoff}_f(v,\sigma) = f(\mathbb{E}^\sigma_v[c(\pi_0)]\mathbb{E}^\sigma_v[c(\pi_1)]\cdots])$, where for each i we treat the expression $c(\pi_i)$ as a random variable returning the colour (i.e. a number) which labels the i-th edge of the randomly produced play (recall here that we index edges from 0).

Depending on the concrete quantitative objective and on the shape of σ, the path- and step-based payoffs from a given vertex might or might not be equal. Nevertheless, in this chapter we study only objectives for which these two semantics yield the *same optimization criteria:* no matter which of the two semantics we use, the optimal values will be the same and strategy that is optimal w.r.t. one of the semantics is also optimal for the other one. Hence, we will fix the play-based approach as the default one, writing just $\mathsf{Payoff}_f(v,\sigma)$ instead of $\mathsf{p\text{-}Payoff}_f(v,\sigma)$. We will prove the equivalence with step-based payoff where necessary. Also, we will drop the subscript f when the payoff function is known from the context.

Optimal strategies and decision problems Let us fix an MDP $\mathscr{M}$ and an objective given by a random variable f. The value of a vertex $v \in V$ is the number $\mathrm{val}(v) = \sup_\sigma \mathsf{Payoff}_f(v,\sigma)$. We let $\mathrm{val}(\mathscr{M})$ denote the $|V|$-dimensional vector whose component indexed by v equals $\mathrm{val}(v)$.

We say that a strategy σ is ε-optimal in v, for some $\varepsilon \geq 0$, if $\mathsf{Payoff}_f(v,\sigma) \geq \mathrm{val}(v) - \varepsilon$. A 0-optimal strategy is simply called optimal.

For qualitative objectives, there are additional modes of objective satisfaction. Given such an objective Ω, we say that a strategy σ is *almost-surely winning* from v if $\mathbb{E}^\sigma_{\mathscr{M},v}[\mathbf{1}_\Omega] = 1$, i.e. if the run produced by σ falls into Ω with probability 1. We also say that σ is *positively winning* from v if $\mathbb{E}^\sigma_{\mathscr{M},v}[\mathbf{1}_\Omega] > 0$. For strategies that are winning in the non-stochastic game sense, i.e. that *cannot* produce a run not belonging to Ω, are usually called *surely winning* to distinguish them from the above concepts. We denote by $W_{>0}(\mathscr{M},\Omega)$ and $W_{=1}(\mathscr{M},\Omega)$ the sets of all vertices of $\mathscr{M}$ from which there exists

a positively or almost-surely winning strategy for the objective Ω, respectively.

The problems pertaining to the existence of almost-surely or positively winning strategy are often called *qualitative problems* in the MDP literature, while the notion *quantitative problems* covers the general notion of optimizing the expectation of some random variable. We do not use such a nomenclature here so as to avoid confusion with qualitative vs. quantitative objectives as defined in Chapter 1. Instead, we will refer directly to, e.g. 'almost-sure reachability' while using the term 'optimal reachability' to refer to the expectation-maximization problem.

5.1 Positive and almost-sure reachability and safety in MDPs

We start our study of algorithmic problems for MDPs with the reachability objectives: we write $\texttt{Reach}(\mathrm{Win})$ for Win a set of colours. By a small abuse, we also write Win for the set of vertices of colours Win.

Positive reachability Analogously to attractor computations in reachability games (cf. Section 2.1), we define a one-step *positive probability* predecessor operator $\mathrm{Pre}_{>0}$ as follows: for $U \subseteq V$ we put

$$\mathrm{Pre}_{>0}(U) = \{v \in V \mid \exists a \in A, \exists u \in U : \Delta(u \mid v,a) > 0\}.$$

We define an operator $\mathscr{P}_{>0}$ on subsets of vertices: for $X \subseteq V$ we have

$$\mathscr{P}_{>0}(X) = \mathrm{Win} \cup \mathrm{Pre}_{>0}(X).$$

We note that this operator is monotonic when equipping the powerset of vertices with the inclusion preorder: if $X \subseteq X'$ then $\mathscr{P}_{>0}(X) \subseteq \mathscr{P}_{>0}(X')$. Hence Theorem 4 applies: this operator has a least fixed point computed by the following sequence: we let $X_0 = \emptyset$ and $X_i = \mathscr{P}_{>0}(X_{i-1})$. This constructs a sequence $(X_i)_{i\in\mathbb{N}}$ of non-decreasing subsets of V. Considering the graph induced by the MDP, it is see to see that X_i is the set of vertices from which a vertex of Win is reachable via a finite play of length at most $i-1$. Hence the sequence stabilises after at most $n-1$ steps.

We have the following simple characterization of the positively winning set:

Theorem 53 (Characterisation of the positively winning set)**.** *Let $\mathscr{M}$ an MDP. Then the positively winning region $W_{>0}(\mathscr{M}, \texttt{Reach}(Win))$ is the least fixed point of the operator $\mathscr{P}_{>0}$. Consequently, a vertex v belongs to $W_{>0}(\mathscr{M}, \texttt{Reach}(Win))$ if and only if there exists a (possibly empty) finite play from v to a vertex of colour Win. Moreover, there exists a uniform memoryless deterministic strategy that is positively winning from every vertex in $W_{>0}(\mathscr{M}, \texttt{Reach}(Win))$.*

Proof. Let us write $X_0 = \emptyset$ and $X_{i+1} = \mathscr{P}_{>0}(X_i)$, thanks to Theorem 4 this sequence of subsets of states converges to the least fixed point X_∞ of $\mathscr{P}_{>0}$. Since it is non-decreasing, the sequence is stationary and is reached after at most n iterations, meaning $X_n = X_\infty$. To simplify notations let us write $W_{>0} = W_{>0}(\mathscr{M}, \texttt{Reach}(\mathrm{Win}))$. We show two properties:

- For all i, we have $X_i \subseteq W_{>0}$.
- $W_{>0}$ is a pre-fixed point of $\mathscr{P}_{>0}$.

The first property implies that $X_\infty \subseteq W_{>0}$, and the second the converse implication.

We prove the first property by induction on i. The case $i = 0$ is clear. Let $v \in X_{i+1}$, either $v \in \mathrm{Win}$ and then it is in $W_{>0}$, or v in $\mathrm{Pre}_{>0}(X_i)$. Choosing the action witnessing that inclusion yields a positive probability to land in X_i. By induction hypothesis, $X_i \subseteq W_{>0}$. Hence we have constructed a strategy winning positively from v, implying that $v \in W_{>0}$.

We now prove the second property: $W_{>0} \subseteq \mathrm{Win} \cup \mathrm{Pre}_{>0}(W_{>0})$. Let $v \in W_{>0}$, by definition either $v \in \mathrm{Win}$ or there exists an action which yields a positive probability of winning, implying that $v \in \mathrm{Pre}_{>0}(W_{>0})$.

So far we have proved that $W_{>0}$ is the least fixed point of $\mathscr{P}_{>0}$. We conclude the proof by constructing a memoryless deterministic strategy σ positively winning from all of $W_{>0}$. For a vertex $v \in W_{>0}$, let $\mathrm{rank}(v)$ be the smallest i such that $v \in X_i$. For each $v \in W_{>0}$, either $v \in \mathrm{Win}$ or there exists an action a and vertex u such that $\Delta(u \mid v,a) > 0$ and $\mathrm{rank}(u) < \mathrm{rank}(v)$. Define $\sigma(v) = a$. A simple induction on the rank shows that σ is positively winning from every vertex of $W_{>0}$. □

As for complexity, we can focus on the problem of determining whether a given vertex belongs to $W_{>0}(\mathscr{M}, \mathtt{Reach}(\mathrm{Win}))$.

Corollary 12 (Complexity of deciding positive reachability)**.** *The problem of deciding whether a given vertex of a given MDP belongs to $W_{>0}(\mathscr{M}, \mathtt{Reach}(Win))$ is* NL*-complete. Moreover, the set $W_{>0}(\mathscr{M}, \mathtt{Reach}(Win))$ can be computed in linear time.*

Proof. Theorem 53 gives a blueprint for a logspace reduction from this problem to the *s-t-connectivity* problem for directed graphs, and vice versa. The latter problem is well known to be NL-complete [Sav70]. Moreover, the set of states from which a target colour is reachable can be computed by a simple graph search (e.g. by BFS), hence in linear time. □

Almost-sure safety We define the *almost-sure predecessor operator* $Pre_{=1}$: for $X \subseteq V$ we have

$$\mathrm{Pre}_{=1}(X) = \{v \in V \mid \exists a \in A, \forall t \in V : \Delta(t \mid v,a) > 0 \Rightarrow t \in X\}.$$

It is clearly monotonic: if $X \subseteq X'$, then $\mathrm{Pre}_{=1}(X) \subseteq \mathrm{Pre}_{=1}(X')$. One might be tempted to mimic the positive reachability case in order to solve the almost-sure reachability case: compute the least fixed point of the monotonic $X \mapsto \mathrm{Win} \cup \mathrm{Pre}_{=1}(X)$ and hope that it yields $W_{=1}(\mathscr{M}, \mathtt{Reach}(\mathrm{Win}))$. But this is not correct: consider an MDP with two vertices, u, v, the latter one being coloured by Win. We have only one action a: in v, the action self loops on v, while in u playing the action either moves us to v or leaves us in u, both options having probability $\frac{1}{2}$. The probability that we *never* reach v from u is equal to $\lim_{n\to\infty} \left(\frac{1}{2}\right)^n = 0$, and hence $W_{=1}(\mathscr{M}, \mathtt{Reach}(\mathrm{Win})) = \{u, v\}$. However, the least fixed of the operator above is $\{v\}$, excluding u. Note that there indeed exists

an infinite play which *can* be generated by the strategy and which never visits v, but the probability of generating such a play is 0.

Let us define the operator $\mathsf{P}_{=1}$ on subsets of vertices: for $X \subseteq V$ we have

$$\mathsf{P}_{=1}(X) = \mathrm{Win} \cap \mathrm{Pre}_{=1}(X).$$

Clearly $\mathsf{P}_{=1}$ is monotonic.

Lemma 49 (Characterization of the almost-sure safety winning region)**.** *Let $\mathscr{M}$ an MDP. Then the almost-sure safety winning region $W_{=1}(\mathscr{M}, \mathtt{Safe}(Win))$ is the greatest fixed point of the operator $\mathsf{P}_{=1}$. Moreover, there exists a memoryless deterministic strategy that is almost-surely winning from every vertex of $W_{=1}(\mathscr{M}, \mathtt{Safe}(Win))$.*

Proof. Let us write $X_0 = V$ and $X_{i+1} = \mathsf{P}_{=1}(X_i)$, thanks to Theorem 4 this sequence of subsets of states converges to the greatest fixed point X_∞ of $\mathsf{P}_{=1}$. Since it is non-increasing, the sequence is stationary and is reached after at most n iterations, meaning $X_n = X_\infty$. To simplify notations let us write $W_{=1} = W_{=1}(\mathscr{M}, \mathtt{Safe}(\mathrm{Win}))$. We show two properties:

- For all i, we have $W_{=1} \subseteq X_i$.
- $W_{=1}$ is a post-fixed point of $\mathsf{P}_{=1}$.

The first property implies that $W_{=1} \subseteq X_\infty$, and the second the converse implication.

We prove the first property by induction on i. The case $i = 0$ is clear. Let $v \in W_{=1}$. Clearly we must have that $v \in \mathrm{Win}$. Since $v \in W_{=1}$, there exists an action ensuring to stay in $W_{=1}$ with probability 1. By induction hypothesis $v \in X_i$, so this implies that $v \in \mathsf{P}_{=1}(X_i)$.

We now prove the second property: $W_{=1} \supseteq \mathrm{Win} \cap \mathrm{Pre}_{=1}(W_{=1})$. Let $v \in \mathrm{Win} \cap \mathrm{Pre}_{=1}(W_{=1})$, there exists an action ensuring to win with probability 1, implying that $v \in W_{=1}$.

So far we have proved that $W_{=1}$ is the greatest fixed point of $\mathsf{P}_{=1}$. We conclude the proof by constructing a memoryless deterministic strategy σ almost-surely winning from all of $W_{=1}$. For a vertex $v \in W_{=1}$, there exists an action a such that if $\Delta(u \mid v, a) > 0$ then $u \in W_{=1}$. Define $\sigma(v) = a$. Clearly σ is almost-surely winning from every vertex of $W_{=1}$. □

Algorithm 5.1: An algorithm computing $W_{=1}(\mathscr{M}, \mathtt{Safe}(\mathrm{Win}))$

Data: An MDP $\mathscr{M}$, a colour Win
$X \leftarrow V$;
repeat
  $X' \leftarrow X$;
  $X \leftarrow \mathrm{Win} \cap \mathrm{Pre}_{=1}(X)$
until $X' = X$;
return X

Corollary 13 (Complexity of the almost-sure safety winning set)**.** *The problem of deciding whether a given vertex of a given MDP belongs to $W_{=1}(\mathscr{M}, \texttt{Safe}(Win))$ can be solved in strongly polynomial time, together with a memoryless deterministic strategy which is almost-surely winning for all vertices in $W_{>0}(\mathscr{M}, \texttt{Safe}(Win))$.*

Algorithm 5.1 computes the set $W_{=1}(\mathscr{M}, \texttt{Safe}(\text{Win}))$ in strongly polynomial time.

Proof. The algorithm makes at most linearly many iterations, each of which has at most linear complexity. Hence, the complexity is at most quadratic. The strong polynomiality is testified by the fact that the algorithm only tests whether a probability of a given transition is positive or not, and the exact values of positive probabilities are irrelevant. □

Almost-sure reachability With almost-sure safety solved, we can now solve almost-sure reachability. Consider the one-step *almost-sure safety* operator $Safe_{=1}$ acting on sets of vertices:

$$Safe_{=1}(X) = X \cap \text{Pre}_{=1}(X).$$

This operator gives rise to the notion of a closed set, which is important for the study of safety objectives in MDPs.

Definition 10 (Closed set in an MDP)**.** *A set X of vertices is* closed *if $Safe_{=1}(X) = X$. The* sub-MDP *of an MDP $\mathscr{M}$ induced by the closed subset X is the MDP $\mathscr{M}_X = (X, E_X, \Delta_X, c_X)$ defined as follows:*

- *E_X is obtained from E by removing edges incident to a vertex from $V \setminus X$;*
- *Δ_X is obtained from $\Delta \subseteq V \times A \times \mathscr{D}(E)$ by removing all triples (v, a, f) where either $v \notin X$ or where the support of f is not contained in X;*
- *c_X is a restriction of c to X.*

We denote by $Cl(\mathscr{M})$ the set of all closed sets in $\mathscr{M}$.

One can show that a set is closed if Max has a strategy ensuring that she stays in the set forever.

Algorithm 5.2: An algorithm computing $W_{=1}(\mathscr{M}, \texttt{Reach}(\text{Win}))$

Data: An MDP $\mathscr{M}$, a colour Win
$W \leftarrow V$ **repeat**
  $W' \leftarrow W$ $Z \leftarrow W_{>0}(\mathscr{M}_W, \texttt{Reach}(\text{Win}))$ $W \leftarrow W_{=1}(\mathscr{M}_W, \texttt{Safe}(W \setminus Z))$
until $W' \neq W$;
return W ; `// A positive winning strategy in` $\mathscr{M}_W$ `is now almost-surely winning from` W `in` $\mathscr{M}$ `.`

The pseudocode for solving almost sure reachability is given in Algorithm 5.2. Note that in the first iteration, the algorithm computes the set $W_{=1}(\mathscr{M}, \texttt{Safe}(Z))$ where

$Z = W_{>0}(\mathscr{M}, \texttt{Reach}(\text{Win}))$. We might be tempted to think that this set already equals $W_{=1}(\mathscr{M}, \texttt{Reach}(\text{Win}))$, but this is not the case. To see this, consider an MDP with three states u, v, t and two actions a, b such that t is coloured by Win, both actions self loop in v and t, and $\Delta(t \mid u,a) = \Delta(v \mid u,a) = \frac{1}{2}$ while $\Delta(u \mid u,b) = 1$. Then $W_{=1}(\mathscr{M}, \texttt{Reach}(\text{Win})) = \{t\}$, but $W_{=1}(\mathscr{M}, \texttt{Safe}(W_{>0}(\mathscr{M}, \texttt{Reach}(\text{Win})))) = \{u,t\}$. However, iterating the computation works, as shown in the following theorem.

Theorem 54 (Algorithm for the almost-sure reachability winning set)**.** *The problem of deciding whether a given vertex of a given MDP belongs to $W_{=1}(\mathscr{M}, \texttt{Reach}(\mathit{Win}))$ can be solved in strongly polynomial time, together with a memoryless deterministic strategy which is almost-surely winning for all vertices in $W_{>0}(\mathscr{M}, \texttt{Reach}(\mathit{Win}))$.*

The pseudocode is given in Algorithm 5.2.

Proof. Since the set W can only decrease in each iteration, the algorithm terminates. We prove that upon termination, W equals $W_{=1}(\mathscr{M}, \texttt{Reach}(\text{Win}))$.

We start with the $\subseteq$ direction. We have $W \subseteq W_{>0}(\mathscr{M}_W, \texttt{Reach}(\text{Win}))$. By Theorem 53 there exists an MD strategy σ in $\mathscr{M}_W$ which is positively winning from each vertex of W. We show that the same strategy is also almost-surely winning from each vertex of W in $\mathscr{M}_W$ and thus also from each vertex of W in $\mathscr{M}$, which also proves the second part of the theorem. Let v be any vertex of W and denote $|W|$ by ℓ. Since σ is memoryless, it guarantees that a vertex of Win is reached with a positive probability in at most ℓ steps (see also the construction of σ in the proof of Theorem 53), and since it is also deterministic, it guarantees that the probability p of reaching Win in at most ℓ steps is at least $p_{\min}^{\ell}$, where $p_{\min}$ is the smallest non-zero edge probability in $\mathscr{M}_W$. Now imagine that ℓ steps have elapsed and we have not yet reached Win. This happens with a probability at most $(1 - p_{\min}^{\ell})$. However, even after these ℓ steps we are still in W, since σ is a strategy in $\mathscr{M}_w$. Hence, the probability that we do not reach Win within the first 2ℓ steps is bounded by $(1 - p_{\min}^{\ell})^2$. To realize why this is the case, note that any finite play π of length 2ℓ can be split into two halves, π', π'' of length ℓ, and then $\mathbb{P}_v^{\sigma}(\mathit{Cyl}(\pi)) = \mathbb{P}_v^{\sigma}(\mathit{Cyl}(\pi')) \cdot \mathbb{P}_{\text{last}(\pi')}^{\sigma}(\mathit{Cyl}(\pi''))$ (here we use the fact that σ is memoryless). Using this and some arithmetic, one can show that, denoting Avoid_i the set of all plays that avoid the vertices of Win in steps $\ell \cdot (i-1)$ to $\ell \cdot (i) - 1$, it holds

$$\mathbb{P}_v^{\sigma}(\mathit{Avoid}_1 \cap \mathit{Avoid}_2) \leq \mathbb{P}_v^{\sigma}(\mathit{Avoid}_1) \cdot \max_{u \in W \setminus \text{Win}} \mathbb{P}_u^{\sigma}(\mathit{Avoid}_1) \leq (1 - p_{\min}^{\ell})^2.$$

One can then continue by induction to show that $\mathbb{P}_v^{\sigma}(\bigcap_{i=1}^{j} \mathit{Avoid}_i) \leq (1 - p_{\min}^{\ell})^j$, and hence

$$\mathbb{P}_v^{\sigma}(\texttt{Reach}(\text{Win})) = 1 - \mathbb{P}_v^{\sigma}(\bigcap_{i=1}^{\infty} \mathit{Avoid}_i) \leq 1 - \lim_{j \to \infty} (1 - p_{\min}^{\ell})^j = 1 - 0 = 1.$$

Now we prove the $\supseteq$ direction. Denote $X = W_{=1}(\mathscr{M}, \texttt{Reach}(\text{Win}))$. We prove that $W \supseteq X$ is an invariant of the iteration. Initially this is clear. Now assume that this holds before an iteration takes place. It is easy to check that X is closed, so $\mathscr{M}_X$ is well-defined. We prove that $X \subseteq W_{=1}(\mathscr{M}_W, \texttt{Safe}(W \setminus Z))$, where Z is defined during the iteration. A strategy in $\mathscr{M}$ that reaches Win with probability 1 must never

visit a vertex from $V \setminus X$ with a positive probability. Hence, each such strategy can be viewed also as a strategy in $\mathscr{M}_X$. It follows that $X = W_{=1}(\mathscr{M}_X, \texttt{Reach}(\text{Win})) = W_{>0}(\mathscr{M}_X, \texttt{Reach}(\text{Win})) \subseteq W_{>0}(\mathscr{M}_W, \texttt{Reach}(\text{Win})) = Z$, the middle inclusion following from induction hypothesis. Now by Lemma 49 and Corollary 13, we have that the set $W_{=1}(\mathscr{M}_W, \texttt{Safe}(W \setminus Z))$ is the largest closed set contained in Z. But X is also closed, and as shown above, it is contained in Z. Hence, $X \subseteq W_{=1}(\mathscr{M}_W, \texttt{Safe}(W \setminus Z))$.

The complexity follows form Corollary 12 and Corollary 13; and also from the fact that the main loop must terminate in $\leq |V|$ steps. The strong polynomiality again follows from the algorithm being oblivious to precise probabilities. □

We also have a complementary hardness result.

Theorem 55 (Complexity of the almost-sure reachability winning set)**.** *The problem of determining whether a given vertex of a given MDP belongs to $W_{=1}(\mathscr{M}, \texttt{Reach}(\mathit{Win}))$ is* P*-complete.*

Proof. We proceed by a reduction from the *circuit value problem (CVP)*. An instance of *CVP* is a directed acyclic graph $\mathscr{C}$, representing a boolean circuit: each internal node represents either an OR gate or an AND gate, while each leaf node is labelled by *true* or *false*. Each internal node is guaranteed to have exactly two children. Each node of $\mathscr{C}$ evaluates to a unique truth value: the value of a leaf is given by its label and the value of an internal node v is given by applying the logical operator corresponding to the node to the truth values of the two children of v, the evaluation proceeding in a backward topological order. The task is to decide whether a given node w of $\mathscr{C}$ evaluates to *true*. *CVP* was shown to be P-hard (under logspace reductions) in [Lad75]. In [CDH10a], the following logspace reduction from CVP to almost-sure reachability in MDPs is presented: given a boolean circuit $\mathscr{C}$, construct an MDP $\mathscr{M}_{\mathscr{C}}$ whose vertices correspond to the gates of $\mathscr{C}$. There are two actions, call them *left* and *right*. In each vertex corresponding to an OR gate g, the *left* action transitions with probability 1 to the vertex representing the left child of g, and similarly for the action *right* and the right child. In a vertex corresponding to an AND gate g, both actions behave the same: the transition into each of the two children of g with probability $\frac{1}{2}$. Vertices corresponding to leafs have self loop as the only outgoing edges, and moreover, they are coloured with the respective labels in $\mathscr{C}$. It is easy to check that a gate of $\mathscr{C}$ evaluates to *true* if and only if the corresponding vertex belongs to $W_{=1}(\mathscr{M}_{\mathscr{C}}, \texttt{Reach}(\mathit{true}))$. □

Positive safety We conclude this section by a discussion of positive safety.

Theorem 56 (Algorithm for the positive safety winning set)**.** *Let $\mathscr{M}_{\bar{Win}}$ be an MDP obtained from $\mathscr{M}$ by changing all Win-coloured vertices to sinks (i.e. all actions in these vertices are self loops). Then*

$$W_{>0}(\mathscr{M}, \texttt{Safe}(\mathit{Win})) = W_{>0}(\mathscr{M}_{\bar{Win}}, \texttt{Reach}(W_{=1}(\mathscr{M}_{\bar{Win}}, \texttt{Safe}(\mathit{Win})))).$$

In particular, the set $W_{>0}(\mathscr{M}, \texttt{Safe}(\mathit{Win}))$ can be computed in a strongly polynomial time and there exists a memoryless deterministic strategy, computable in strongly polynomial time, that is positively winning from every vertex of $W_{>0}(\mathscr{M}, \texttt{Safe}(\mathit{Win}))$.

Proof. Clearly $W_{>0}(\mathscr{M}, \texttt{Safe}(\text{Win})) = W_{>0}(\mathscr{M}_{\bar{\text{Win}}}, \texttt{Safe}(\text{Win}))$ and $W_{=1}(\mathscr{M}, \texttt{Safe}(\text{Win})) = W_{=1}(\mathscr{M}_{\bar{\text{Win}}}, \texttt{Safe}(\text{Win}))$; and moreover the corresponding winning strategies easily transfer between the two MDPs (for a safety objective, the behaviour after visiting a Win-coloured state is inconsequential). Hence, putting $Z = W_{=1}(\mathscr{M}_{\bar{\text{Win}}}, \texttt{Safe}(\text{Win}))$, it is sufficient to show that $W_{>0}(\mathscr{M}_{\bar{\text{Win}}}, \texttt{Safe}(\text{Win})) = W_{>0}(\mathscr{M}_{\bar{\text{Win}}}, \texttt{Reach}(Z))$

The $\supseteq$ inclusion is clear as well as the construction of the witnessing MD strategy (in the vertices of that are outside of Z, we behave as the positively winning MD strategy for reaching Z, while inside Z we behave as the almost-sure winning strategy for $\texttt{Safe}(\text{Win})$).

For the other inclusion, let $X = V \setminus W_{>0}(\mathscr{M}_{\bar{\text{Win}}}, \texttt{Reach}(Z))$. We prove that $X \subseteq V \setminus W_{>0}(\mathscr{M}_{\bar{\text{Win}}}, \texttt{Safe}(\text{Win}))$. Assign ranks to vertices inductively as follows: each vertex coloured by Win gets rank 0. Now if ranks $\leq i$ have been already assigned, then a vertex v is assigned rank $i+1$ if it does not already have a lower rank but for all actions $a \in A$ there exists a vertex u of rank $\leq i$ s.t. $\Delta(u \mid v,a) > 0$. Then each vertex in X is assigned a finite rank: indeed, the set of vertices without a rank is closed and does not contain Win-coloured vertices, hence it is contained in Z. Now fix any strategy σ starting in a vertex $v \in X$. By definition of X, σ never reaches Z and hence never visits an unranked state. At the same time, whenever σ is in a ranked state, there is, by definition of ranks, a probability at least $p_{\min}$ (the minimal edge probability in $\mathscr{M}$) of transitioning to a lower-ranked state in the next step. Hence, in every moment, the probability of σ reaching a Win-coloured state within the next $|V|$ steps is at least $p_{\min}^{|V|}$. By a straightforward adaptation of the second part of the proof of Theorem 54, σ eventually visits Win with probability 1. Since σ was arbitrary, this shows that $v \notin W_{>0}(\mathscr{M}_{\bar{\text{Win}}}, \texttt{Safe}(\text{Win}))$.

The complexity follows from the results on positive reachability and almost-sure safety. □

5.2 Discounted payoff in MDPs

In this section, we consider MDPs with edges coloured by rational numbers and with the objective `DiscountedPayoff`. We start the chapter by proving that using the play-based semantics for the discounted payoff objective yields no loss of generality.

Lemma 50 (Equivalence of the definitions for discounted payoffs)**.** *In a discounted payoff MDP, for each strategy σ and each vertex v we have*

$$\mathit{p\text{-}Payoff}(v,\sigma) = \mathit{s\text{-}Payoff}(v,\sigma).$$

Proof. We have

$$\begin{aligned}\mathsf{p\text{-}Payoff}(v,\sigma) &= \mathbb{E}_v^\sigma[(1-\lambda)\lim_{k\to\infty}\sum_{i=0}^{k-1}\lambda^i c(\pi_i)] = (1-\lambda)\lim_{k\to\infty}\mathbb{E}_v^\sigma[\sum_{i=0}^{k-1}\lambda^i c(\pi_i)] \\ &= (1-\lambda)\cdot\lim_{k\to\infty}\sum_{i=0}^{k-1}\lambda^i\mathbb{E}_v^\sigma[c(\pi_i)] = \mathsf{s\text{-}Payoff}(v,\sigma).\end{aligned}$$

Here, the last equality on the first line follows from the dominated convergence theorem [ADD00, Theorem 1.6.9] and the following equality comes from the linearity of expectation. (To apply the domination convergence theorem, we use the fact that for each k we have $\texttt{DiscountedPayoff}^k(\pi) \leq \max_c$.) □

Optimal values and memoryless optimality

In this subsection we give a characterization of the value vector $\mathrm{val}(\mathscr{M})$ and prove that there always exists a memoryless deterministic strategy that is optimal in every vertex. Our exposition follows (in a condensed form) the one in [Put05], the techniques being somewhat similar to the ones in the previous chapter.

We define an operator $\mathbb{O}\colon \mathsf{R}^V \to \mathsf{R}^V$ as follows: each vector $\vec{x}$ is mapped to a vector $\vec{y} = \mathbb{O}(\vec{x})$ such that:

$$\vec{y}_v = \max_{a \in A} \sum_{u \in V} \Delta(u \mid v,a) \cdot ((1-\lambda) \cdot c(v,u) + \lambda \cdot \vec{x}_u)\,.$$

Lemma 51 (Contraction mapping)**.** *The operator $\mathbb{O}$ is a contraction mapping. Hence, $\mathbb{O}$ has a unique fixed point $\vec{x}^*$ in R^V, and $\vec{x}^* = \lim_{k\to\infty} \mathbb{O}^k(\vec{x})$, for any $\vec{x} \in \mathsf{R}^V$.*

Proof. The proof proceeds by a computation analogous to the one in the first half of the proof of Lemma 38; we just need to reason about actions rather than edges (and of course, use the formula defining $\mathbb{O}$ instead of the one for games). The second part follows from the Banach fixed point theorem. □

We aim to prove that $\mathrm{val}(\mathscr{M})$ is the unique fixed point $\vec{x}^*$ of $\mathbb{O}$. We start with an auxiliary definition.

Definition 11 (Safe actions)**.** *Let $\vec{x} \in \mathsf{R}^V$. We say that an action a is $\vec{x}$-*safe *in a vertex v if*

$$a = \arg\max_{a' \in A} \sum_{u \in V} \Delta(u \mid v,a') \cdot ((1-\lambda) \cdot c(v,u) + \lambda \cdot \vec{x}_u)\,. \tag{5.1}$$

A strategy σ is $\vec{x}$-safe, if for each play π ending in a vertex v, all actions that are selected with a positive probability by σ for π are $\vec{x}$-safe in v.

Given a strategy σ we define $\vec{x}^\sigma$ to be the vector such that $\vec{x}^\sigma_v = \mathsf{p\text{-}Payoff}(v,\sigma)$. For memoryless strategies, $\vec{x}^\sigma$ can be computed efficiently as follows: Each memoryless strategy σ defines a *linear* operator $\mathbb{O}^\sigma$ which maps each vector $\vec{x} \in \mathsf{R}^V$ to a vector $\vec{y} = \mathbb{O}^\sigma(\vec{x})$ such that

$$\vec{y}_v = \sum_{a \in A} \sigma(a \mid v) \cdot \left(\sum_{u \in V} \Delta(u \mid v,a) \cdot ((1-\lambda) \cdot c(v,u) + \lambda \cdot \vec{x}_u) \right).$$

Lemma 52 (Operator using a fixed strategy)**.** *Let σ be a memoryless strategy using rational probabilities. Then the operator $\mathbb{O}^\sigma$ has a unique fixed point, which is equal to $\vec{x}^\sigma$ and which can be computed in polynomial time.*

Proof. The operator $\mathbb{O}^\sigma$ can be seen as an instantiation of $\mathbb{O}$ in an MDP where there is no choice, since the action probabilities are chosen according to σ. Lemma 51 shows that $\vec{x}^\sigma$ is a fixed point of $\mathbb{O}^\sigma$. Since $\mathbb{O}^\sigma$ is again a contraction, it has a unique fixed point; and since it is linear, the fixed point can be computed in polynomial time, e.g. by Gaussian elimination (in its polynomial bit-complexity variant known as Bareiss algorithm [Bar68]). □

We now prove that there is a memoryless strategy ensuring outcome given by the fixed point of $\mathbb{O}$. Hence, the fixed point gives a lower bound on the values of vertices.

Lemma 53 (Any fixed point induces an MD strategy)**.** *Let $\vec{x}^*$ be the unique fixed point of $\mathbb{O}$. Then there exists an MD strategy that is $\vec{x}^*$-safe. Moreover, for each $\vec{x}^*$-safe memoryless strategy it holds that $\mathsf{p\text{-}Payoff}(v,\sigma) = \vec{x}^*_v$ for each $v \in V$.*

Proof. Note that for each $\vec{x} \in \mathsf{R}^V$ and each $v \in V$ there always exists at least one action that is $\vec{x}$-safe in v. Hence, there is a memoryless deterministic strategy which in each v chooses an arbitrary (but fixed) action that is $\vec{x}^*$-safe in v.

Now let σ be an arbitrary $\vec{x}^*$-safe memoryless strategy. By Lemma 52, the vector $\vec{x}^\sigma$ is the unique fixed point of $\mathbb{O}^\sigma$. But since σ is $\vec{x}^*$-safe, $\vec{x}^*$ is also a fixed point of $\mathbb{O}^\sigma$. Hence, $\vec{x}^* = \vec{x}^\sigma$. □

It remains to prove that $\vec{x}^*$ gives, for each vertex, an upper bound on the expected discounted payoff achievable by any strategy from that vertex. We introduce some additional notation to be used in the proof of the next lemma, as well as in some later results: namely, we denote by $\texttt{DiscountedPayoff}^{(k)}(\pi)$ the discounted payoff accumulated during the first k steps of π, i.e. the number $(1-\lambda)\sum_{i=0}^{k-1} \lambda^i c(\pi_i)$. The following lemma can be proved by an easy induction.

Lemma 54 (Properties of the sequence of iterates)**.** *For each $k \geq 0$ and each vertex v we have*

$$\sup_\sigma \mathbb{E}^\sigma_v[\texttt{DiscountedPayoff}^{(k)}] = (\mathbb{O}^k(\vec{0}))_v$$

(here $\vec{0}$ is a $|V|$-dimensional vector of zeroes).

The previous lemma is used to prove the required upper bound on $\mathrm{val}(v)$.

Lemma 55 (Upper bound on the optimal value)**.** *For each vertex v it holds $val(v) \leq \vec{x}^*_v$, where $\vec{x}^*$ is the unique fixed point of $\mathbb{O}$.*

Proof. We have $\texttt{DiscountedPayoff}(\pi) = \lim_{k\to\infty} \texttt{DiscountedPayoff}^{(k)}(\pi)$ (for each π) and hence, by dominated convergence theorem we have $\mathbb{E}^\sigma_v[\texttt{DiscountedPayoff}] = \lim_{k\to\infty} \mathbb{E}^\sigma_v[\texttt{DiscountedPayoff}^{(k)}]$. Hence,

$$\begin{aligned} \mathrm{val}(v) &= \sup_\sigma \mathbb{E}^\sigma_v[\texttt{DiscountedPayoff}] \\ &= \sup_\sigma \lim_{k\to\infty} \mathbb{E}^\sigma_v[\texttt{DiscountedPayoff}^{(k)}] \end{aligned} \tag{5.2}$$

It remains to prove that the RHS of Equation (5.2) is not greater than $\vec{x}^* = \lim_{k\to\infty} \mathbb{O}^k(\vec{0}) = \lim_{k\to\infty} \sup_\sigma \mathbb{E}^\sigma_v[\texttt{DiscountedPayoff}^{(k)}]$ (the last inequality follows by Lemma 54).

It suffices to show that for each σ' we have $\lim_{k\to\infty} \mathbb{E}_v^{\sigma'}[\texttt{DiscountedPayoff}^{(k)}] \leq \lim_{k\to\infty} \sup_\sigma \mathbb{E}_v^{\sigma}[\texttt{DiscountedPayoff}^{(k)}]$. But this immediately follows from the fact that the inequality holds for each concrete k. □

The following theorem summarizes the results.

Theorem 57 (Characterisation of the optimal values)**.** *The vector of values $val(\mathscr{M})$ in a discounted sum MDP $\mathscr{M}$ is the unique fixed point $\vec{x}^*$ of the operator $\mathbb{O}$. Moreover, there exists a memoryless deterministic strategy that is optimal in every vertex.*

Proof. The characterisation of $\mathrm{val}(\mathscr{M})$ follows directly from Lemmata 53 and 55. The MD optimality follows from Lemma 53. □

In the rest of this section we discuss several algorithms for computing the values and optimal strategies in discounted payoff MDPs.

Value iteration

The value iteration algorithm works in the same way as in the case of discounted payoff games: we simply iterate the operator $\mathbb{O}$. We know that $\mathrm{val}(\mathscr{M}) = \lim_{k\to\infty} \mathbb{O}^k(\vec{0})$, and hence, iterating $\mathbb{O}$ yields an approximation of $\mathrm{val}(\mathscr{M})$. The iteration might not reach the fixed point (i.e. $\mathrm{val}(\mathscr{M})$) in a finite number of steps, but we can provide simple bounds on the precision of the approximation.

Lemma 56 (Properties of the iterates)**.** *For each $k \in \mathbb{N}$, $\|val(\mathscr{M}) - \mathbb{O}^k(\vec{0})\|_\infty \leq \lambda^k \cdot \max_c$.*

Proof. This follows immediately from Lemma 54 and from the fact that for each play π, $\sum_{i=k}^{\infty} \lambda^i \cdot c(\pi_i) \leq \frac{1}{1-\lambda} \cdot \lambda^k \cdot \max_c$. □

Similar analysis can be applied to strategies induced by the approximate vectors.

Lemma 57 (Relating iterates and optimal strategies)**.** *Let $\varepsilon > 0$ be arbitrary and let*

$$k(\varepsilon) = \left\lceil \frac{\log_2\left(\frac{\varepsilon(1-\lambda)}{4\max_c}\right)}{\log_2(\lambda)} \right\rceil .$$

Then, every $\mathbb{O}^{k(\varepsilon)}(\vec{0})$-safe memoryless strategy is ε-optimal in every vertex.

Proof. Let σ be any $\mathbb{O}^{k(\varepsilon)}(\vec{0})$-safe memoryless strategy and let $\mathbb{O}^\sigma$ be the corresponding operator. We have that

$$\begin{aligned} \|\mathrm{val}(\mathscr{M}) - \vec{x}^\sigma\|_\infty &= \|\mathrm{val}(\mathscr{M}) - \mathbb{O}^{k(\varepsilon)}(\vec{0}) + \mathbb{O}^{k(\varepsilon)}(\vec{0}) - \vec{x}^\sigma\|_\infty \\ &\leq \|\mathrm{val}(\mathscr{M}) - \mathbb{O}^{k(\varepsilon)}(\vec{0})\|_\infty + \|\mathbb{O}^{k(\varepsilon)}(\vec{0}) - \vec{x}^\sigma\|_\infty. \end{aligned} \tag{5.3}$$

The first term in Equation (5.3) is $\leq \varepsilon/2$ by the choice of $k(\varepsilon)$ and Lemma 56. We prove that the second term in Equation (5.3) is also bounded by $\varepsilon/2$. Note that we have

$\vec{x}^\sigma = \mathbb{O}^\sigma(\vec{x}^\sigma)$ (as was already proved in Lemma 53) and $\mathbb{O}(\mathbb{O}^{k(\varepsilon)}(\vec{0})) = \mathbb{O}^\sigma(\mathbb{O}^{k(\varepsilon)}(\vec{0}))$ (since σ is $\mathbb{O}^{k(\varepsilon)}(\vec{0})$-safe). Using this we get

$$\begin{aligned}
\|\mathbb{O}^{k(\varepsilon)}(\vec{0}) - \vec{x}^\sigma\|_\infty &= \|\mathbb{O}^{k(\varepsilon)}(\vec{0}) - \mathbb{O}^{k(\varepsilon)+1}(\vec{0}) + \mathbb{O}^{k(\varepsilon)+1}(\vec{0}) - \vec{x}^\sigma\|_\infty \\
&\leq \|\mathbb{O}^{k(\varepsilon)}(\vec{0}) - \mathbb{O}^{k(\varepsilon)+1}(\vec{0})\|_\infty + \|\mathbb{O}^{k(\varepsilon)+1}(\vec{0}) - \vec{x}^\sigma\|_\infty \\
&= \|\mathbb{O}^{k(\varepsilon)}(\vec{0}) - \mathbb{O}^{k(\varepsilon)+1}(\vec{0})\|_\infty + \|\mathbb{O}^\sigma(\mathbb{O}^{k(\varepsilon)}(\vec{0})) - \mathbb{O}^\sigma(\vec{x}^\sigma)\|_\infty \\
&\leq \|\mathbb{O}^{k(\varepsilon)}(\vec{0}) - \mathbb{O}^{k(\varepsilon)+1}(\vec{0})\|_\infty + \lambda \cdot \|(\mathbb{O}^{k(\varepsilon)}(\vec{0})) - \vec{x}^\sigma\|_\infty
\end{aligned}$$

Re-arranging yields $\|\mathbb{O}^{k(\varepsilon)}(\vec{0}) - \vec{x}^\sigma\|_\infty \leq \frac{1}{1-\lambda} \cdot \|\mathbb{O}^{k(\varepsilon)}(\vec{0}) - \mathbb{O}^{k(\varepsilon)+1}\|_\infty$.

It follows from Lemma 56 that $\|\mathbb{O}^{k(\varepsilon)}(\vec{0}) - \mathbb{O}^{k(\varepsilon)+1}(\vec{0})\|_\infty \leq 2 \cdot \lambda^{k(\varepsilon)} \cdot \max|c| \leq \frac{(1-\lambda)\varepsilon}{2}$, the last inequality holding by the choice of $k(\varepsilon)$. Plugging this into the formula above yields $\|\mathbb{O}^{k(\varepsilon)}(\vec{0}) - \vec{x}^\sigma\|_\infty \leq \frac{\varepsilon}{2}$, as required. □

Using a value-gap result (similar to the game case, but proved using a different technique), one can show that sufficiently precise iterates can be used to compute an *optimal* strategy. This is summarized in the following lemma due to [Tse90].

Lemma 58 (Sufficiently many iterates yield an optimal strategy)**.** *Let d be the least common multiple of denominators of the following numbers: λ, all transition probabilities, and all edge colourings in $\mathscr{M}$. Next, let $\varepsilon^* = \frac{1}{d^{(3|V|+3)} \cdot |V|^{\frac{V}{2}}}$. Then, any $\mathbb{O}^{k(\varepsilon^*)}(\vec{0})$-safe memoryless deterministic strategy is optimal in every vertex.*

Proof. Let σ^* be any MD optimal strategy (it is guaranteed to exist by Theorem 57). By the same theorem, we have that $\mathrm{val}(\mathscr{M}) = \mathbb{O}^\sigma(\mathrm{val}(\mathscr{M}))$. By the definition of $\mathbb{O}^\sigma$, we can write the above equation as $\mathrm{val}(\mathscr{M}) = (1-\lambda) \cdot \vec{c} + \lambda \cdot P \cdot \mathrm{val}(\mathscr{M})$, where $\vec{c}$ is a vector whose each component is a sum of several terms, each term being a product of an edge colour and of a transition probability; and P is a matrix containing transition probabilities. Multiplying the equation by d^3 yields $d^3\mathrm{val}(\mathscr{M}) = d^3(1-\lambda) \cdot \vec{c} + d^3\lambda \cdot P \cdot \mathrm{val}(\mathscr{M})$. Since this equation has a unique fixed point (due to $\mathbb{O}^\sigma$ being a contraction), the matrix $A = d^3(I - \lambda P)$ (where I is the unit matrix) is regular, and moreover, composed of integers (ue to the choice of d). By Cramer's rule, each entry of $\mathrm{val}(\mathscr{M})$ is equal to $\det(B)/\det(A)$, where B is a matrix obtained by replacing some column of A with $d^3(1-\lambda)\vec{c}$ (which is again an integer vector, due to the multiplication by d^3). Hence, each entry of $\mathrm{val}(\mathscr{M})$ is a rational number with denominator $\det(A)$. Hadamard's inequality [Gar07] implies $|\det(A)| \leq d^{3|V|}|V|^{\frac{|V|}{2}}$.

Now let σ be any $\mathbb{O}^{k(\varepsilon^*)}(\vec{0})$-safe MD strategy. By Lemma 57, σ is ε^*-optimal. We prove that all actions used by σ are $\vec{x}^*$-safe, which means that σ is optimal by Lemma 53. Assume that in some vertex v the strategy σ uses action a that is not $\vec{x}^*$-safe. Denote $\vec{y} = \mathbb{O}^\sigma(\vec{x}^*)$. We have $|\vec{y}_v - \vec{x}^*_v| > 0$, since otherwise a would be $\vec{x}^*$-safe. But then we can obtain a lower bound on the difference by investigating the bitsize of the numbers

involved:

$$\begin{aligned}
|\vec{y}_v - \vec{x}^*_v| &= \left| \frac{d^3}{d^3}\vec{y}_v - \frac{d^3}{d^3}\vec{x}^*_v \right| \\
&= \frac{1}{d^3} \left| \sum_{u \in V} \underbrace{d \cdot \Delta(u \mid v,a)}_{\text{integer}} \cdot \Big(\underbrace{d^2(1-\lambda) \cdot c(u,v)}_{\text{integer}} + \underbrace{d^2 \cdot \lambda \cdot \vec{x}^*_u) - d^3 \vec{x}^*}_{\text{int. multiples of } 1/\det(A)} \right| \\
&\geq \frac{1}{d^3 \cdot \det(A)} \geq \frac{1}{d^{(3|V|+3)} \cdot |V|^{\frac{|V|}{2}}}.
\end{aligned}$$

Now put $\vec{z} = \mathbb{O}^\sigma(\mathbb{O}^{k(\varepsilon)}(\vec{0}))$. We have the following (using, on the first line, the fact that $|a+b| \geq |a| - |b|$):

$$\begin{aligned}
|\vec{z}_v - \vec{x}^*_v| &= |\vec{z}_v - \mathbb{O}^\sigma(\vec{x}^*)_v + \mathbb{O}^\sigma(\vec{x}^*)_v - \vec{x}^*_v| \geq |\mathbb{O}^\sigma(\vec{x}^*)_v - \vec{x}^*_v| - |\vec{z}_v - \mathbb{O}^\sigma(\vec{x}^*)_v| \\
&\geq \frac{1}{d^{(3|V|+3)} \cdot |V|^{\frac{|V|}{2}}} - |\mathbb{O}^\sigma(\mathbb{O}^{k(\varepsilon)}(\vec{0}))_v - \mathbb{O}^\sigma(\vec{x}^*)_v| \quad \text{(as shown above)} \\
&\geq \frac{1}{d^{(3|V|+3)} \cdot |V|^{\frac{|V|}{2}}} - |\mathbb{O}^\sigma(\mathbb{O}^{k(\varepsilon^*)}(\vec{0}) - \vec{x}^*)_v| \quad \text{(since } \mathbb{O}^\sigma \text{ is linear)} \\
&\geq \varepsilon^* - \lambda \cdot \|\mathbb{O}^{k(\varepsilon^*)}(\vec{0}) - \vec{x}^*\|_\infty > \varepsilon^* - \frac{\varepsilon^*}{2} \quad \text{(Lemma 56)} \\
&= \frac{\varepsilon^*}{2}.
\end{aligned}$$

In particular, it must hold that $\vec{z}_v < \vec{x}^*_v$. Otherwise we would have $\mathbb{O}^{k(\varepsilon^*)+1}(\vec{0})_v \geq \mathbb{O}^\sigma(\mathbb{O}^{k(\varepsilon^*)}(\vec{0}))_v \geq \vec{x^*}_v + \frac{\varepsilon^*}{2}$, a contradiction with $\mathbb{O}^{k(\varepsilon^*)+1}(\vec{0})$ being an $\frac{\varepsilon^*}{4}$-approximation of $\vec{x}^*$ (by Lemma 56 and the choice of $k(\varepsilon^*)$).

At the same time, $|\mathbb{O}(\mathbb{O}^{k(\varepsilon^*)}(\vec{0}))_v - \vec{x}^*| \leq \frac{\varepsilon^*}{4}$, due to the choice of $k(\varepsilon^*)$. Since $\vec{z}_v \leq \vec{x}^*_v$, we get $\vec{z}_v < \vec{x}^*_v - \frac{\varepsilon}{2} \leq \mathbb{O}(\mathbb{O}^{k(\varepsilon^*)}(\vec{0}))_v$, a contradiction with σ being $\mathbb{O}^{k(\varepsilon^*)}(\vec{0})$-safe. □

Corollary 14 (Complexity of discounted payoff MDPs with a fixed discount factor)**.** *An optimal MD strategy in discounted payoff MDPs with a fixed discount factor can be computed in polynomial time.*

Proof. The number $1/\varepsilon^*$, where ε^* is from Lemma 58, has bitsize polynomial in the size of the MDP when the discount factor is fixed. Hence, the number $k(\varepsilon^*)$ defined as in Lemma 57 has a polynomial magnitude, so it suffices to perform only polynomially many iterations. Since each iteration requires polynomially many arithmetic operations that involve only summation and multiplication by a constant, the result follows. □

Strategy improvement, linear programming, and (strongly) polynomial time

The strategy (or policy) improvement (also called strategy/policy iteration in the literature) for MDPs works similarly as for games, see Algorithm 5.3. In the algorithm, we

use $\mathbb{O}_{a,v}(\vec{x})$ as a shortcut for $\sum_{u \in V} \Delta(u \mid v,a)\,((1-\lambda)\cdot c(v,u)+\lambda\cdot\vec{x}_u)$. The computation of $\vec{x}^{\sigma_i}$ uses Lemma 52.

Algorithm 5.3: An algorithm computing an optimal MD strategy in a discounted MDP

Data: A discounted payoff MDP $\mathscr{M}$
$i \leftarrow 0$;
$\sigma_i \leftarrow$ arbitrary MD strategy;
repeat
  compute $\vec{x}^{\sigma_i} = \left(\mathbb{E}_v^{\sigma_i}[\texttt{DiscountedPayoff}]\right)_{v \in V}$;
  foreach $v \in V$ **do**
    $\mathit{Improve}(v) \leftarrow \sigma_i(v)$;
    foreach $a \in A$ **do**
      if $\mathbb{O}_{a,v}(\vec{x}^{\sigma_i}) > \mathbb{O}_{a,\mathit{Improve}(v)}(\vec{x}^{\sigma_i})$ **then** $\mathit{Improve}(v) \leftarrow a$;
    $\sigma_{i+1}(v) \leftarrow \mathit{Improve}(v)$
  $i \leftarrow i+1$
until $\sigma_i = \sigma_{i-1}$;
return σ_i

Theorem 58 (Strategy improvement for discounted MDPs)**.** *The strategy improvement algorithm for discounted MDPs terminates in a finite (and at most exponential) number of steps and returns an optimal MD strategy.*

Proof. First we need to show that whenever $\sigma_{i+1} \neq \sigma_i$, then $\vec{x}^{\sigma_{i+1}} \geq \vec{x}^{\sigma_i}$ and $\vec{x}^{\sigma_{i+1}} \neq \vec{x}^{\sigma_i}$ (we write this by $\vec{x}^{\sigma_{i+1}} \succ \vec{x}^{\sigma_i}$). So fix some i s.t. an improvement is performed in the i-th iteration of the repeat-loop. We have $\mathbb{O}_{\sigma_{i+1}}(\vec{x}^{\sigma_i}) \succ \mathbb{O}_{\sigma_i}(\vec{x}^{\sigma_i}) = \vec{x}^{\sigma_i}$, i.e. $\mathbb{O}_{\sigma_{i+1}}(\vec{x}^{\sigma_i}) - \vec{x}^{\sigma_i} \succ 0$. Let P, $\vec{c}$ be the matrix and vector such that the equation $\vec{x} = \mathbb{O}_{\sigma_{i+1}}(\vec{x})$ can be written as $\vec{x} = (1-\lambda)\cdot\vec{c} + \lambda\cdot P\cdot\vec{x}$. Since the equation $\vec{x} = \mathbb{O}_{\sigma_{i+1}}(\vec{x})$ has a unique fixed point (as $\mathbb{O}_{\sigma_{i+1}}$ is a contraction), the matrix $I-\lambda P$ is invertible. Then $\mathbb{O}_{\sigma_{i+1}}(\vec{x}^{\sigma_i}) - \vec{x}^{\sigma_i} \succ 0$ can be written as $(1-\lambda)\vec{c} + (\lambda P - I)\vec{x}^{\sigma_i} \succ 0$, or equivalently $\vec{x}^{\sigma_i} \prec (1-\lambda)\vec{c}\cdot(I - \lambda P)^{-1}$. But the RHS of this inequality is equal to the fixed point of $\mathbb{O}_{\sigma_{i+1}}$, i.e. to $\vec{x}^{\sigma_{i+1}}$.

Now there are only finitely (exponentially) many MD strategies and since $\vec{x}^{\sigma_{i+1}} \succ \vec{x}^{\sigma_i}$, we have that no strategy is considered twice. Hence, the algorithm eventually terminates. Upon termination, there is no improving action, so the algorithm has found a strategy σ whose value vector $\vec{x}^{\sigma}$ is the fixed point of $\mathbb{O}$. Such a strategy is optimal by Theorem 57. □

While each of the steps (1.)–(4.) can be performed in polynomial time, the worst-case number of iterations is exponential [HDJ12]. However, the algorithm has nice properties in the case when the discount factor is fixed, as we'll see below. It is also intimately connected to the linear programming approach.

We can indeed aim to directly solve the equation $\vec{x} = \mathbb{O}(\vec{x})$, thus obtaining the fixed point of $\mathbb{O}$, by using a suitable LP. While the operator $\mathbb{O}$ is not in itself linear, solving the equation can be encoded into a linear program. The main idea can be described as follows: given some numbers y, z, the solution $\bar{x}$ to the equation $x = \max\{y, z\}$ is

exactly the smallest solution to the set of inequalities $x \geq y$, $x \geq z$. Hence, to solve the equation $\vec{x} = \mathbb{O}(\vec{x})$, we set up the following linear program $\mathscr{L}_{disc}$ with variables x_v, $v \in V$.

$$\begin{array}{ll} \text{minimize} & \sum_{v \in V} x_v \\ \text{subject to} & x_v \geq \sum_{u \in V} \Delta(u \mid v,a) \cdot ((1-\lambda) \cdot c(v,u) + \lambda \cdot x_u) \quad \text{for all } v \in V \text{ and } a \in A. \end{array}$$

Lemma 59 (Properties of the linear program)**.** *The linear program $\mathscr{L}_{disc}$ has a unique optimal solution $\vec{\bar{x}}$ that satisfies $\vec{\bar{x}} = val(\mathscr{M})$.*

Proof. Let $\vec{x}^* = \text{val}(\mathscr{M})$ be the unique fixed point of $\mathbb{O}$. Clearly setting $x_v = \vec{x}^*_v$ yields a feasible solution of $\mathscr{L}_{disc}$. We show that $\vec{x}^*$ is actually an optimal solution, by proving that for each feasible solution $\vec{x}$ it holds $\vec{x} \geq \vec{x}^*$. (This also shows the uniqueness, since it says that an optimal solution is the infimum of the set of all feasible solutions.) First, note that for any feasible solution $\vec{x}$ it holds $\mathbb{O}(\vec{x}) \leq \vec{x}$, by the construction of $\mathscr{L}_{disc}$. Next, if $\vec{x}$ is a feasible solution, then $\mathbb{O}(\vec{x})$ is also a feasible solution; otherwise, there would be some v and $a \in A$ such that

$$\begin{aligned} \mathbb{O}(\vec{x})_v &< \sum_{u \in V} \Delta(u \mid v,a) \cdot ((1-\lambda) \cdot c(v,u) + \lambda \cdot \mathbb{O}(\vec{x})_u) \\ &\leq \sum_{u \in V} \Delta(u \mid v,a) \cdot ((1-\lambda) \cdot c(v,u) + \lambda \cdot \vec{x}_u) \leq \mathbb{O}(\vec{x})_v. \end{aligned}$$

Here, the first inequality on the second line follows from $\mathbb{O}(\vec{x}) \leq \vec{x}$, while the second inequality follows from the definition of $\mathbb{O}$. But $\mathbb{O}(\vec{x})_v < \mathbb{O}(\vec{x})_v$ is an obvious contradiction. So $\mathbb{O}(\vec{x})$ is indeed a feasible solution and by applying the argument above, we get $\mathbb{O}^2(\vec{x}) \leq \mathbb{O}(\vec{x})$. By a simple induction, $\mathbb{O}^{i+1}(\vec{x}) \leq \mathbb{O}^i(\vec{x}) \leq \vec{x}$ for each $i \geq 0$. Hence, also $\vec{x}^* = \lim_{i \to \infty} \mathbb{O}^i(\vec{x}) \leq \vec{x}$ (the first equality comes from Lemma 51). □

Linear programming can be solved in polynomial time (Theorem 2). Hence, we get the following.

Theorem 59 (Properties of discounted payoff MDPs)**.** *The following holds for discounted payoff MDPs:*

1. *The value of each vertex as well as an MD optimal strategy can be computed in polynomial time.*
2. *The problem whether the value of a given vertex v is at least a given constant (say 1) is* P*-complete (under logspace reductions). The hardness result holds even for a fixed discount factor.*

Proof. (1.) The first part comes directly from Lemma 59. Once the optimal value vector $\text{val}(\mathscr{M})$ is computed, we can choose any $\text{val}(\mathscr{M})$-safe MD strategy as the optimal one (Lemma 53).

(2.) Let λ be the fixed discount factor. We show the lower bound, by extending the reduction from the CVP problem used for almost-sure reachability. First, we transform the input circuit into an MDP in the same way as in the reachability case, and we let

v be the vertex corresponding to a gate we wish to evaluate. Assume, for a while, that each path from v to a target state has the same length ℓ. Then we simply assign reward $\frac{1}{(1-\lambda)\cdot\lambda^{\ell-1}}$ to each edge entering a target state, and 0 to all other edges. It is easy to check that the value of v in the resulting discounted MDP is equal to the value of v in the reachability MDP. If the reachability MDP $\mathscr{M}$ does not have the 'uniform path length' property, we modify it by producing $|V|$ copies of itself so that each new vertex carries, apart from the original name, an index from $\{1,\ldots,n\}$. The transition function in the new MDP mimics the original one, but from vertices with index $j<n$ we transition to the appropriate vertices of index $j+1$. The target vertices in the new MDP are those vertices of index n that correspond to a target vertex of the original MDP (this does not break down the reduction, as target vertices in the original vertices can be assumed to have no outgoing edges other than self loops). This new MDP has the desired property and can be produced in a logarithmic space. □

The previous theorem shows that discounted payoff MDPs can be solved in polynomial time even if the discount factor is not fixed (i.e., it is taken as a part of the input). This is an important difference from the 2-player setting. However, the proof, resting on polynomial-time solvability of linear programming, leaves opened a question whether the discounted payoff MDPs be solved in strongly polynomial time. An answer was provided by Ye [Ye11]: already the classic simplex algorithm of Dantzig solves $\mathscr{L}_{disc}$ in a strongly polynomial time in MDPs with a fixed discount factor. Formally, Ye proved that the number of iterations taken by the simplex method is bounded by $\frac{|V|^2\cdot(|A|-1)}{1-\lambda}\cdot\log(\frac{|V|^2}{1-\lambda})$, with each iteration requiring $\mathscr{O}(|V|^2\cdot|A|)$ arithmetic operations. This has also an impact on the strategy improvement method: it can be shown that strategy improvement in discounted MDPs is really just a 're-implementation' of the simplex algorithm using a different syntax. Hence, the strongly polynomial complexity bound for a fixed discount factor holds there as well.

Theorem 60 (Discounted payoff MDPs with a fixed discount factor)**.** *For MDPs with a fixed discount factor, the value of each vertex as well as an optimal MD strategy can be computed in a strongly polynomial time.*

5.3 Mean payoff in MDPs: General properties and linear programming

We will use the lim inf variant of mean payoff:

$$\texttt{MeanPayoff}^{-}(\pi)=\liminf_{n}\frac{1}{n}\sum_{i=0}^{n-1}c(\pi_i)$$

In particular, the play-based expected mean payoff achieved by σ from v is defined as $\mathsf{p\text{-}Payoff}(v,\sigma)=\mathbb{E}_v^\sigma[\texttt{MeanPayoff}^{-}]$, while for the step-based counterpart we have $\mathsf{s\text{-}Payoff}(v,\sigma)=\liminf_{n\to\infty}\frac{1}{n}\sum_{i=0}^{n-1}\mathbb{E}_v^\sigma[c(\pi_i)]$.

The use of lim inf is natural in the formal verification setting: taking the lim inf rather than lim sup emphasizes the worst-case behaviour along a play. However, all the

results in this section hold also for lim sup-based mean payoff, though some proofs are more complex. See the bibliographic remarks for more details.

In general, s-Payoff(v,σ) can be different from p-Payoff(v,σ). However, we have the following simple consequence of the dominated convergence theorem:

Lemma 60. *Let U be the set of all plays π for which $\lim_{n\to\infty}\frac{1}{n}\sum_{i=0}^{n-1}[c(\pi_i)]$ is undefined. If v_0,σ are such that $\mathbb{P}^{\sigma}_{v_0}(U)=0$, then s-Payoff$(v_0,\sigma)$ = p-Payoff(v_0,σ).*

In particular, for any finite-memory strategy σ, the two values coincide, since applying such a strategy turns an MDP into a finite Markov chain where the existence of the limit can be inferred using decomposition into strongly connected components and applying the Ergodic theorem. We will show that in our case of finite-state MDPs, the two approaches coincide not only at the levels of finite-memory strategies, but also as optimality criteria: that is, no matter which of the two semantics we use, the optimal values are the same and a strategy that is optimal w.r.t. one of the semantics is also optimal for the other one. We caution the reader that such a result does not automatically transfer to more complex settings, such as infinite-state MDPs or multi-objective optimization.

To simplify the subsequent notation, we define the *expected one-step* reward of a vertex-action pair (v,a) to be the number $\sum_{w\in V}\Delta(w \mid v,a)\cdot c(v,w)$. Overloading the notation, we denote this quantity by $c(v,a)$.

In mean payoff MDPs, a crucial role is played by the linear program $\mathscr{L}_{mp}$ defined in Equation (5.4). The variables are $x_{(v,a)}$ for $v\in V$ and $a\in A$.

$$\begin{aligned}
\text{maximise}\quad & \sum_{v\in V,a\in A} x_{(v,a)}\cdot c(v,a) \\
\text{subject to}\quad & \sum_{u\in V,a\in A} x_{(u,a)}\cdot\Delta(v\mid u,a)=\sum_{a\in A} x_{v,a} && \text{for all } v\in V \\
& \sum_{v\in V,a\in A} x_{v,a}=1 && \text{for all } v\in V,a\in A \\
& x_{v,a}\geq 0 && \text{for all } v\in V,a\in A
\end{aligned} \tag{5.4}$$

Let us name the equations:

$$\textstyle\sum_{u\in V,a\in A} x_{(u,a)}\cdot\Delta(v\mid u,a)=\sum_{a\in A} x_{v,a} \quad \text{for all } v\in V \tag{5.5}$$

$$\textstyle\sum_{v\in V,a\in A} x_{v,a}=1 \quad \text{for all } v\in V,a\in A \tag{5.6}$$

$$x_{v,a}\geq 0 \quad \text{for all } v\in V,a\in A \tag{5.7}$$

There is a correspondence between feasible solutions of $\mathscr{L}_{mp}$ and the strategies in the corresponding MDP. Intuitively, in a solution corresponding to a strategy σ, the value of a variable $x_{v,a}$ describes the expected frequency with which σ visits v and plays a upon such a visit. These frequencies must obey the *flow constraints* Equation (5.5) and must represent a probability distribution, which is ensured by Equation (5.6) and Equation (5.7). The objective function then represents the expected

mean payoff. Of high importance is also the linear program which is *dual* to $\mathscr{L}_{mp}$. This dual program, denoted $\mathscr{L}_{mp}^{dual}$, is defined in Section 5.3. (For a refresher on linear programming and duality, see Section 1.2).

$$\begin{array}{ll} \text{minimize} & g \\ \text{subject to} & g - c(v,a) + \sum_{u \in V} \Delta(u \mid v,a) \cdot y_u \geq y_v \quad \text{for all } v \in V, a \in A \end{array}$$

A feasible solution of $\mathscr{L}_{mp}$ is a vector $\vec{x} \in \mathsf{R}^{V \times A}$ s.t. setting $x_{(v,a)} = \vec{x}_{(v,a)}$ for all (v,a) satisfies the constraints in $\mathscr{L}_{mp}$. A feasible solution of $\mathscr{L}_{mp}^{dual}$ is a tuple $(g, \vec{y})$, where $g \in \mathsf{R}$ (using the same notation for the number and the variable should not cause confusion here) and $\vec{y} \in \mathsf{R}^V$ s.t. setting the corresponding variables to numbers prescribed by g and $\vec{y}$ satisfies the constraints.

The variable g in $\mathscr{L}_{mp}^{dual}$ is often called *gain* while the vector of y-variables is called *bias*. This is because it provides information on how much does the payoff (under some strategy) accumulated up to a certain step deviate from the estimate provided by the mean payoff value of that strategy. This is illustrated in the following lemma, which forms the first step of our analysis.

Lemma 61 (Properties of feasible solutions)**.** *Let $(g, \vec{y})$ be a feasible solution of $\mathscr{L}_{mp}^{dual}$ and let $Y^{(i)}$, where $i \geq 0$, be a random variable such that $Y^{(i)}(\pi) = \vec{y}_{In(\pi_i)}$. Then for each strategy σ, each vertex v_0, and each $n \geq 0$ it holds $\mathbb{E}^{\sigma}_{v_0}[\sum_{i=0}^{n-1} c(\pi_i)] \leq n \cdot g - \vec{y}_{v_0} + \mathbb{E}^{\sigma}_{v_0}[Y^{(n)}]$.*

Proof. By induction on n. For $n = 0$, both sides are equal to 0. Now assume that the inequality holds for some $n \geq 0$. By the induction hypothesis

$$\mathbb{E}^{\sigma}_{v_0}[\sum_{i=0}^{n} c(\pi_i)] = \mathbb{E}^{\sigma}_{v_0}[\sum_{i=0}^{n-1} c(\pi_i)] + \mathbb{E}^{\sigma}_{v_0}[c(\pi_n)] \leq ng - \vec{y}_{v_0} + \mathbb{E}^{\sigma}_{v_0}[Y^{(n)}] + \mathbb{E}^{\sigma}_{v_0}[c(\pi_n)] \quad (5.8)$$

We now obtain a bound for the third term on the RHS of Equation (5.8). In the following, we denote by Π_n the set of all plays of length n. Then we have

$$\begin{aligned} \mathbb{E}^{\sigma}_{v}[Y^{(n)}] &= \sum_{v \in V} \vec{y}_v \cdot \mathbb{P}^{\sigma}_{v_0}(\mathrm{In}(\pi_n) = v) = \sum_{v \in V} \vec{y}_v \cdot \Bigg(\sum_{\substack{\pi' \in \Pi_n \\ \mathrm{last}(\pi') = v}} \mathbb{P}^{\sigma}_{v_0}(\pi') \Bigg) \\ &= \sum_{v \in V} \vec{y}_v \cdot \Bigg(\sum_{\substack{\pi' \in \Pi_n \\ \mathrm{last}(\pi') = v}} \mathbb{P}^{\sigma}_{v_0}(\pi') \cdot \underbrace{\Big(\sum_{a \in A} \sigma(a \mid \pi') \Big)}_{=1} \Bigg) \\ &= \sum_{\substack{v \in V \\ a \in A}} \vec{y}_v \cdot \Bigg(\sum_{\substack{\pi' \in \Pi_n \\ \mathrm{last}(\pi') = v}} \mathbb{P}^{\sigma}_{v_0}(\pi') \cdot \sigma(a \mid \pi') \Bigg) \\ &\leq \sum_{\substack{v \in V \\ a \in A}} \Bigg(g - c(v,a) + \sum_{u \in V} \Delta(u \mid v,a) \cdot \vec{y}_u \Bigg) \cdot \Bigg(\sum_{\substack{\pi' \in \Pi_n \\ \mathrm{last}(\pi') = v}} \mathbb{P}^{\sigma}_{v_0}(\pi') \cdot \sigma(a \mid \pi') \Bigg) \end{aligned}$$

$$
\begin{aligned}
&= g \cdot \underbrace{\sum_{\substack{v \in V \\ a \in A}} \sum_{\substack{\pi' \in \Pi_n \\ \text{last}(\pi')=v}} \mathbb{P}^{\sigma}_{v_0}(\pi') \cdot \sigma(a \mid \pi')}_{=1} \\
&\quad - \underbrace{\sum_{\substack{v \in V \\ a \in A}} \sum_{\substack{\pi' \in \Pi_n \\ \text{last}(\pi')=v}} \mathbb{P}^{\sigma}_{v_0}(\pi') \cdot \sigma(a \mid \pi') \cdot c(v,a)}_{=\mathbb{E}^{\sigma}_{v_0}[c(\pi_n)]} \\
&\quad + \underbrace{\sum_{\substack{v,u \in V \\ a \in A}} \sum_{\substack{\pi' \in \Pi_n \\ \text{last}(\pi')=v}} \mathbb{P}^{\sigma}_{v_0}(\pi') \cdot \sigma(a \mid \pi') \cdot \Delta(u \mid v,a) \cdot \vec{y}_u}_{=\mathbb{E}^{\sigma}_{v_0}[Y^{(n+1)}]} .
\end{aligned}
$$

The first inequality follows from Section 5.3.

Plugging this into Equation (5.8) yields the desired $\mathbb{E}^{\sigma}_{v_0}[\sum_{i=0}^{n} c(\pi_i)] \leq (n+1)g - \vec{y}_{v_0} + \mathbb{E}^{\sigma}_{v_0}[Y^{(n+1)}]$. □

Corollary 15 (Feasible solutions imply upper bounds)**.** *Let g be the objective value of some feasible solution of $\mathscr{L}^{dual}_{mp}$. Then for every strategy σ and every vertex v_0 it holds $\mathsf{p\text{-}Payoff}(v_0,\sigma) \leq \mathsf{s\text{-}Payoff}(v_0,\sigma) \leq g$.*

Proof. Let $(g,\vec{y})$ be any feasible solution of $\mathscr{L}^{dual}_{mp}$. By Lemma 61 we have, for every $n \geq 0$, that $\mathbb{E}^{\sigma}_{v_0}[\frac{1}{n}\sum_{i=0}^{n-1} c(\pi_i)] \leq g - \frac{\vec{y}_{v_0}}{n} + \frac{1}{n}\mathbb{E}^{\sigma}_{v_0}[Y^{(n)}]$. Since $\vec{y}_{v_0}$ is a constant and $|\mathbb{E}^{\sigma}_{v_0}[Y^{(n)}]|$ is bounded by the constant $\max_{v \in V} |\vec{y}_v|$ that is independent of n, the last two terms on the RHS vanish as n goes to ∞. Hence, we also have that $\mathsf{s\text{-}Payoff}(v_0,\sigma) = \liminf_{n\to\infty} \mathbb{E}^{\sigma}_{v_0}[\frac{1}{n}\sum_{i=0}^{n-1} c(\pi_i)] \leq g$. It remains to show that we have $\mathsf{p\text{-}Payoff}(v_0,\sigma) \leq \mathsf{s\text{-}Payoff}(v_0,\sigma)$, but this immediately follows from the Fatou's lemma [ADD00, Theorem 1.6.8]. □

Corollary 16 (Existence of optimal solutions for the linear programs)**.** *Both the linear programs $\mathscr{L}_{mp}$ and $\mathscr{L}^{dual}_{mp}$ have a feasible solution. Hence, both have an optimal solution and the optimal values of the objective functions in these programs are equal.*

Proof. One can easily construct a feasible solution for $\mathscr{L}^{dual}_{mp}$ by setting all the y-variables to 0 and g to $\max_c$. By the duality theorem for linear programming, to show that also $\mathscr{L}_{mp}$ has feasible solution it suffices to show that the objective function of $\mathscr{L}^{dual}_{mp}$ is bounded from below. But this follows from Corollary 15, since there is at least one strategy σ giving us the lower bound (in particular, the objective function is bounded from below by $-\max_c$). The second part follows immediately by linear programming duality. □

5.4 Mean payoff optimality in strongly connected MDPs

As shown in the previous section, the optimal solution of any of the programs $\mathscr{L}_{mp}$, $\mathscr{L}^{dual}_{mp}$ gives us an upper bound on the optimal value. In this sub-section we show that

in strongly connected MDPs: a) a value of every vertex is the same; b) from a solution of $\mathscr{L}_{mp}$ one can extract a memoryless deterministic strategy σ whose expected mean payoff is well defined (i.e., the preconditions of Lemma 60 are satisfied)) and equal to the objective value of the solution. Moreover, if the solution in question is optimal, then σ is optimal for both p-Payoff- and s-Payoff-semantics.

Definition 12 (Strongly connected MDPs)**.** *An MDP is* strongly connected *if for each pair of vertices u, v there exists a strategy which, when starting in u, reaches v with a positive probability.*

For the rest of this section we fix an optimal solution $\bar{\vec{x}}_{v,a}$ of $\mathscr{L}_{mp}$. We denote by S the set of all vertices for which there exists action a s.t. $\bar{\vec{x}}_{v,a} > 0$. From the shape of $\mathscr{L}_{mp}$ it follows that S is non-empty and closed, and hence we can consider a sub-MDP $\mathscr{M}_S$ induced by S. In $\mathscr{M}_S$ we then define a memoryless randomized strategy σ by putting

$$\sigma(a \mid v) = \frac{\bar{\vec{x}}_{(v,a)}}{\sum_{b \in A} \bar{\vec{x}}_{(v,b)}}.$$

Fixing a strategy σ yields a *Markov chain* $\mathscr{M}_S^\sigma$. Markov chain can be viewed as an MDP with a single action (and hence, with no non-determinism). $\mathscr{M}_S^\sigma$ in particular can be viewed an MDP with the same vertices, edges, and colouring as $\mathscr{M}_S$, but with a single action (as non-determinism was already resolved by σ). The probability of transitioning from a vertex u to a vertex v in a Markov chain is denoted by $P_{u,v}$. In $\mathscr{M}_S^\sigma$ we have $P_{u,v} = \sum_{a \in A} \Delta(v \mid u,a) \cdot \sigma(a \mid u)$, the right-hand side being computed in the original MDP $\mathscr{M}$. Both $\mathscr{M}_S$ and $\mathscr{M}_S^\sigma$ have the same sets of plays and for each initial vertex, the probability measure induced by σ in $\mathscr{M}$ equals the probability measure arising (under the unique policy) in $\mathscr{M}_S^\sigma$. Hence, to prove anything about σ it suffices to analyse $\mathscr{M}_S^\sigma$.

A refresher on Markov chains. We review some fundamental notions of Markov chain theory [Nor98]. A Markov chain that is strongly connected is called *irreducible*. The one-step transition probabilities in a Markov chain can be arranged into a square matrix P, which has one row and one column for each vertex. The cell in the row corresponding to a vertex u and in the column corresponding to a vertex v bears the value $P_{u,v}$ defined above. An easy induction shows that the matrix P^k contains k-step transition probabilities. That is, the probability of being in v after k steps from vertex u is equal to the (u,v)-cell of P^k, which we denote by $P^{(k)}_{u,v}$.

A vertex u of a Markov chain is *recurrent* if, when starting from u, it is revisited infinitely often with probability 1. On the other hand, if the probability that u is revisited only finitely often is one, then the vertex is *transient*. It is known [Nor98, Theorem 1.5.3] that each vertex of a finite Markov chain is either recurrent or transient, and that these two properties can be equivalently characterized as follows: vertex u is recurrent if and only if $\sum_{k=0}^{\infty} P^{(k)}_{u,u} = \infty$, otherwise it is transient.

An *invariant distribution* in a Markov chain with a vertex set V is a $|V|$-dimensional non-negative row vector $\vec{z}$ which adds up to 1 and satisfies $\vec{z} \cdot P = \vec{z}$.

The following lemma holds for arbitrary finite Markov chains.

Lemma 62 (Invariant distributions witness recurrent states)**.** *Let $\vec{z}$ be an invariant distribution and v a vertex such that $\vec{z}_v > 0$. Then v is recurrent.*

Proof. Let n be the number of vertices in the chain and $p_{\min}$ the minimum non-zero entry of P. Assume, for the sake of contradiction, that v is transient. We show that in such a case, for each vertex u it holds $\lim_{k\to\infty} P^{(k)}_{u,v} = 0$. For $u = v$ this is immediate, since the sum $\sum_{k=0}^{\infty} P^{(k)}_{v,v}$ converges for transient v. Otherwise, let $f_{u,v,i}$ be the probability that a play starting in u visits v for the *first time* in exactly i steps. Then $P^{(k)}_{u,v} = \sum_{i=0}^{k} f_{u,v,i} \cdot P^{(k-i)}_{v,v}$. Now when starting in a vertex from which v is reachable with a positive probability, at least one of the following events happens with probability $\geq p^n_{\min}$ in the first n steps: either we reach a vertex from which v is not reachable with positive probability, or we reach v. If neither of the events happens, we are, after n steps, still in a vertex from which v can be reached with a positive probability. In such a case, the argument can be inductively repeated (analogously to the proof of Theorem 54) to show that $f_{u,v,i} \leq (1-p^n_{\min})^{\lfloor \frac{i}{n} \rfloor} \leq (1-p^n_{\min})^{\frac{i-n}{n}}$.

Since $\sum_{k=0}^{\infty} P^{(k)}_{v,v}$ converges, for each $\varepsilon > 0$ there exists j_ε such that $\sum_{i=j_\varepsilon}^{\infty} P^{(i)}_{v,v} < \frac{\varepsilon}{2}$. Similarly, there exists ℓ_ε such that

$$\sum_{i=\ell_\varepsilon}^{\infty} (1-p^n_{\min})^{\frac{i-n}{n}} = \frac{(1-p^n_{\min})^{\frac{\ell_\varepsilon}{n}}}{\left(1-(1-p^n_{\min})^{\frac{1}{n}}\right) \cdot (1-p^n_{\min})} < \frac{\varepsilon}{2},$$

and hence $\sum_{i=\ell_\varepsilon}^{\infty} f_{u,v,i} < \frac{\varepsilon}{2}$.

Now we put $m_\varepsilon = \max\{j_\varepsilon, \ell_\varepsilon\}$. For any $k \geq 2m_\varepsilon$ we have $P^{(k)}_{u,v} = \sum_{i=0}^{k} f_{u,v,i} \cdot P^{(k-i)}_{v,v} \leq \sum_{i=m_\varepsilon}^{k} f_{u,v,i} + \sum_{i=0}^{m_\varepsilon} P^{(k-i)}_{v,v} \leq \sum_{i=m_\varepsilon}^{k} f_{u,v,i} + \sum_{i=m_\varepsilon}^{k} P^{(i)}_{v,v} < \varepsilon$ (note that all the series involved are non-negative). This proves that $P^{(k)}_{u,v}$ vanishes in the limit.

Finally, we derive the contradiction. Since $\vec{z}$ satisfies $\vec{z} \cdot P = \vec{z}$, we also have $\vec{z} \cdot P^k = \vec{z}$ for all k. Hence, the v-component of $\vec{z} \cdot P^k$ is equal to $\vec{z}_v > 0$. But as shown above, the v-column of P^k converges to the all-zero vector as $k \to \infty$, so also $(\vec{z} \cdot P^k)_v$ vanishes in the limit, a contradiction. □

Towards the optimality of σ. We now turn back to the chain $\mathscr{M}^\sigma_S$, where the memoryless strategy σ is obtained from the optimal solution of $\mathscr{L}_{mp}$. In general, $\mathscr{M}^\sigma_S$ does not have to be irreducible. Hence, we use the following lemma and its corollary to extract an irreducible sub-chain, to which we can apply known results of Markov chain theory.

Lemma 63 (Invariant distributions witness recurrent MDPs)**.** *Let $\bar{\vec{z}}$ be a vector such that for each $v \in S$ it holds $\bar{\vec{z}}_v = \sum_{a\in A} \bar{\vec{x}}_{v,a}$. Then $\bar{\vec{z}}$ is an invariant distribution of $\mathscr{M}^\sigma_S$. Consequently, all vertices of $\mathscr{M}^\sigma_S$ are recurrent.*

Proof. The first part follows directly from the fact that $\bar{\vec{x}}_{v,a}$ is a feasible solution of $\mathscr{L}_{mp}$. The second part follows from Lemma 62 and from the fact that $\bar{\vec{z}}$ is positive (by the definition of S). □

Corollary 17 (Extraction of strongly connected components)**.** *The set S can be partitioned into subsets $S_1, S_2, \ldots, S_m$ such that each S_i induces a strongly connected sub-chain of $\mathscr{M}_S^\sigma$.*

Proof. Let $v \in S$ be arbitrary and let $U_v \subseteq S$ be the set of all vertices reachable with positive probability from v in $\mathscr{M}_S^\sigma$. Then v is reachable (in $\mathscr{M}_S^\sigma$) with positive probability from each $u \in U_v$: otherwise, there would be a positive probability of never revisiting v, a contradiction with each vertex being recurrent in $\mathscr{M}_S^\sigma$ (Lemma 63). Hence, U_v induces a strongly connected 'sub-MDP' (or sub-chain) of $\mathscr{M}_S^\sigma$. It is easy to show that if $U_v \neq U_w$ for some $v \neq w$, then the two sets must be disjoint. □

Hence, we can extract from S a set Q inducing a strongly-connected sub-chain of $\mathscr{M}_S^\sigma$, which we denote $\mathscr{M}_Q^\sigma$. The set Q also induces a strongly connected sub-MDP of $\mathscr{M}$ denoted by $\mathscr{M}_Q$. The chain $\mathscr{M}_Q^\sigma$ arises by fixing, in $\mathscr{M}_Q$, a strategy formed by a restriction of σ to Q. We use the following powerful theorem to analyse $\mathscr{M}_Q^\sigma$.

Theorem 61 (*Ergodic theorem*)**.** *In a strongly connected Markov chain (with a finite set of vertices V) there exists a unique invariant distribution $\vec{z}$. Moreover, for every vector $\vec{h} \in \mathsf{R}^V$ the following equation holds with probability 1:*

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \vec{h}_{In(\pi_i)} = \sum_{v\in V} \vec{z}_v \cdot \vec{h}_v.$$

(In particular, the limit is well-defined with probability 1).

We refer to Theorem 1.10.2 in [Nor98] for a proof of the Ergodic theorem.

We can use the Ergodic theorem to shows that the expected mean payoff achieved by σ in $\mathscr{M}_Q$ matches the optimal value of $\mathscr{L}_{mp}$, in a very strong sense: the probability of a play having a mean payoff equal to this optimal value is 1 under σ.

Theorem 62. *Let σ_Q be the restriction of σ to Q. Then for every $v \in Q$ it holds that $\mathbb{P}^{\sigma_Q}_{\mathscr{M}_Q,v}(\texttt{MeanPayoff}^{-} = r^*) = 1$, where r^* is the is the optimal value of $\mathscr{L}_{mp}$.*

Proof. Let $\vec{w} \in \mathsf{R}^{V\times A}$ be a vector such that $\vec{w}_{(v,a)} = \bar{\vec{x}}_{(v,a)} / \sum_{(q,a)\in Q\times A} \bar{\vec{x}}_{(q,a)}$ for every $(v,a) \in Q \times A$, and $\vec{w}_{(v,a)} = 0$ for all other (v,a). We claim that $\vec{w}$ is also an optimal solution of $\mathscr{L}_{mp}$.

To prove feasibility, note that setting $\vec{w}_{(v,a)} = 0$ for each $v \in V \setminus Q$ does not break the constraints Equation (5.5). This is because Q induces a strongly connected sub-chain of $\mathscr{M}_S^\sigma$, and hence there are no $v \in V$, $u \in V \setminus Q$ such that $\bar{\vec{x}}_{(u,a)} \cdot \Delta(v \mid u,a) > 0$. Next, Equation (5.5) is invariant w.r.t. multiplication of variables by a constant, so normalizing the remaining values preserves Equation (5.5) and ensures that Equation (5.6) holds.

To prove optimality, assume that the objective value of $\vec{w}$ is smaller than r^*. Then we can mirror the construction from the previous paragraph and produce a feasible solution $\hat{\vec{w}}_{(v,a)}$ whose $(Q\times A)$-indexed components are zero and the rest are normalized components of $\bar{\vec{x}}$. Then r^* is a convex combination of the objective values of $\vec{w}$ and $\hat{\vec{w}}$, so $\hat{\vec{w}}$ must have a strictly larger value than r^*, a contradiction with the latter's optimality.

We now plug $\vec{w}$ into the ergodic theorem as follows: As in Lemma 63, it easy to prove that setting $\vec{z}_v = \sum_{a\in A} \vec{w}_{(v,a)}$ yields an invariant distribution. Now put $\vec{h}_v = \sum_{a\in A} \sigma(a \mid v) \cdot c(v,a) (= \sum_{w\in V} P_{v,w} \cdot c(v,w))$. From the Ergodic theorem we get that $\lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \vec{h}_{\mathrm{In}(\pi_i)}$ almost-surely exists and equals

$$\begin{aligned}
\sum_{v\in Q} \vec{z}_v \cdot \vec{h}_v &= \sum_{v\in V} \left(\Big(\sum_{d\in A} \vec{w}_{(v,d)} \Big) \cdot \Big(\sum_{a\in A} \sigma(a \mid v) \cdot c(v,a) \Big) \right) \\
&= \sum_{v\in Q} \left(\left(\frac{\sum_{d\in A} \bar{\vec{x}}_{(v,d)}}{\sum_{\substack{q\in Q \\ b\in A}} \bar{\vec{x}}_{(q,b)}} \right) \cdot \left(\frac{\sum_{a\in A} \bar{\vec{x}}_{(v,a)} \cdot c(v,a)}{\sum_{d\in A} \bar{\vec{x}}_{(v,d)}} \right) \right) \\
&= \frac{1}{\sum_{\substack{q\in Q \\ b\in A}} \bar{\vec{x}}_{(q,b)}} \cdot \sum_{\substack{v\in Q \\ a\in A}} \bar{\vec{x}}_{(v,a)} \cdot c(v,a) = \sum_{\substack{v\in Q \\ a\in A}} \vec{w}_{(v,a)} \cdot c(v,a) = r^*. \qquad (5.9)
\end{aligned}$$

It remains to take a step from the left-hand side of Equation (5.9) towards the mean payoff. To this end, we construct a new Markov chain $\mathscr{M}'_Q$ from $\mathscr{M}_Q$ by 'splitting' every edge (u,v) with a new dummy vertex $d_{u,v}$ (i.e., $d_{u,v}$ has one edge incoming from u with probability $P_{u,v}$ and one edge outgoing to v with probability 1). In $\mathscr{M}'_Q$ we define a vector $\vec{h}'$ s.t. for each vertex $d_{u,v}$ the vector $\vec{h}'$ has the $d_{u,v}$-component equal to $c(u,v)$, while the components corresponding to the original vertices are zero. It is easy to check that $\mathscr{M}'_Q$ is strongly connected and that it has an invariant distribution $\vec{z}'$ defined by $\vec{z}'_v = \vec{z}_v/2$ for v in Q and $\vec{z}'_{d_{u,v}} = \frac{\vec{z}_u \cdot P_{u,v}}{2}$ for (u,v) an edge of $\mathscr{M}_Q$. Also, by easy induction, for each play π of length n in $\mathscr{M}_Q$ it holds $\frac{1}{n} \sum_{i=0}^{n-1} c(\pi_i) = \frac{1}{n} \sum_{i=0}^{2n-1} \vec{h}'_{\mathrm{In}(\pi'_i)}$, where π' is the unique play in $\mathscr{M}'_Q$ obtained from π by splitting edges with appropriate dummy vertices. Hence,

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} c(\pi_i) = 2 \cdot \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \vec{h}'_{\mathrm{In}(\pi'_i)}, \qquad (5.10)$$

provided that both limits exist. By the ergodic theorem applied to $\mathscr{M}'_Q$, we have that the RHS limit in Equation (5.10) is defined with probability 1 and equal to

$$\begin{aligned}
\underbrace{\sum_{v\in Q} \vec{z}'_v \cdot \vec{h}'_v}_{=0} + \sum_{u,v\in Q} \vec{z}'_{d_{u,v}} \cdot \vec{h}'_{d_{u,v}} &= \frac{1}{2} \sum_{u\in Q} \vec{z}_u \cdot \left(\sum_{v\in Q} P_{u,v} \cdot c(u,v) \right) \\
&= \frac{1}{2} \sum_{u\in Q} \vec{z}_u \cdot \vec{h}_u = \frac{r^*}{2},
\end{aligned}$$

the last equality being shown above. Plugging this into Equation (5.10) yields that if a limit on the LHS (i.e., the mean payoff of a play) is well-defined with probability 1, then it is equal to r^* also with probability 1. But if there was a set L of positive probability in $\mathscr{M}_Q$ with $\lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} c(\pi_i)$ undefined for each $\pi \in L$, by splitting the plays in L we

would obtain a positive-probability set of plays in $\mathscr{M}'_Q$ in which $\lim_{n\to\infty} \frac{1}{n}\sum_{i=0}^{n-1} \vec{h}'_{\mathrm{In}(\pi'_i)}$ is also undefined, a contradiction with the Ergodic theorem. □

So far, we have constructed an optimal strategy σ_Q but only on the part Q of the original MDP $\mathscr{M}$. To conclude the construction, we define a memoryless strategy σ^* in $\mathscr{M}$ as follows: we fix a memoryless deterministic strategy $\sigma_{=1}$ that is winning, from each vertex of $\mathscr{M}$, for the objective of almost-sure reaching of Q (such a strategy exists since $\mathscr{M}$ is strongly connected, see also Theorem 54. Then we put $\sigma^*(v) = \sigma_{=1}(v)$ if $v \notin Q$ and $\sigma^*(v) = \sigma_Q(v)$ otherwise. Hence, starting in any vertex, σ^* eventually reaches Q with probability 1 and then it starts behaving as σ_Q. The optimality of such a strategy follows from the prefix independence of mean payoff, as argued in the next theorem.

Theorem 63 (Prefix independence of mean payoff)**.** *For any sequence of numbers $c_0, c_1, \ldots$ and any $k \in \mathbb{N}$ it holds $\liminf_{n\to\infty} \frac{1}{n}\sum_{i=0}^{n-1} c_i = \liminf_{m\to\infty} \frac{1}{m}\sum_{i=0}^{m-1} c_{k+i}$. As a consequence, for every vertex v in $\mathscr{M}$ it holds $\mathbb{P}^{\sigma_Q}_{\mathscr{M}_Q,v}(\texttt{MeanPayoff}^{-} = r^*) = 1$, where r^* is the optimal value of $\mathscr{L}_{mp}$. Hence, $\mathbb{E}^{\sigma^*}_v[\texttt{MeanPayoff}^{-}] = r^*$.*

Proof. We have

$$\liminf_{n\to\infty} \frac{c_0 + \cdots c_{n-1}}{n} = \liminf_{n\to\infty} \left(\underbrace{\frac{k}{n}}_{\text{vanishes for } n\to\infty} \cdot \frac{c_0 + \cdots + c_{k-1}}{k} + \underbrace{\frac{n-k}{n}}_{\to 1 \text{ for } n\to\infty} \cdot \frac{c_k + \cdot + c_{n-1}}{n-k} \right)$$
$$= \liminf_{m\to\infty} \frac{c_k + \cdots + c_{k+m-1}}{m}.$$

A similar argument holds for $\limsup$.

With probability 1, a play has an infinite suffix consisting of plays from $\mathscr{M}^{\sigma}_Q$, and thus also $\texttt{MeanPayoff}^{-}$ and $\texttt{MeanPayoff}^{+}$ determined by this suffix. By Theorem 62, these quantities are equal to r^* with probability 1. □

The following theorem summarizes the computational aspects.

Theorem 64 (Complexity of solving strongly connected mean payoff MDPs)**.** *In a strongly connected mean payoff MDP, one can compute, in polynomial time, a memoryless randomized strategy which is optimal from every vertex, as well as the (single) optimal value of every vertex.*

Proof. We obtain, in polynomial time, an optimal solution of $\mathscr{L}_{mp}$, with the optimal objective value being the optimal value of every vertex (Theorem 63). We then use this optimal solution $\vec{x}$ to construct the strategy σ and the Markov chain $\mathscr{M}^{\sigma}_S$. From this chain we extract a strongly connected subset of vertices Q (in polynomial time, by a simple graph reachability analysis). With the subset in hand, we can construct strategies σ_Q and $\sigma_{=1}$, all polynomial-time computations (see Theorem 54). These two strategies are then combined to produce the optimal strategy σ^*. □

Deterministic optimality in strongly connected MDPs

It remains to prove that we can actually compute a memoryless *deterministic* strategy that is optimal in every vertex. Looking back at the construction that resulted in Theorem 64, we see that the optimal strategy σ^* might be randomized because the computed optimal solution $\bar{\vec{x}}$ of $\mathscr{L}_{mp}$ can contain two components (v,a), (v,b) with $a \neq b$ and both $\bar{\vec{x}}_{(v,a)}$ and $\bar{\vec{x}}_{(v,b)}$ being positive. To prove memoryless deterministic optimality, we will show that there is always an optimal solution which yields a deterministic strategy, and that such a solution can be obtained in polynomial time.

The previous section implicitly defined two mappings: First, a mapping Ψ, which maps every solution $\vec{x}$ of $\mathscr{L}_{mp}$ to a memoryless strategy in some sub-MDP of $\mathscr{M}$, by putting $\Psi(\vec{x}) = \sigma$ where $\sigma(a \mid v) = \vec{x}_{(v,a)} / \sum_{b \in A} \vec{x}_{(v,b)}$. Second, mapping Ξ, which maps each memoryless strategy σ that induces a strongly connected Markov chain to a solution $\Xi(\sigma)$ of $\mathscr{L}_{mp}$ such that $\Xi(\sigma)_{(v,a)} = \vec{z}_v \cdot \sigma(a \mid v)$, where $\vec{z}$ is the unique invariant distribution of the chain induced by σ.

Lemma 64 (Correspondence between solutions and strategies)**.** *Let X be the set containing exactly those solutions $\vec{x}$ of $\mathscr{L}_{mp}$ for which the strategy $\Psi(\vec{x})$ induces a strongly connected Markov chain. Then the mappings Ψ and Ξ are bijections between X and the set of all memoryless strategies in some sub-MDP of $\mathscr{M}$ that induce a strongly connected Markov chain.*

Proof. A straightforward computation shows that $\Xi \circ \Psi$ and $\Psi \circ \Xi$ are identity functions on the respective sets. □

Definition 13 (Pure solutions)**.** *A solution $\vec{x}$ of $\mathscr{L}_{mp}$ is* pure *if for every vertex v there is at most one action a such that $\vec{x}_{(v,a)} > 0$.*

The following lemma follows from the way in which strategies σ and σ^* were constructed in the previous sub-section.

Lemma 65 (Solutions of the linear programs)**.** *Let $\bar{\vec{x}}$ be a pure optimal solution of $\mathscr{L}_{mp}$ and denote $S = \{v \in V \mid \exists a \text{ s.t. } \bar{\vec{x}}_{(v,a)} > 0\}$. Then the strategy $\sigma = \Psi(\bar{\vec{x}})$ is an MD strategy in $\mathscr{M}_S$. Hence, in such a case, the strategy σ^* constructed from σ as in Theorem 64 is an optimal MD strategy in $\mathscr{M}$.*

It remains to show how to find a pure optimal solution of $\mathscr{L}_{mp}$. To this end we exploit some fundamental properties of linear programs.

A linear program is in the *standard* (or equational) form if its set of constraints can be expressed as $A \cdot \vec{x} = \vec{b}$, $\vec{x} \geq 0$, where $\vec{x}$ is a vector of variables, $\vec{b}$ is a non-negative vector, and A is a matrix of an appropriate dimension. In this notation, all the vectors are column vectors, i.e. A has one column per each variable. Note that $\mathscr{L}_{mp}$ is a program in the standard form. A feasible solution $\vec{x}$ of such a program is *basic* if the columns of A corresponding to variables whose value is positive in $\vec{x}$ form a linearly independent set of vectors. Since the maximal number of linearly independent columns equals the maximal number of linearly independent rows (a number called a *rank* of A), we know that each basic feasible solution has at most as many positive entries as there are rows of A.

The next two lemmas prove some fundamental properties of basic feasible solutions.

Lemma 66 (Properties of basic feasible solutions: uniqueness)**.** *Assume that a linear program in a standard form has two basic feasible solutions $\vec{x},\vec{x}'$ such that both solutions have the same set of non-zero components, and the cardinality of this set equals the number of equality constraints in the program. Then $\vec{x}=\vec{x}'$.*

Proof. Write $A\cdot\vec{x}=\vec{b}$ the equational constraints of the LP. If $\vec{x}$ is a basic feasible solution, then it solves the equation $A_N\cdot\vec{x}_N=\vec{b}$, where A_N (N stands for 'non-zero') is obtained from A by removing all columns corresponding to zero components of $\vec{x}$, and $\vec{x}_N$ is obtained from $\vec{x}$ by removing all zero components.

Since $\vec{x}$ has as many non-zero components as there are rows of A, it follows that A_N is a square matrix. Since $\vec{x}$ is a basic solution, A_N is regular (its columns are linearly independent) and $\vec{x}=A_N^{-1}\cdot\vec{b}$ is uniquely determined by A_N. Repeating the same argument for $\vec{x}'$ yields $\vec{x}'=A_N^{-1}\cdot\vec{b}=\vec{x}$. □

Lemma 67 (Properties of basic feasible solutions)**.** *If a linear program in a standard form has an optimal solution, then it has also a basic optimal solution. Moreover, a basic optimal solution can be found in polynomial time.*

Sketch. The existence of a basic optimal solution is a well-known linear programming fact, e.g. the standard simplex algorithm works by traversing the set of basic feasible solutions until it finds an optimal one [Mat07]. For computing an optimal basic solution, we can use one of the polynomial-time interior-point methods for linear programming, such as the path-following method [Kar84, Gon92]. While these methods work by traversing the interior of the polyhedron of feasible solutions, they converge, in polynomial time, to a point that is closer to the optimal basic solution than to all the other basic solutions. By a process called *purification,* such a point can be then converted to the closest basic solution, i.e. to the optimal one [Gon92]. □

Theorem 65 (Complexity of computing optimal deterministic strategies in strongly connected mean payoff MDPs)**.** *One can find, in polynomial time, an optimal deterministic strategy in a given strongly connected mean payoff MDP.*

Proof. First, we use Lemma 67 to find a basic optimal solution $\bar{\vec{x}}$ of $\mathscr{L}_{mp}$. We check if it is pure. If yes, we are done. Otherwise, there is $v\in V$ and two distinct actions a,b such that $\bar{\vec{x}}_{(v,a)}>0$ and $\bar{\vec{x}}_{(v,b)}>0$. Let $S=\{v\in V\mid\exists a \text{ s.t. } \bar{\vec{x}}_{(v,a)}>0\}$. By Corollary 17, we can partition S into several subsets, each of which induces a strongly connected sub-MDP of $\mathscr{M}$. Let Q be a class of this partition containing v. We have that the optimal mean payoff value of every vertex in $\mathscr{M}_Q$ is the same as in $\mathscr{M}$. This is because, as in the beginning of the proof of Theorem 62, we can transform $\bar{\vec{x}}$ into another optimal solution of the same value as $\bar{\vec{x}}$ which has non-zero entries only for components indexed by (q,a) with $q\in Q$. All these computations can be easily implemented in polynomial time.

We argue that Q is a strict subset of V. Indeed, assume that $Q=V$. Then $\bar{\vec{x}}$ induces a randomized strategy σ in $\mathscr{M}$. Moreover, since $\bar{\vec{x}}$ is a basic solution, it has at most $|V|+1$ positive entries, and since it is non-pure, it must have exactly $n+1$ positive

entries, i.e. Lemma 66 is applicable to $\bar{\vec{x}}$, since $\mathscr{L}_{mp}$ has exactly $|V|+1$ constraints. Now we define a new strategy σ' in $\mathscr{M}$ by slightly changing the behaviour in v. To this end, choose some $\varepsilon > 0$ and put $\sigma'(a \mid v) = \sigma(a \mid v) - \varepsilon$ and $\sigma'(b \mid v) = \sigma(b \mid v) + \varepsilon$; we choose ε small enough so that both quantities are still non-zero. The chain $\mathscr{M}^{\sigma'}$ is still strongly connected. Now let $\vec{x}' = \Xi(\sigma')$. Then $\vec{x}'$ is a solution of $\mathscr{L}_{mp}$ which is still basic, with a set of non-zero components being the same as in $\bar{\vec{x}}$. At the same time, $\vec{x}' \neq \bar{\vec{x}}$, since $\sigma \neq \sigma'$ and Ξ is a bijection (Lemma 64). But this is a contradiction with Lemma 66.

Hence, $\mathscr{M}_Q$ is a strict sub-MDP of $\mathscr{M}$ in which the value of every vertex is the same as in the original MDP. We can perform a recursive call of the aforementioned computation on $\mathscr{M}_Q$ (compute basic optimal solution of $\mathscr{L}_{mp}$, check purity, possibly extract and recurse on a sub-MDP). The depth of recursion is bounded by $|V|$, so the running time is polynomial. Since each sub-MDP obtained during the recursion is non-empty, and the size of the MDPs decreases, the recursion must eventually terminate with a basic optimal solution (in some sub-MDP $\mathscr{M}'$) that is pure. This yields a memoryless deterministic strategy in $\mathscr{M}'$ whose value is equal to the optimal value in $\mathscr{M}$. Such a strategy can be extended to whole $\mathscr{M}$ by solving almost sure reachability to $\mathscr{M}'$, as described in the previous sub-section. □

5.5 End components

To solve mean payoff optimization in general MDPs, as well as general optimization problems for ω-regular objectives, we need to introduce a crucial notion of an *end component*.

Definition 14. *An* end component (EC) *of an MDP is any set M of vertices having the following two properties:*

- *For each $u \in M$ there exists an action a that is M-safe in u, i.e. satisfies that for all vertices v with $\Delta(v \mid u,a) > 0$ it holds $v \in M$.*
- *For each pair of distinct vertices $u, v \in M$ there is a path from u to v visiting only the states from M.*

In other words, M is an EC of $\mathscr{M}$ if and only if M is a closed set and the sub-MDP $\mathscr{M}_M$ is strongly connected.

From the player's point of view, the following property of ECs is important.

Lemma 68. *Let M be an EC and $v \in M$. Then there is an MD strategy σ which, when starting in a vertex inside M, never visits a vertex outside of M and at the same time ensures that with probability one, the vertex v is visited infinitely often. Moreover, σ can be computed in polynomial time.*

Proof. From Theorem 54 we know that we can compute, in polynomial time, an MD strategy σ in the sub-MDP $\mathscr{M}_M$ ensuring that v is reached with probability 1 from any initial vertex in M. Indeed, this is because $M = W_{>0}(\mathscr{M}_M, \texttt{Reach}(v))$, due to the second condition in the definition of a MEC. Since M is closed, this strategy never leaves M.

Whenever, the strategy leaves v, it guarantees that we return to v with probability 1. Hence, for each k, the probability of event V_k — visiting v at least k times — is 1. Since $V_{k+1} \subseteq V_k$, it follows that also the probability of $\bigcap_{i=1}^{\infty} V_i$ is equal to 1, which is what we aimed to prove. □

The main reason for introducing ECs is that they are crucial for understanding the limiting behaviour of MDPs.

Definition 15. *We denote by* $\texttt{Inf}(\pi)$ *the set of vertices that appear infinitely often along a play* π.

Lemma 69. *For any* v_0 *and* σ *it holds* $\mathbb{P}^{\sigma}_{v_0}(\{\pi \mid \texttt{Inf}(\pi) \text{ is an EC } \}) = 1$.

Proof. Assume the converse. Then in some MDP there is a set of vertices X which is not an EC but satisfies $\mathbb{P}^{\sigma}_{v_0}(\texttt{Inf} = X) > 0$. Since X is not an EC, there is a vertex $v \in X$ in which any (even randomized) choice of action results in leaving X with probability at least $p_{\min} > 0$ (recall that $p_{\min}$ is the smallest non-zero edge probability in the MDP).

Let $Stay_k$ be the set of plays in $\{\texttt{Inf} = X\}$ which, from step k on, never visit a vertex outside of X. Since $\{\texttt{Inf} = X\} = \bigcup_{i=1}^{\infty} Stay_i$, by union bound we get $\mathbb{P}^{\sigma}_{v_0}(Stay_{k_0}) > 0$ for some $k_0 \in \mathbb{N}$. Let Vis_j denote the set of all plays in $Stay_{k_0}$ that visit v at least j times *after* the step k_0. Since $Stay_{k_0} \subseteq \{\texttt{Inf} = X\}$, we have $Stay_{k_0} \cap Vis_j = Stay_{k_0}$ for each j. But an easy induction shows that $\mathbb{P}^{\sigma}_{v_0}(Stay_{k_0} \cap Vis_{j+1}) \leq \mathbb{P}^{\sigma}_{v_0}(\{\mathrm{In}(\pi_{k_0}) \in X\}) \cdot p_{\min}^{j}$, since every visit to v brings a risk at least $p_{\min}$ of falling out of X. The latter number converges to zero, so $\mathbb{P}^{\sigma}_{v_0}(Stay_{k_0}) = \lim_{j\to\infty} \mathbb{P}^{\sigma}_{v_0}(Stay_{k_0}) = \lim_{j\to\infty} \mathbb{P}^{\sigma}_{v_0}(Stay_{k_0} \cap Vis_{j+1}) = 0$, a contradiction. □

In general, there can be exponentially many end components in an MDP (e.g. for a complete underlying graph and one action per edge, each subset of vertices is an EC). However, we can usually restrict to analysing *maximal* ECs.

Definition 16. *An end component* M *is a* maximal end component (MEC) *if no other end-component* M' *is a superset of* M. *We denote by* $MEC(\mathscr{M})$ *the set of all MECs of* $\mathscr{M}$.

If two ECs have a non-empty intersection, then their union is again an EC. Hence, every EC is contained in exactly one MEC, and the total number of MECs is bounded by $|V|$, since two distinct MECs must be disjoint. Moreover, the decomposition of an MDP into MECs can be computed in polynomial time.

Theorem 66. *The set of all MECs in a given MDP can be computed in polynomial time.*

Proof. There are several known algorithms, a simple one is pictured in Algorithm 5.4. Each iteration goes as follows: we first take the underlying directed graph of the MDP and find its strongly connected components using some of the well-known polynomial algorithms [CLRS09]. We identify the bottom SCCs, i.e. those from which there is no outgoing edge in the graph. It is easy to see that each such SCC must form a MEC of $\mathscr{M}$, and conversely, each MDP has at least one MEC that is also a bottom SCC of its underlying graph. Moreover, for each such bottom SCC B we compute the *random*

Algorithm 5.4: Algorithm for MEC decomposition of an MDP.

Data: An MDP $\mathscr{M}$
$List \leftarrow \emptyset$; // List of found MECs
$G \leftarrow (V,E)$; // The underlying graph of $\mathscr{M}$
while *G is non-empty* **do**
  Decompose G into strongly connected components;
  $R \leftarrow \emptyset$; // The list of vertices to remove.
  foreach *bottom SCC B of G* **do**
    B is a MEC of $\mathscr{M}$, add it to *List*;
    $R \leftarrow R \cup (V \setminus W_{=1}(\mathscr{M}, \mathtt{Safe}(V \setminus B)))$; // Schedule removal of vertices from which B cannot be avoided in $\mathscr{M}$.
  remove vertices in R from G along with adjacent edges
return *List*

attractor of B, i.e. the set of vertices of $\mathscr{M}$ from which B cannot be avoided under any strategy. To this end, we compute, in polynomial time, the almost-surely winning set $W_{=1}(\mathscr{M}, \mathtt{Safe}(V \setminus B))$ which is the largest largest (w.r.t. set inclusion) subset of V from which the player can ensure to stay in $V \setminus B$ forever (i.e. the complement of the random attractor of B). The computation can be done in polynomial time by Corollary 13). No vertex of the random attractor of B can belong to a MEC different from B: such a MEC would be disjoint from B but the player could not force avoiding B from within this MEC, a contradiction with MEC being a closed set. Hence, all MECs of $\mathscr{M}$ which are not a bottom SCC of G are subsets of $W_{=1}(\mathscr{M}, \mathtt{Safe}(V \setminus B)) \subseteq V \setminus R$, so we can remove all vertices in R from the graph and continue to the next iteration (note that removing vertices in R from $\mathscr{M}$ again yields a MDP, since the complement of R is an intersection of closed sets, and thus again a closed set). The main loop performs at most $|V|$ iterations, which yields the polynomial complexity. □

5.6 Reductions to optimal reachability

The MEC decomposition can be used to reduce several optimization problems (including general mean payoff optimization) to optimizing reachability probability. Recall that in the optimal reachability problem, we are given an MDP $\mathscr{M}$ (with coloured vertices) and a colour Win $\in C$. The task is to find a strategy σ that maximizes $\mathbb{P}^{\sigma}_{v_0}(\mathtt{Reach}(\mathrm{Win}))$, the probability of reaching a vertex coloured by Win. The main result on reachability MDPs, which we prove in Section 5.7, is as follows:

Theorem 67 (Solving reachability MDPs)**.** *In reachability MDPs, the value of each vertex is rational and computable in polynomial time. Moreover, we can compute, in polynomial time, a memoryless deterministic strategy that is optimal in every vertex.*

From optimal Büchi to reachability

In Büchi MDPs, the vertices are assigned colours from the set $\{1,2\}$ and our aim is to find a strategy maximizing $\mathbb{P}^{\sigma}_{v_0}(\texttt{Buchi})$, i.e. maximizing the probability that a vertex coloured by 2 is visited infinitely often. We say that a MEC M of a Büchi MDP is *good* if it contains a vertex coloured by 2.

Theorem 68 (Solving Büchi MDPs)**.** *In Büchi MDPs, the value of each vertex is rational and computable in polynomial time. Moreover, we can compute, in polynomial time, a memoryless deterministic strategy that is optimal in every vertex.*

Proof. Let $\mathscr{M}_b$ be a Büchi MDP and let $\mathscr{M}_r$ be a reachability MDP obtained from $\mathscr{M}_b$ by repainting each vertex belonging to a good MEC with the colour Win. Note that $\mathscr{M}_r$ can be computed in polynomial time by performing the MEC decomposition of $\mathscr{M}_b$ (Algorithm 5.4) and checking goodness of each MEC.

We prove that the value of each vertex in $\mathscr{M}_b$ is equal to the value of the corresponding vertex in $\mathscr{M}_r$.

First, fix any σ and v_0 (due to equality of underlying graphs, we can view these as a strategy/initial vertex both in $\mathscr{M}_b$ and $\mathscr{M}_r$). By Lemma 69, the probability of visiting infinitely often a vertex outside of a MEC is 0. Hence, the probability of visiting infinitely often a vertex coloured by 2 (in $\mathscr{M}_b$) is the same as the probability of visiting infinitely often a vertex coloured by 2 which belongs to (a necessarily good) MEC, which is in turn bounded from above by the probability that σ visits (in $\mathscr{M}_r$) a vertex coloured by Win.

Conversely, let σ^* be the MD reachability-optimal strategy in $\mathscr{M}_r$ (which exists by Theorem 67). We construct a strategy σ in $\mathscr{M}_b$ which achieves, in every vertex, the same Büchi-value as the reachability value achieved in that vertex by σ^* in $\mathscr{M}_r$. Outside of any good MEC, σ behaves exactly as σ^*. Inside a good MEC M, σ behaves as the MD strategy from Lemma 68, ensuring that some fixed vertex in M of colour 2 is almost-surely visited infinitely often. Since σ is stitched together from MD strategies on non-overlapping domains, it is memoryless deterministic and it ensures that once a good MEC is reached, the Büchi condition is satisfied almost-surely.

The construction of σ in the aforementioned paragraph is effective: given the optimal strategy σ^* for reachability, σ can be constructed in polynomial time. □

From optimal parity to optimal reachability

In parity MDPs, the vertices are labelled by colours form the set $\{1,\dots,d\}$ (w.l.o.g. we stipulate that $d \leq |V|$) and the goal is to find a strategy maximizing $\mathbb{P}^{\sigma}_{v_0}(\texttt{Parity})$, i.e. maximizing the probability that the largest priority appearing infinitely often along a play is even.

Theorem 69 (Solving parity MDPs)**.** *In Parity MDPs, the value of each vertex is rational and computable in polynomial time. Moreover, we can compute, in polynomial time, a memoryless deterministic strategy that is optimal in every vertex.*

Proof. Let $\mathscr{M}_p$ be a parity MDP. We will proceed similarly to Theorem 68, constructing a reachability MDP $\mathscr{M}_r$ with the same underlying graph as $\mathscr{M}_p$.

To this end, let $\mathscr{M}_i$ be the largest sub-MDP of $\mathscr{M}_p$ containing only the vertices of priority $\leq i$. Formally, we set $V_i = W_{=1}(\mathscr{M}_p, \mathtt{Safe}(c^{-1}(\{i+1,\ldots,d\})))$ and define $\mathscr{M}_i$ to be the sub-MDP induced by V_i (note that $\mathscr{M}_i$ might be empty). We say that a vertex of $\mathscr{M}_p$ is i-good if it is contained in some MEC M of $\mathscr{M}_i$ such that the largest vertex priority inside M is equal to i. We say that a vertex is even-good if it is i-good for some even i. We set up a reachability MDP $\mathscr{M}_r$ by taking $\mathscr{M}_p$ and re-colouring each its even-good vertex with colour Win. To do this, we need to compute, for each even priority i, the MDP $\mathscr{M}_i$ and its MEC-decomposition. This can be done in polynomial time (Algorithm 5.4).

We again prove that the value of every vertex in $\mathscr{M}_p$ is equal to the value of the corresponding vertex in $\mathscr{M}_r$.

Let σ and v_0 be arbitrary. By Lemma 69, $\mathbb{P}^{\sigma}_{\mathscr{M}_p,v_0}(\mathtt{Parity})$ is equal to the probability that $\mathtt{Inf}(\pi)$ is an EC in which the largest priority is even. But each such EC is also an EC of some $\mathscr{M}_i$ with even i, and thus is also contained in a MEC of a $\mathscr{M}_i$ in which the largest priority is i. Hence, $\mathbb{P}^{\sigma}_{\mathscr{M}_p,v_0}(\mathtt{Parity}) \leq \mathbb{P}^{\sigma}_{\mathscr{M}_r,v_0}(\mathtt{Reach}(\mathrm{Win}))$.

Conversely, let σ^* be the MD reachability-optimal strategy in $\mathscr{M}_r$. We construct an MD strategy σ in $\mathscr{M}_p$ as follows: in a vertex v which is not even-good, σ behaves as σ^*. For a vertex v that is even-good, we identify the smallest even i such that v is i-good. This means that v belongs to some MEC M of $\mathscr{M}_i$ in which the largest priority is i. By Lemma 68, we can compute, in polynomial time, an MD strategy σ_M which ensures that the largest-priority vertex in $(\mathscr{M}_i)_M$ is visited infinitely often, and we set $\sigma(v)$ to $\sigma_M(v)$. Note that given σ^*, the strategy σ can be constructed in polynomial time. It remains to show that $\mathbb{P}^{\sigma}_{\mathscr{M}_p,v_0}(\mathtt{Parity}) \geq \mathbb{P}^{\sigma^*}_{\mathscr{M}_r,v_0}(\mathtt{Reach}(\mathrm{Win}))$.

By the construction of σ, once we reach a vertex which is i-good for some even i, all the following vertices will be j-good for some even $j \leq i$. From this and from Lemma 69 it follows that $\mathbb{P}^{\sigma^*}_{\mathscr{M}_r,v_0}(\mathtt{Reach}(\mathrm{Win}))$ is equal to the probability that σ produces a play π with the following property: $\exists i$ even such that all but finitely many vertices on π are i-good but are not j-good for any even $j < i$. This can be in turn rephrased as the probability that $\mathtt{Inf}(\pi)$ is an EC whose all vertices are i-good for some even i but none of them is j-good for an even $j < i$; we call such an EC *i-definite*. But within such an EC, σ forever behaves as σ_M for some MEC $\mathscr{M}$ of $\mathscr{M}_i$ in which the maximal priority is i. Hence, once an i-definite EC is reached, the strategy almost-surely ensures that priority i is visited infinitely often and ensures that no larger priority is ever visited. It follows that $\mathbb{P}^{\sigma^*}_{\mathscr{M}_r,v_0}(\mathtt{Reach}(\mathrm{Win})) = \mathbb{P}^{\sigma}_{\mathscr{M}_p,v_0}(\inf(\pi) \text{ is } i\text{-definite for even } i) = \mathbb{P}^{\sigma}_{\mathscr{M}_p,v_0}(\mathtt{Parity})$. □

From general mean payoff to optimal reachability

We already know how to solve strongly connected mean payoff MDPs. We now combine this result with MEC decomposition to reduce the general (not strongly connected) mean payoff optimization to MDP reachability.

We start with a strengthening of Theorem 63.

Lemma 70. *Let $\mathscr{M}$ be a strongly connected mean payoff MDP and r^* the value of each of its vertices. Then, for each σ and v_0 we have $\mathbb{P}^{\sigma}_{v_0}(\mathtt{MeanPayoff}^{-} > r^*) = 0$.*

Proof. Assume that the statement is not true. Then there exist σ, v_0 as well as numbers $\varepsilon, \delta > 0$ and $n_0 \in \mathbb{N}$ s.t. the probability of the following set of plays X_{ε,n_0} is at least δ: a play π belongs to X_{ε,n_0} if for every $n \geq n_0$ it holds $\frac{1}{n}\sum_{i=0}^{n-1} c(\pi_i) \geq x^* + \varepsilon$. We construct a new strategy σ', which proceeds in a series of episodes. Every episode starts in v_0, and for the first n_0 steps of the, episode σ' mimics σ. After that, it checks, in every step n, whether the payoff accumulated since the start of the episode is at least $n \cdot (r^* + \varepsilon)$. If this holds, we mimic σ for one more step. If the inequality is violated, we immediately 'restart', i.e. return to v_0 (can be performed with probability 1 due to the MDP being strongly connected) and once in v_0, start a new episode which mimics σ from the beginning. By our assumption, the probability of not performing a reset in a given episode is at least $\delta > 0$. Hence, with probability 1 we witness only finitely many resets, after which we produce a play whose suffix has mean payoff at least $r^* + e$. By prefix independence of mean payoff (Theorem 63), $\mathbb{E}_{v_0}^{\sigma'}[\texttt{MeanPayoff}^{-}] \geq r^* + \varepsilon$, a contradiction. □

We will need to strengthen the previous lemma so that it applies not only to strongly connected MDPs, but also to MECs in some larger MDPs. The strengthening is performed in the following two lemmas. The first lemma says that once we exit a MEC, with some positive probability we will never return.

Lemma 71. *Let M be a MEC of an MDP $\mathscr{M}$ and let $v \in M$, $a \in A$ be such that a* is not *M-safe in v. Then there exists t s.t. $\Delta(t \mid v,a) > 0$ and $t \notin W_{=1}(\mathscr{M}, \texttt{Reach}(M))$.*

Proof. Assume that a is not M-safe in v and that all t's with $\Delta(t \mid v,a) > 0$ belong to $W_{=1}(\mathscr{M}, \texttt{Reach}(M))$. Fix the MD strategy σ which is almost-surely winning for reaching M from each vertex of $W_{=1}(\mathscr{M}, \texttt{Reach}(M))$ (Theorem 54). For each t s.t. $\Delta(t \mid v,a) > 0$, let M_t denote the set of vertices which can be (with a positive probability) visited under σ. Put $M' = M \cup (\bigcup_{t \in V, \Delta(t|v,a)>0} M_t)$. Then M' is closed, since M is closed and since for every u in some M_t there exists an action (the one selected by σ for u) under which we surely stay in M_t. Moreover, the M'-induced sub-MDP is strongly connected: each t with $\Delta(t \mid v,a) > 0$ is reachable from within M (through v) and thus each vertex in some M_t is reachable from M. In turn, from each vertex in some M_t (where $\Delta(t \mid v,a) > 0$) we can reach M without leaving M_t, due to the definition of σ. Hence, M' is a MEC which strictly contains M, a contradiction with the maximality of M. □

Given a play π and strategy σ, we define a *slice* of σ as a strategy σ_{π^-} such that for each π' starting in $\text{last}(\pi)$ it holds $\sigma_{\pi^-}(\pi') = \sigma(\pi\pi')$, while on other plays σ_{π^-} just mimics σ.

Lemma 72. *Let M be a MEC of $\mathscr{M}$ and r^* the mean payoff value of every vertex in the strongly connected sub-MDP induced by M. Then the set E of all plays that have $\texttt{Inf}(\pi) \subseteq M$ and at the same time mean payoff greater than r has probability zero under any strategy σ.*

Proof. Assume, for contradiction, that there is a strategy σ and $\delta > 0$ such that the probability of E under σ is at least δ. Note that we do not immediately have a contradiction with Lemma 70, since σ might leave M (and then return back).

We say that a play π *cheats* in step i if it is inside M in i-th step and outside of M in the next step (which can only be caused by an M-unsafe action being played). From Lemma 71 we have that there is $p > 0$ s.t. upon every exit from M we return with probability at most $(1-p)$. Hence, the probability that a play cheats infinitely often is 0. It follows that there is $k \in \mathbb{N}$ s.t. $\mathbb{P}^{\sigma}_{v_0}(\pi \text{ cheats after } \geq k \text{ steps}) \leq (\delta \cdot p_{\min})/4$, where $p_{\min}$ is the smallest non-zero edge probability in $\mathscr{M}$.

Whenever we are in some $v \in M$ and play an action that is not M-safe in v, this results into a cheat with probability at least $p_{\min}$. Thus, the total probability that this happens after at least k steps, i.e. the quantity

$$q = \sum_{i \geq k} \sum_{v \in M} \sum_{a \text{ not } M\text{-safe in v}} \mathbb{E}^{\sigma}_{v_0}[A^{\sigma}_{a,i} \cdot \mathbf{1}_{\mathrm{Out}(\pi_i)=v}], \tag{5.11}$$

is bounded by $\mathbb{P}^{\sigma}_{v_0}(\pi \text{ cheats after more than } k \text{ steps})/p_{\min} \leq \delta/4$.

Let's go back to E now. On each play in E there is a step i from which on the play stays in M forever: we say that the play is i-definite and we denote by E_k the set of all i-definite plays in E. By union bound, there is $\ell \in \mathbb{N}, \ell \geq k$ s.t. $\mathbb{P}^{\sigma}_{v_0}(E_\ell) \geq \delta/2$.

We define a new strategy σ' as follows: on each play prefix, σ' by default mimics σ, except for the case when at least ℓ steps have elapsed, the current vertex v is in M, and σ prescribes to play, with positive probability, an action which is not M safe in v. In such a case, σ is overridden and we play any action that is M-safe in v instead (after which we return to simulating σ, until the override kicks in again). The probability that such an override happens is bounded by the quantity q from Equation (5.11), and hence by $\delta/4$. Since $\mathbb{P}^{\sigma}_{v_0}(E_\ell) \geq \delta/2$, at least half the measure of E_ℓ stays untouched by the overrides; hence $\mathbb{P}^{\sigma'}_{v_0}(E_\ell) \geq \delta/4$.

We are ready to apply the final argument. There are only finitely many plays of length ℓ. Hence, by union bound, there is a play π of length ℓ such that $\mathbb{P}^{\sigma'}_{v_0}(E_\ell \cap Cyl(\pi)) > 0$. Consider the strategy σ'_{π^-}. Starting in $\mathrm{last}(\pi)$, we have that σ'_{π^-} never leaves M, due to the overrides in σ'. Hence, σ'_{π^-} can be seen as a strategy in the strongly connected MDP $\mathscr{M}_M$. Now consider the set $E' = \{\pi' \mid \pi' \exists \pi'' \in E \text{ s.t. } \pi'' = \pi\pi'\}$. Then $\mathbb{P}^{\sigma'_{\pi^-}}_{\mathrm{last}(\pi)}(E') = \mathbb{P}^{\sigma'}_{v_0}(E_\ell \cap Cyl(\pi)) > 0$; but due to the prefix independence of mean payoff, all plays in E' have payoff $> r^*$, a contradiction with Lemma 70. □

Theorem 70. *In mean payoff MDPs, the value of each vertex is rational and computable in polynomial time. Moreover, we can compute, in polynomial time, a memoryless deterministic strategy that is optimal in every vertex.*

Proof. First, note that we can w.l.o.g. restrict to MDPs in which each edge is coloured by a number between 0 and 1. To see this, let $\mathscr{M}$ be an MDP and a, b any two numbers, with a non-negative. We can construct an MDP $\mathscr{M}'$ by re-colouring each edge (u,v) of $\mathscr{M}$ with colour $a \cdot c(u,v) + b$, where c is the original colouring in $\mathscr{M}$. It is then easy to see that for each strategy σ it holds $\mathbb{E}^{\sigma}_{\mathscr{M},v_0}[\texttt{MeanPayoff}^{-}] = (\mathbb{E}^{\sigma}_{\mathscr{M}',v_0}[\texttt{MeanPayoff}^{-}]/a) - b$, so a strategy optimizing the mean payoff in $\mathscr{M}'$ is also optimal in $\mathscr{M}$. Hence, we always can re-scale the colouring into the unit interval while preserving the optimization criterion.

So now let $\mathscr{M}_{mp}$ be a mean payoff MDP with edge-colouring c. We construct, in polynomial time, a new reachability MDP $\mathscr{M}_r$ as follows: first, we compute the MEC

decomposition of $\mathscr{M}_{mp}$ (Algorithm 5.4). Let $M_1, \ldots, M_k$ be all the resulting MECs. For each MEC M_i we compute the optimal mean payoff value r_i^* in the sub-MDP induced by M_i (which is shared by all vertices of this sub-MDP, by Theorem 63), along with the corresponding memoryless deterministic optimal strategy. We already know how to do this in polynomial time (Theorems 64 and 65). Now we add new vertices v_{good}, v_{bad}, both with self loops, and edges incoming to these vertices from each vertex that belongs to some MEC of $\mathscr{M}_{mp}$. The vertex v_{good} is the only vertex coloured by Win in $\mathscr{M}_r$. Finally, we add a new action fin which behaves as follows: For each vertex v belonging to a MEC M_i we set $\Delta(v_{good} \mid v, fin) = r_i^*$ and $\Delta(v_{bad} \mid v, fin) = 1 - r_i^*$. In a non-MEC vertex v, we put $\Delta(v, fin) = \Delta(v, a)$ for some $a \in A$, $a \neq fin$, so that no new behaviour is introduced in these vertices.

We show that for any original vertex (i.e. all vertices but v_{good}, v_{bad}) the optimal values in both MDPs are the same and the optimal strategies are easily transferable from one MDP to the other.

First, let σ be an ε-optimal strategy in $\mathscr{M}_{mp}$. We have

$$\begin{array}{rcl} \mathbb{E}^{\sigma}_{v_0}[\texttt{MeanPayoff}^{-}] & = & \sum_{i=1}^{k} \mathbb{E}^{\sigma}_{v_0}[\texttt{MeanPayoff}^{-} \cdot \mathbf{1}_{\texttt{Inf} \subseteq M_i}] \\ & \leq & \sum_{i=1}^{k} \mathbb{E}^{\sigma}_{v_0}[r_i^* \cdot \mathbf{1}_{\texttt{Inf} = M_i}] \\ & = & \sum_{i=1}^{k} r_i^* \cdot \mathbb{P}^{\sigma}_{v_0}(\texttt{Inf} = M_i) \end{array}$$

Here the first equation follows from Lemma 69 and the subsequent inequality from Lemma 72. Moreover, for each i there is a number n_0^i such that the probability of all plays that stay inside M_i in all the steps from n_0^i to infinity is at least $\mathbb{P}^{\sigma}_{v_0}(\texttt{Inf} \subseteq M_i) - \frac{\varepsilon}{k}$. Let $n_0 = \max_{1 \leq i \leq k} n_0^i$.

We construct a reachability strategy σ_r which mimics σ for the first n_0 steps. After n_0 steps it performs a switch: if the current vertex is in some M_i we immediately play the action fin, otherwise we start to behave arbitrarily. We have

$$\begin{array}{rcl} \mathbb{P}^{\sigma_r}_{v_0}(\texttt{Reach}(\text{Win})) & \geq & \sum_{i=1}^{k} r_i^* \cdot \mathbb{P}^{\sigma_r}_{v_0}(\text{last}(\pi_{\leq n_0}) \in M_i) \\ & \geq & \sum_{i=1}^{k} r_i^* \cdot \mathbb{P}^{\sigma}_{v_0}(\texttt{Inf} \subseteq M_i) - \varepsilon \\ & \geq & \mathbb{E}^{\sigma}_{v_0}[\texttt{MeanPayoff}^{-}] - \varepsilon. \end{array}$$

This is the last equality shown in the previous paragraph. Since σ is ε-optimal for mean payoff, $\mathbb{P}^{\sigma_r}_{v_0}(\texttt{Reach}(\text{Win}))$ is at most 2ε away from the mean payoff value of v. Since $\varepsilon > 0$ was chosen arbitrarily, we get that the reachability value in $\mathscr{M}_r$ is at least as large as the mean payoff value in $\mathscr{M}_{mp}$.

Conversely, let σ^* be the optimal MD strategy in $\mathscr{M}_r$. We say that σ^* ends in a vertex v if $\sigma^*(v) = fin$. We can assume that if σ^* ends in some $v \in M_i$ then it ends in all vertices of M_i. This is because whenever σ^* ends in some vertex $v \in M_i$, the reachability value of v must be equal to r_i^*, otherwise playing fin would not be optimal here. But the optimal reachability value in every vertex of a given MEC is the same (due to Lemma 68), so if playing fin is optimal in some vertex of M_i, it is optimal in all such vertices. Now we can define an MD strategy σ_{mp} in $\mathscr{M}_{mp}$ to initially mimic σ^*, and upon encountering any MEC M_i in which σ^* ends, immediately switch to the MD strategy that is optimal in the mean payoff sub-MDP $\mathscr{M}_i$. We have $\mathbb{E}^{\sigma_{mp}}_{v_0}[\texttt{MeanPayoff}^{-}] = \sum_{i=1}^{k} \mathbb{P}^{\sigma^*}_{v_0}(\text{end in } M_i) \cdot r_i^* = \mathbb{P}^{\sigma^*}_{v_0}(\texttt{Reach}(\text{Win}))$. Since σ^*

as well as the optimal strategies in all M_i can be computed in polynomial time (Theorems 65 and 67), we get the result. □

5.7 Optimal reachability

In this final section, we prove Theorem 67. The proof bears many similarities to the methods for discounted MDPs, hence we only sketch the process and point out the key differences. Throughout the section we assume that *targets are sinks*, i.e. that a vertex coloured by Win has only a self loop as the single outgoing edge. Modifying an MDP to accommodate this does not influence reachability probabilities in any way.

Consider the reachability operator $\mathscr{R}\colon [0,1]^V \to [0,1]^V$ such that for $\vec{y} = \mathscr{R}(\vec{x})$ it holds

$$\vec{y}_v = \begin{cases} \max_{a\in A} \sum_{u\in V} \Delta(u \mid v,a)\cdot x_u & c(v) \neq \text{Win} \\ 1 & c(v) = \text{Win}. \end{cases}$$

Lemma 73. *For each initial vector $\vec{x}$, the limit $\lim_{k\to\infty}\mathscr{R}^k(\vec{x})$ exists. Moreover, if $\vec{x} \leq \mathscr{R}(\vec{x})$, then the limit is equal to the least fixed point of $\mathscr{R}$ that is greater than or equal to $\vec{x}$; if $\mathscr{R}(\vec{x}) \leq \vec{x}$, then the limit is equal to the greatest fixed point of $\mathscr{R}$ that is less than or equal to $\vec{x}$.*

Proof. The existence of the limit follows from the monotonicity of $\mathscr{R}$. In addition, it can be easily checked that the set $[0,1]^V$ is a directed complete partial order and that $\mathscr{R}$ is a Scott-continuous operator on this set. Hence, the result follows from the Kleene's theorem (see also Tarski-Kantorovich principle). □

We denote by $\texttt{Reach}^k(\text{Win})$ the set of all plays that reach Win within the first k steps. Clearly, for each σ and v_0 we have $\lim_{k\to\infty}\mathbb{P}^\sigma_{v_0}(\texttt{Reach}^k(\text{Win})) = \mathbb{P}^\sigma_{v_0}(\texttt{Reach}(\text{Win}))$.

Lemma 74. *For each $k \in \mathbb{N}$ and $v \in V$, $\mathscr{R}^k(\vec{0})_v = \sup_\sigma \mathbb{P}^\sigma_v(\texttt{Reach}^k(Win))$. In particular, the vector $\vec{x}^* = \lim_{k\to\infty}\mathscr{R}^k(\vec{0})$ is the least fixed point of $\mathscr{R}$ and it is equal to the vector of reachability values.*

Proof. The first part can be proved by a straightforward induction, the second part follows by Lemma 73 and a simple limiting argument. □

Similarly to Definition 11 we say that an action a is $\vec{x}$-safe in v if it holds that $a = \arg\max_{a'\in A} \sum_{u\in V} \Delta(u \mid v,a')\cdot\vec{x}_u$. Recall that a strategy σ is $\vec{x}$-safe if all actions selected in a vertex with non-zero probability are $\vec{x}$-safe in that vertex.

Lemma 75. *Let $\vec{x}^*$ be as in Lemma 74. Next, let $Z^{(n)}$ be a random variable which for a given time step n looks at the current vertex v after n steps and returns the value $\vec{x}^*_v$. Then for every $\vec{x}^*$-safe strategy σ it holds $\mathbb{E}^\sigma_{v_0}[Z^{(n)}] = \vec{x}^*_{v_0}$. Moreover, it holds $\mathbb{E}^\sigma_{v_0}[Z^{(n)}\cdot\mathbf{1}_{c(Out(\pi_n))=Win}] = \mathbb{P}^\sigma_{v_0}(\texttt{Reach}^n(Win))$.*

Proof. By an easy induction on n, using the fact that target states are sinks. □

Now an analogue of Lemma 53 does not hold for reachability: a strategy playing only $\vec{x}^*$-safe actions might not be optimal (indeed, it might not reach Win at all). Instead, we proceed as follows: Let $\mathscr{M}^*$ be an MDP in which we 'disable', in each state v, all actions that are not $\vec{x}^*$-safe in v. This can be formally done by adding a new non-target sink vertex *sink*, an edge from each original vertex to *sink*, and stipulating that each action a that is disabled in a vertex v chooses, when played in v in $\mathscr{M}^*$, the edge leading to *sink* with probability 1.

Lemma 76. *The vectors of reachability values $val(\mathscr{M})$ and $val(\mathscr{M}^*)$ are equal. In particular, $W_{>0}(\mathscr{M}, \texttt{Reach}(Win)) = W_{>0}(\mathscr{M}^*, \texttt{Reach}(Win))$.*

Proof. Let $\vec{x}^*$ again denote the vector of optimal values in $\mathscr{M}$. If all actions in $\mathscr{M}$ are $\vec{x}^*$-safes, then the lemma clearly holds. Otherwise there is some $\delta \in (0,1)$ such that for each action a that is not $\vec{x}^*$-safe in some vertex v it holds $\sum_{u \in V} \Delta(u \mid v,a) \cdot \vec{x}_u \leq \vec{x}^*_v - \delta$.

Let $\varepsilon \in (0,\delta)$ be arbitrary and fix an ε-optimal strategy σ in $\mathscr{M}$. We will show that there is a $(2\varepsilon/\delta)$-optimal strategy σ' which only uses $\vec{x}^*$-safe actions. Since ε can be chosen arbitrarily close to 0, this shows that $\vec{x}^*$-safe strategies can get arbitrarily close to the value, hence $\mathrm{val}(\mathscr{M}^*) = \mathrm{val}(\mathscr{M})$.

The strategy σ' initially mimics σ up to the first point in time when an action that is not $\vec{x}^*$-safe in the current vertex is to be selected. At this point σ' switches to behave as any $\vec{x}^*$-safe strategy. To analyse the value achieved by σ', we need to bound the probability of the event *NonSafe* that the switch occurs. By the same reasoning as in Lemma 75, we can show that for all n it holds $\mathbb{P}^\sigma_{v_0}(\texttt{Reach}^n(\mathrm{Win})) \leq \mathbb{E}^\sigma_{v_0}[Z^{(n)}] \leq \vec{x}^*_{v_0} - \delta \cdot \mathbb{P}^\sigma_{v_0}(\mathit{NonSafe}^{(n)})$, where $\mathit{NonSafe}^{(n)}$ is the probability that a switch occurs in the first n steps. By taking n to the limit we get $\mathbb{P}^\sigma_{v_0}(\texttt{Reach}(\mathrm{Win})) \leq \vec{x}^*_{v_0} - \delta \cdot \mathbb{P}^\sigma_{v_0}(\mathit{NonSafe})$. At the same time $\vec{x}^*_{v_0} - \varepsilon \leq \mathbb{P}^\sigma_{v_0}(\texttt{Reach}(\mathrm{Win}))$. Combining these two inequalities yields $\mathbb{P}^\sigma_{v_0}(\mathit{NonSafe}) \leq \frac{\varepsilon}{\delta}$. Now clearly $\mathbb{P}^{\sigma'}_{v_0}(\texttt{Reach}(\mathrm{Win})) \geq \vec{x}^*_{v_0} - \varepsilon - \mathbb{P}^\sigma_{v_0}(\mathit{NonSafe}) \geq \vec{x}^*_{v_0} - \varepsilon - \varepsilon/\delta \geq \vec{x}^*_{v_0} - 2\varepsilon/\delta$. □

Lemma 77. *Given the vector $\vec{x}^*$ of optimal reachability values, we can compute, in polynomial time, the optimal MD reachability strategy in $\mathscr{M}$.*

Proof. Given $\vec{x}^*$, we construct the MDP $\mathscr{M}^*$ and compute the winning strategy σ for positive reachability in $\mathscr{M}^*$. We already know that σ can be taken memoryless and computed in polynomial time (Theorem 53). We claim that σ is an optimal reachability strategy in $\mathscr{M}$. By Lemma 76 it suffices to show that σ is optimal in $\mathscr{M}^*$. Let W be the winning region for positive reachability in $\mathscr{M}^*$. Since σ is memoryless, with probability 1 we reach either Win or a vertex of value 0 (from which we cannot return to W anymore); in other words, for almost all plays π we have that $\mathbf{1}_{\mathrm{Out}(\pi_n) \in W}$ eventually equals $\mathbf{1}_{c(\mathrm{Out}(\pi_n)) = \mathrm{Win}}$. Hence, using Lemma 75 we get $\vec{x}^*_{v_0} = \lim_{n\to\infty} \mathbb{E}^\sigma_{v_0}[Z^{(n)}] = \mathbb{E}^\sigma_{v_0}[\lim_{n\to\infty} Z^{(n)}] = \mathbb{E}^\sigma_{v_0}[\lim_{n\to\infty} Z^{(n)} \cdot \mathbf{1}_{\mathrm{Out}(\pi_n) \in W}] = \mathbb{E}^\sigma_{v_0}[\lim_{n\to\infty} Z^{(n)} \cdot \mathbf{1}_{c(\mathrm{Out}(\pi_n)) = \mathrm{Win}}] = \mathbb{P}^\sigma_{v_0}(\texttt{Reach}(\mathrm{Win}))$. Here, the third equality holds since $\vec{x}^*_v$ is zero for $v \notin W$, while the swapping of expectations and limits can be performed due to the dominated convergence theorem. □

To finish the proof of Theorem 67, it remains to prove that the vector of optimal values $\vec{x}^*$ can be computed in polynomial time. We again employ linear programming and define the linear program $\mathscr{L}_{reach}$ with variables x_v, $v \in V$.

$$
\begin{array}{lll}
\text{minimise} & \sum_{v \in V} x_v & \\
\text{subject to} & x_v = 1 & \text{if } c(v) = \text{Win} \\
& x_v = 0 & \text{if } v \notin W_{>0}(\mathscr{M}, \texttt{Reach}(\text{Win})) \\
& x_v \geq \sum_{u \in V} \Delta(u \mid v,a) \cdot x_u & \text{for all other } v \in V, a \in A.
\end{array}
$$

Lemma 78. *The linear program $\mathscr{L}_{reach}$ in Section 5.7 has a unique optimal solution $\bar{\vec{x}}$ such that $\bar{\vec{x}} = \vec{x}^*$.*

Proof. Clearly $\vec{x}^*$ is a feasible solution of $\mathscr{L}_{reach}$. Similarly to Lemma 59 we prove that each feasible solution $\vec{x}$ of $\mathscr{L}_{reach}$ satisfies $\vec{x} \geq \vec{x}^*$. We can proceed analogously to Lemma 59, just replacing the operator $\mathbb{O}$ with $\mathscr{R}$. The proof can be mimicked up to the point where we get that $\lim_{k\to\infty} \mathscr{R}^k(\vec{x}) \leq \vec{x}$ (the limit exists by Lemma 73). Since $\mathscr{R}(\vec{x}) \leq \vec{x}$ for each feasible solution $\vec{x}$, from Lemma 73 we get that the limit is a fixed point of $\mathscr{R}$, and in hence it is greater or equal to the least fixed point of $\mathscr{R}$, i.e. $\vec{x}^*$ (Lemma 74). Hence, also $\vec{x} \geq \lim_{k\to\infty} \mathscr{R}^k(\vec{x}_0) \geq \vec{x}^*$. □

Lemmata 77 and 78 give us Theorem 67.

Bibliographic references

There is a broad field of study related to Markov decision processes, with a history going as far as 1950's [Bel57]. It is beyond the scope of this chapter to provide a comprehensive overview of the related literature. Nonetheless, in this section we provide pointers to the most significant works connected to our techniques as well as to works that can serve as a starting point for a further study.

One of the most widely used references for MDP-related research is the textbook by Puterman [Put05]. The textbook views MDPs from an operations research point-of-view, focusing on finite-horizon, discounted, total-reward, and average reward (an alternative name for mean payoff) objectives. Regular objectives fall outside of the book's focus, though reachability can be viewed as a special case of the "positive bounded total reward" objectives studied in the book. An in-depth study of the textbook will impart to its reader the knowledge of many useful techniques for MDP analysis, though a reader who is a newcomer to MDPs might feel somewhat intimidated by its sheer volume and generality. In this chapter, we follow Puterman's exposition mainly in the discounted payoff, albeit in a rather condensed form.

For mean payoff MDPs, [Put05] follows similar blueprint as in the discounted case: first characterizing the optimal values via a suitable optimality equation and then deriving the value iteration, strategy improvement, and linear programming methods from this characterization. We use the linear programming as our foundational stone, focusing on the relationship between strategies and feasible solutions of the program. We note that value and strategy iteration for mean payoff MDPs come with super-polynomial lower bounds, see, e.g. [Fea10a, Fea10b], or [Put05], where it is shown that strategy improvement converges at least as fast as value iteration.

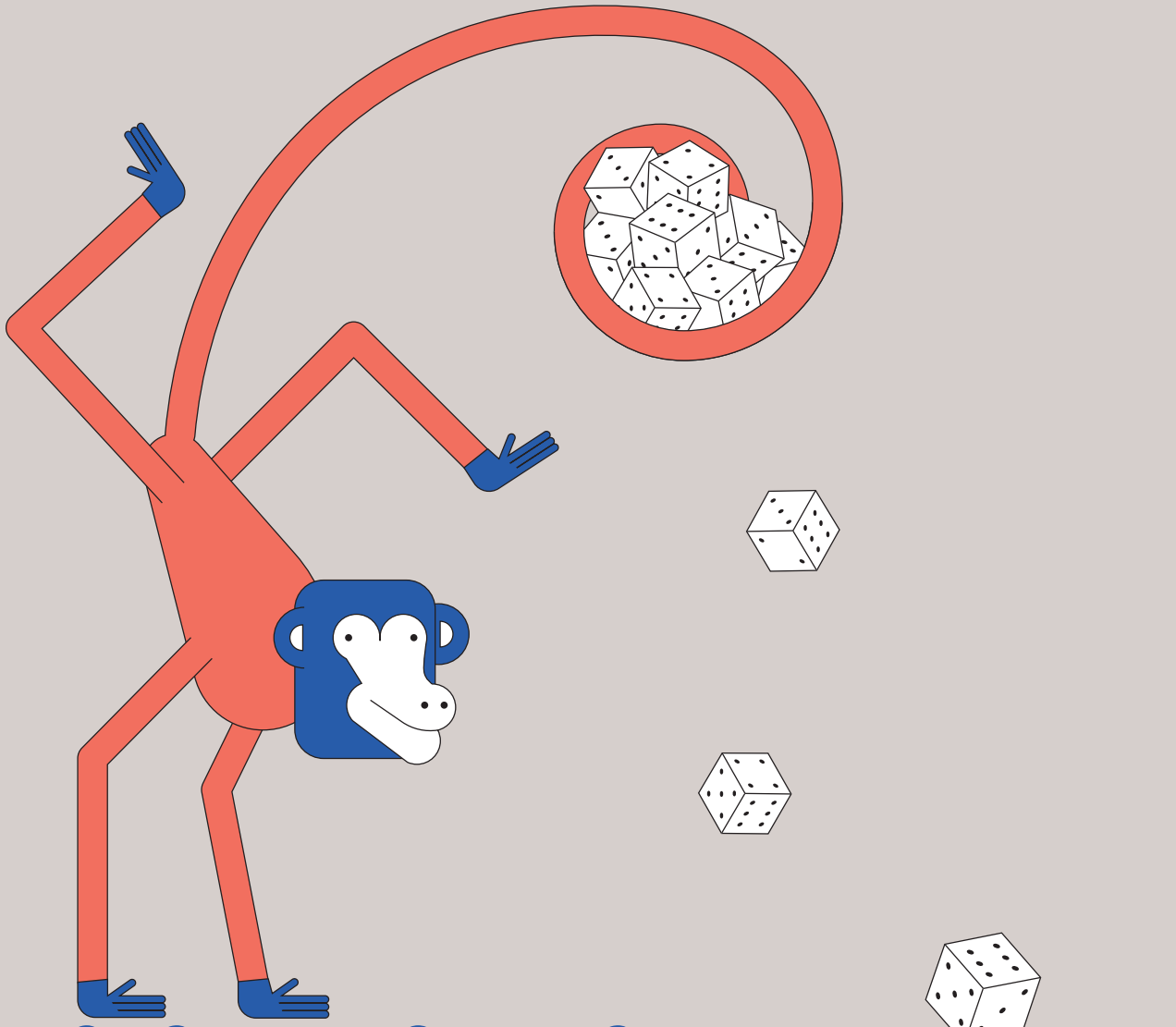
Also, [Put05] makes the initial analysis of mean payoff MDPs in the context of *unichain* MDPs, and then extends to arbitrary MDPs, with strongly connected MDPs

treated as a special case of the latter. While unichain is an important theoretical concept, in the context of formal methods and automata it is preferable to work with strongly connected MDPs. We also note that all the results in the mean payoff sections hold also for $\texttt{MeanPayoff}^{+}$. Almost all of the proofs are the same, with an important exception of Corollary 15, where Fatou's lemma cannot be used to prove that $\mathsf{p\text{-}Payoff}(v_0,\sigma) \leq \mathsf{s\text{-}Payoff}(v_0,\sigma)$. Instead, we could use *martingale techniques* here. Martingales are an important concept in probability theory [Wil91], with applications e.g. in analysis of infinite-state MDPs and stochastic games [BBEK11]. We can use martingales to strengthen Lemma 61 by showing that the probability of $\sum_{i=0}^{n-1} c(\pi_i) \geq \sqrt{n} \cdot \mathbb{E}_{v_0}^{\sigma}[\sum_{i=0}^{n-1} c(\pi_i)]$ converges (with an exponential rate of decay) to 0 as $n \to \infty$, which allows us to prove the required bound for lim sup.

The notion of a (M)EC as well as many techniques we use in the EC section are due to de Alfaro, whose thesis [dA97] details the evolution of the concept and its relation to similar notions. The algorithm for MEC decomposition is taken from [CH11], where more advanced algorithms as well as use of MECs in parity MDPs are discussed.

For an overview of literature related to verification of temporal properties in MDPs, we refer the reader to the monograph [BK08].

MDPs are also used as a prime model in reinforcement learning (RL), one of the classical yet rapidly evolving sub-fields of AI. For RL-centric view of MDPs, we point the reader towards the textbooks [SB18, Ber07a].



STOCHASTIC GAMES

Chapter 6

Stochastic Games

NATHALIE BERTRAND, PATRICIA BOUYER-DECITRE, NATHANAËL FIJALKOW, MATEUSZ SKOMRA

In this chapter, we introduce and review results on stochastic games with two players. On the one hand, they extend 2-player games with random vertices; on the other hand, they extend Markov decision processes with a second player.

When equipped with a simple reachability objective, the objective of Max is to maximise the probability to reach the target. Most of the chapter focuses on stochastic games with reachability objectives. We show uniform positional determinacy in Section 6.1, using a fixed point characterisation of the values. Section 6.2 deals with a number of normalisation steps that are often useful when constructing algorithms for stochastic games. This immediately yields a value iteration algorithm, discussed in Section 6.3, and with a bit more work a strategy improvement algorithm, defined in Section 6.4. We construct two algorithms based on permutations of random vertices in Section 6.5, and conclude in Section 6.6 showing that many other stochastic games reduce to stochastic games with reachability objectives.

Notations

There are two classical models for stochastic arenas, which we present now. In this chapter, we will use the first one: using random vertices. We present it first, and then quickly introduce the second one and explain why they are equivalent.

The model with random vertices

Let us first define stochastic arenas.

Definition 17 (Stochastic arenas)**.** *A* stochastic arena *is* $\mathscr{A} = (G, V_{\text{Max}}, V_{\text{Min}}, V_{\text{Random}}, \delta)$ *where*

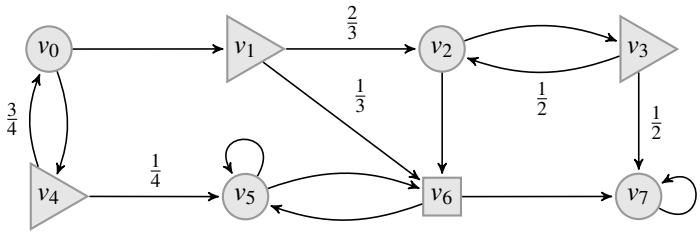


Figure 6.1: Example of a stochastic arena: circle nodes are controlled by Max, square nodes by Min, and triangle nodes are random.

- *G is a graph over the set of vertices V and $V = V_{\text{Max}} \uplus V_{\text{Min}} \uplus V_{\text{Random}}$ partitions the vertices into those controlled by Max, Min, and random vertices.*
- *$\delta : V_{\text{Random}} \to \mathscr{D}(E)$ is the probabilistic transition function.*

Definition 18 (Stochastic games)**.** *Let $\mathscr{A}$ a stochastic arena. A qualitative* stochastic game *is $\mathscr{G} = (\mathscr{A}, W)$ with $W \subseteq Paths_\omega$ a qualitative condition, and a quantitative stochastic game is $\mathscr{G} = (\mathscr{A}, f)$ with $f : Paths_\omega \to \mathsf{R} \cup \{\pm\infty\}$ a quantitative condition.*

Most of the chapter will be devoted to stochastic reachability games, which are induced by `Reach`(Win). For simplicity we assume that Win is a sink (meaning a single vertex with a self-loop), the general case where Win is a subset of edges can be easily reduced to this case. The following definitions are natural extensions of the ones given in Chapter 1 for the stochastic setting.

A *strategy* for Max is a function $\sigma : \text{Paths} \to \mathscr{D}(E)$, and similarly for Min. Note that a strategy is allowed to randomise over its actions. A pure strategy does not use randomisation: $\sigma : \text{Paths} \to E$, and a positional strategy does not use memory: $\sigma : V_{\text{Max}} \to \mathscr{D}(E)$.

When a pair of strategies (σ, τ) and an initial vertex u is fixed, we obtain a stochastic process: we write $\mathbb{P}^u_{\sigma,\tau}$ for the probability measure on infinite plays. For instance, for W a condition, we write $\mathbb{P}^u_{\sigma,\tau}(W)$ for the probability that the infinite path satisfies W when Max plays σ, Min plays τ, and we start from u. Note that we are implicitly assuming that W is measurable. For $f : \text{Paths}_\omega \to \mathsf{R} \cup \{\pm\infty\}$, we write $\mathbb{E}^u_{\sigma,\tau}(f)$ for the expectation of f.

Let us consider a qualitative stochastic game $\mathscr{G} = (\mathscr{A}, W)$. The *value for Max* in $\mathscr{G}$ from u is defined as $\text{val}^{\mathscr{G}}_{\text{Max}}(u) = \sup_\sigma \inf_\tau \mathbb{P}^u_{\sigma,\tau}(W)$, and symmetrically the value for Min is $\text{val}^{\mathscr{G}}_{\text{Min}}(u) = \inf_\tau \sup_\sigma \mathbb{P}^u_{\sigma,\tau}(W)$. Clearly enough, $\text{val}^{\mathscr{G}}_{\text{Max}}(u) \leq \text{val}^{\mathscr{G}}_{\text{Min}}(u)$.

Let us now consider a quantitative stochastic game $\mathscr{G} = (\mathscr{A}, f)$. The *value for Max* in $\mathscr{G}$ from u is defined as $\text{val}^{\mathscr{G}}_{\text{Max}}(u) = \sup_\sigma \inf_\tau \mathbb{E}^u_{\sigma,\tau}(f)$, and symmetrically the value for Min is $\text{val}^{\mathscr{G}}_{\text{Min}}(u) = \inf_\tau \sup_\sigma \mathbb{E}^u_{\sigma,\tau}(f)$. Clearly enough, $\text{val}^{\mathscr{G}}_{\text{Max}}(u) \leq \text{val}^{\mathscr{G}}_{\text{Min}}(u)$.

We say that the game is *determined* if $\text{val}^{\mathscr{G}}_{\text{Max}}(u) = \text{val}^{\mathscr{G}}_{\text{Min}}(u)$, and in that case define the *value* of u in $\mathscr{G}$ as $\text{val}^{\mathscr{G}}(u)$. A strategy σ is optimal from u if $\text{val}^{\sigma}(u) = \text{val}^{\mathscr{G}}(u)$, and simply optimal if it is optimal from all vertices.

We say that the qualitative Ω is positionally determined if for all games with objective Ω, for all vertices v, there exists a pair of optimal positional strategies. We add the adjective purely to indicate that the strategy is pure, and uniformly to express that the same strategy can be used from all vertices. We define as expected the notion of half-positionally determined qualitative objective, and similarly for quantitative objectives.

Remark 10. *For stochastic reachability games we also often assume that the probabilistic transition function is*

$$\delta : V_{\text{Random}} \to \mathscr{D}(V),$$

and similarly strategies are functions $\sigma, \tau : V \to V$.

Back to the example in Figure 6.1, let us consider the stochastic reachability game $\mathscr{G} = (\mathscr{A}, \texttt{Reach}(\{v_7\}))$. Assume that Max and Min play the following pure positional strategies: $\sigma(v_0) = v_1$, $\sigma(v_2) = v_3$, $\sigma(v_5) = v_5$ and $\tau(v_6) = v_5$. Under such a strategy profile, starting in v_0, the probability to reach v_7 is $\mathbb{P}^{v_0}_{\sigma,\tau}(\texttt{Reach}(\{v_7\})) = \frac{2}{3}$. One can show that both strategies are optimal, and $\text{val}^{\mathscr{G}}_{\text{Max}}(v_0) = \text{val}^{\mathscr{G}}_{\text{Min}}(v_0) = \frac{2}{3}$.

The model with explicit actions

We now introduce a second model for stochastic arenas, using the notion of actions. This is the natural extension of the model used in Chapter 5 to two-player games. We let A be a (finite) set of actions, which is the set of choices the players can make at each step of the game.

Definition 19 (Stochastic arenas and games – explicit actions)**.** *A stochastic arena with explicit actions is* $\mathscr{A} = (G, V_{\text{Max}}, V_{\text{Min}}, V_{\text{Random}}, \Delta)$ *where*

- *G is a graph over the set of vertices V and $V = V_{\text{Max}} \uplus V_{\text{Min}}$,*
- *$\Delta : A \to \mathscr{D}(E)$ maps actions to distributions of edges.*

All notions are adapted (or extended from Chapter 5) in a natural way: for instance a strategy is a function $\sigma : \text{Paths} \to \mathscr{D}(A)$.

The transformations between the two models are transparent. The key difference is that the model with random vertices allows us to naturally define a parameter: the number of random vertices, which will be important in Section 6.5. A different parameter arises in the model with actions: the number of actions. At the end of the day, the choice of models is a matter of taste and technical convenience.

6.1 Fixed point characterisation and positional determinacy

In the same way as Martin's theorem implies (determinacy for (essentially) all (non-stochastic) games we consider in this book, the following theorem by Maitra and Sudderth establishes determinacy for stochastic games:

Theorem 71 (Determinacy of stochastic games with bounded quantitative objectives)**.** *Stochastic games with bounded measurable quantitative objectives are determined.*

Thanks to this theorem, the value is well defined in all games we consider.

Theorem 72 (Pure positional determinacy for stochastic reachability games)**.** *Stochastic reachability games are uniformly purely positionally determined.*

Note that Theorem 72 does not make any assumption on the game.

Before giving the proof of Theorem 72, we establish two preliminary results. Let $\mathscr{G}$ a stochastic reachability game. Let Y the set of functions $\mu : V \to [0,1]$, it is a lattice when equipped with the componentwise order. We define the operator $\mathbb{O}^{\mathscr{G}} : Y \to Y$ by:

$$\mathbb{O}^{\mathscr{G}}(\mu)(u) = \begin{cases} 1 & \text{if } u \in \text{Win}, \\ \max\{\mu(v) : u \to v \in E\} & \text{if } u \in V_{\text{Max}}, \\ \min\{\mu(v) : u \to v \in E\} & \text{if } u \in V_{\text{Min}}, \\ \sum_{v \in V} \delta(u)(v) \cdot \mu(v) & \text{if } u \in V_{\text{Random}}. \end{cases}$$

Since $\mathbb{O}^{\mathscr{G}}$ is monotonic, it has a least fixed point, which is also the least pre-fixed point.

Theorem 73. *Let $\mathscr{G}$ a stochastic reachability game. Then $\mathscr{G}$ is determined and the least fixed point of $\mathbb{O}^{\mathscr{G}}$ computes the values of $\mathscr{G}$. Furthermore, any uniform pure positional strategy τ for Min that satisfies*

$$u \in V_{\text{Min}} : \ \tau(u) \in \mathit{argmin} \left\{ \mathit{val}^{\mathscr{G}}(v) : u \to v \in E \right\}$$

is optimal.

Proof. To begin, let us recall that thanks to Kleene fixed point theorem (Theorem 4), the least fixed point of $\mathbb{O}^{\mathscr{G}}$ is computed as follows:

$$\forall u \in V, \ \mu_0(u) = 0 \quad ; \quad \mu_{k+1} = \mathbb{O}^{\mathscr{G}}(\mu_k).$$

We have $\mu_0 \leq \mu_1 \leq \ldots$, and since $\mathbb{O}^{\mathscr{G}}$ preserves suprema, $\mu^* = \lim_k \mu_k$ is the least fixed point of $\mathbb{O}^{\mathscr{G}}$. The crux here is to understand what are the values μ_k for $k = 0, 1, \ldots$. Let us define the truncated reachability objective:

$$\texttt{Reach}_{\leq k}(\text{Win}) = \{\pi : \exists i \leq k, \pi_i = \text{Win}\}.$$

A simple induction on k shows that μ_{k+1} is the values for $\texttt{Reach}_{\leq k}$ in $\mathscr{G}$ and that both players have optimal deterministic strategies for $\texttt{Reach}_{\leq k}$. For any $\varepsilon > 0$, let k be such that $\|\mu_{k+1} - \mu^*\|_\infty \leq \varepsilon$ and let σ be an optimal strategy of Max for $\texttt{Reach}_{\leq k}$. Then, for any u and any strategy τ of Min we have $\mathbb{P}^u_{\sigma,\tau}(\texttt{Reach}(\text{Win})) \geq \mathbb{P}^u_{\sigma,\tau}(\texttt{Reach}_{\leq k}(\text{Win})) \geq \mu^*(u) - \varepsilon$, so $\text{val}^{\mathscr{G}}_{\text{Max}}(u) \geq \mu^*(u)$.

To prove that $\text{val}^{\mathscr{G}}_{\text{Min}}(u) \leq \mu^*(u)$, consider any pure positional strategy τ for Min that satisfies

$$u \in V_{\text{Min}} : \ \tau(u) \in \text{argmin}\{\mu^*(v) : u \to v \in E\}.$$

Let σ be any strategy of Max. We prove by induction that for all k, the following holds:

$$\forall \sigma, \ \mathbb{E}^u_{\sigma,\tau}[\mu^*(\pi_k)] \leq \mu^*(u).$$

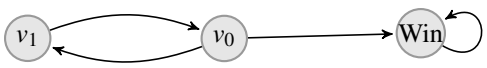


Figure 6.2: An example of a game where some argmax strategy is not optimal.

In words, if we consider the distribution of vertices obtained after k steps, then the expectation of μ^* on that distribution is smaller than or equal to $\mu^*(u)$. This is clear for $k = 0$, with the inequality being an equality. For $k > 0$, we show that for any vertex u we have $\mathbb{E}^u_{\sigma,\tau}[\mu^*(\pi_1)] \leq \mu^*(u)$: this is the one-step special case of the property above. We distinguish four cases:

- if $u = \text{Win}$, this is clear.
- if $u \in V_{\text{Max}}$, then for all $u \to v \in E$ we have $\mu^*(v) \leq \mu^*(u)$, implying the property.
- if $u \in V_{\text{Min}}$, then for $\tau(u) = u \to v$ we have $\mu^*(v) = \mu^*(u)$, implying the property.
- if $u \in V_{\text{Random}}$, then $\sum_{v \in V} \delta(u)(v) \cdot \mu^*(v) = \mu^*(u)$, again implying the property.

To do the induction step for higher k, note that for every v the conditional expected value $\mathbb{E}^u_{\sigma,\tau}[\mu^*(\pi_{k+1}) \mid \pi_1 = v]$ is equal to $\mathbb{E}^v_{\sigma',\tau}[\mu^*(\pi_k)] \leq \mu^*(v)$ for some σ', which gives $\mathbb{E}^u_{\sigma,\tau}[\mu^*(\pi_{k+1})] \leq \sum_v \mathbb{P}^u_{\sigma,\tau}(\pi_1 = v)\mu^*(v) = \mathbb{E}^u_{\sigma,\tau}[\mu^*(\pi_1)] \leq \mu^*(u)$ as claimed. In particular, for any σ and any k we get

$$\mathbb{P}^u_{\sigma,\tau}(\texttt{Reach}_{\leq k}(\text{Win})) \leq \mathbb{E}^u_{\sigma,\tau}[\mu^*(\pi_k)] \leq \mu^*(u),$$

because $\mu^*(\text{Win}) = 1$ and $\mu^* \geq 0$. Taking the limit when k goes to infinity, this yields $\mathbb{P}^u_{\sigma,\tau}(\texttt{Reach}(\text{Win})) \leq \mu^*(u)$. Since σ was arbitrary, we get $\text{val}^{\mathscr{G}}_{\text{Min}}(u) \leq \mu^*(u)$, which implies that μ^* is the value of the game and that the strategy τ is optimal. □

Let us raise an important point here: the set of values does not directly imply a pair of pure and positional optimal strategies. It is very tempting to define them as follows

$$\begin{aligned} u \in V_{\text{Max}} &: \ \sigma(u) \in \text{argmax} \left\{ \text{val}^{\mathscr{G}}(v) : u \to v \in E \right\}, \\ u \in V_{\text{Min}} &: \ \tau(u) \in \text{argmin} \left\{ \text{val}^{\mathscr{G}}(v) : u \to v \in E \right\}. \end{aligned}$$

and to claim that they are optimal. The optimality of τ follows from Theorem 73, but σ may not be optimal, as shown in Figure 6.2. In that example, all vertices have value 1, but the strategy $\sigma(v_0) = v_1$ is not optimal, although it can be obtained by the definition above. Some notion of progress is missing: the optimal strategy must make progress towards the target. Note that this example is actually not a stochastic game: the optimality claim for σ already fails for (non-stochastic) reachability games.

To construct optimal pure and positional strategies for Max, we need to be more precise. Let us fix $0 < \lambda < 1$ and consider the quantitative objective

$$\texttt{QuantitativeReach}(\rho) = \begin{cases} 0 & \text{if } \rho_i \neq \text{Win for all } i, \\ \lambda^i & \text{for } i \text{ the first index such that } \rho_i = \text{Win}. \end{cases}$$

It refines the reachability objective by quantifying the number of steps required to reach Win. A stochastic reachability game can be naturally interpreted as a limit of stochastic quantitative reachability games when $\lambda \to 1$. We now extend Theorem 73 to quantitative games. Let Y be the set of functions $\mu : V \to [0,1]$. We define the operator $\mathbb{O}^{\mathscr{G}} : Y \to Y$ by:

$$\mathbb{O}^{\mathscr{G}}(\mu)(u) = \begin{cases} 1 & \text{if } u \in \text{Win}, \\ \lambda \cdot \max\{\mu(v) : u \to v \in E\} & \text{if } u \in V_{\text{Max}}, \\ \lambda \cdot \min\{\mu(v) : u \to v \in E\} & \text{if } u \in V_{\text{Min}}, \\ \lambda \cdot \sum_{v \in V} \delta(u)(v) \cdot \mu(v) & \text{if } u \in V_{\text{Random}}. \end{cases}$$

Since $\mathbb{O}^{\mathscr{G}}$ is monotonic, it has a least fixed point, which is also the least pre-fixed point.

Theorem 74. *Let $\mathscr{G}$ a stochastic quantitative reachability game. Then $\mathscr{G}$ is uniformly purely positionally determined, $\mathbb{O}^{\mathscr{G}}$ has a unique fixed point, and this point computes the values of $\mathscr{G}$. Furthermore, any uniform pure positional strategies σ, τ that satisfy*

$$\begin{array}{l} \forall u \in V_{\text{Max}} : \ \sigma(u) \in \mathit{argmax}\left\{\mathit{val}^{\mathscr{G}}(v) : u \to v \in E\right\}, \\ \forall u \in V_{\text{Min}} : \ \tau(u) \in \mathit{argmin}\left\{\mathit{val}^{\mathscr{G}}(v) : u \to v \in E\right\}. \end{array}$$

are optimal.

Proof. Let us define the quantitative objective $\texttt{QuantitativeReach}_{\leq k}$:

$$\texttt{QuantitativeReach}_{\leq k}(\rho) = \begin{cases} 0 & \text{if } \rho_i \neq \text{Win for all } i \leq k, \\ \lambda^i & \text{for } i \leq k \text{ the first index such that } \rho_i = \text{Win}. \end{cases}$$

Let μ^* be any fixed point of $\mathbb{O}^{\mathscr{G}}$ and take any pure positional strategy σ such that $\sigma(u) \in \text{argmax}\{\mu^*(v) : u \to v \in E\}$ for all u. We prove by induction that for all $k \geq 1$, the following holds:

$$\forall \tau,\ \mathbb{E}^u_{\sigma,\tau}[\lambda^k \mu^*(\pi_k) + (1-\lambda)\sum_{i=1}^{k} \lambda^{i-1} \texttt{QuantitativeReach}_{\leq k-i}(\pi)] \geq \mu^*(u).$$

To shorten the notation, we write QR instead of QuantitativeReach. To prove the claim for $k = 1$ we distinguish four cases:

- if $u = \text{Win}$, then $\mu^*(u) = 1$.
- if $u \in V_{\text{Max}}$, then for $\sigma(u) = u \to v$ we have $\mu^*(v) = \lambda^{-1}\mu^*(u)$.
- if $u \in V_{\text{Min}}$, then for all $u \to v \in E$ we have $\mu^*(v) \geq \lambda^{-1}\mu^*(u)$.
- if $u \in V_{\text{Random}}$, then $\sum_{v \in V} \delta(u)(v) \cdot \mu^*(v) = \lambda^{-1}\mu^*(u)$.

Therefore $\mathbb{E}^u_{\sigma,\tau}[\lambda\mu(\pi_1) + (1-\lambda)\texttt{QR}_{\leq 0}(\pi)] \geq \mu^*(u)$ for all u. To prove the induction step, note that the claimed inequality is satisfied as equality if $u \in \text{Win}$. Otherwise, for every v there is τ' such that $\mathbb{E}^u_{\sigma,\tau}[\mu(\pi_{k+1}) \mid \pi_1 = v] = \mathbb{E}^v_{\sigma,\tau'}[\mu(\pi_k)]$, $\mathbb{E}^u_{\sigma,\tau}[\texttt{QR}_{\leq 0}(\pi) \mid \pi_1 = v] = 0$, and

$$\mathbb{E}^u_{\sigma,\tau}[\texttt{QR}_{\leq j}(\pi) \mid \pi_1 = v] = \lambda\,\mathbb{E}^v_{\sigma,\tau'}[\texttt{QR}_{\leq j-1}(\pi)]$$

for all $j \geq 1$. Hence, by the induction assumption we get

$$\mathbb{E}^u_{\sigma,\tau}[\lambda^{k+1}\mu(\pi_{k+1}) + (1-\lambda)\sum_{i=1}^{k+1}\lambda^{i-1}\mathtt{QR}_{\leq k+1-i}(\pi) \mid \pi_1 = v] \geq \lambda\mu^*(v).$$

Therefore,

$$\mathbb{E}^u_{\sigma,\tau}[\lambda^{k+1}\mu(\pi_{k+1}) + (1-\lambda)\sum_{i=1}^{k+1}\lambda^{i-1}\mathtt{QR}_{\leq k+1-i}(\pi)] \geq \lambda\mathbb{E}^u_{\sigma,\tau}[\mu(\pi_1)]$$

and the right-hand side is equal to $\mathbb{E}^u_{\sigma,\tau}[\lambda\mu(\pi_1) + (1-\lambda)\mathtt{QR}_{\leq 0}(\pi)] \geq \mu^*(u)$ because $\mathbb{E}^u_{\sigma,\tau}[\mathtt{QR}_{\leq 0}(\pi)] = 0$. To finish the proof, we want to show that the limit

$$\lim_{k\to\infty}\mathbb{E}^u_{\sigma,\tau}[\lambda^k\mu^*(\pi_k) + (1-\lambda)\sum_{i=1}^{k}\lambda^{i-1}\mathtt{QR}_{\leq k-i}(\pi)]$$

exists and is equal to $\mathbb{E}^u_{\sigma,\tau}[\mathtt{QR}(\pi)]$. Since $0 \leq \mu^*(u) \leq 1$ for all u, we get the equality $\lim_{k\to\infty}\mathbb{E}^u_{\sigma,\tau}[\lambda^k\mu^*(\pi_k)] = 0$. Since $\mathbb{E}^u_{\sigma,\tau}[\mathtt{QR}_{\leq k-i}(\pi)] \leq \mathbb{E}^u_{\sigma,\tau}[\mathtt{QR}(\pi)]$ for all k, i, we get

$$\mathbb{E}^u_{\sigma,\tau}[(1-\lambda)\sum_{i=1}^{k}\lambda^{i-1}\mathtt{QR}_{\leq k-i}(\pi)] \leq (1-\lambda^k)\mathbb{E}^u_{\sigma,\tau}[\mathtt{QR}(\pi)].$$

This shows that

$$\limsup_{k\to\infty}\mathbb{E}^u_{\sigma,\tau}[\lambda^k\mu^*(\pi_k) + (1-\lambda)\sum_{i=1}^{k}\lambda^{i-1}\mathtt{QR}_{\leq k-i}(\pi)] \leq \mathbb{E}^u_{\sigma,\tau}[\mathtt{QR}(\pi)].$$

Furthermore, the monotone convergence theorem gives the equality $\mathbb{E}^u_{\sigma,\tau}[\mathtt{QR}(\pi)] = \lim_{k\to\infty}\mathbb{E}^u_{\sigma,\tau}[\mathtt{QR}_{\leq k}(\pi)]$. Fix $\varepsilon > 0$ and k_0 such that $\mathbb{E}^u_{\sigma,\tau}[\mathtt{QR}(\pi)] - \varepsilon \leq \mathbb{E}^u_{\sigma,\tau}[\mathtt{QR}_{\leq k_0}(\pi)]$. Then, for every $k > k_0$ we have the bound

$$\mathbb{E}^u_{\sigma,\tau}[\sum_{i=1}^{k}\lambda^{i-1}\mathtt{QR}_{\leq k-i}(\pi)] \geq \mathbb{E}^u_{\sigma,\tau}[\sum_{i=1}^{k-k_0}\lambda^{i-1}\mathtt{QR}_{\leq k-i}(\pi)],$$

so $\mathbb{E}^u_{\sigma,\tau}[\sum_{i=1}^{k}\lambda^{i-1}\mathtt{QR}_{\leq k-i}(\pi)] \geq \frac{1-\lambda^{k-k_0}}{1-\lambda}(\mathbb{E}^u_{\sigma,\tau}[\mathtt{QR}(\pi)] - \varepsilon)$. By taking k to infinity, we get $\mathbb{E}^u_{\sigma,\tau}[\mathtt{QR}(\pi)] - \varepsilon \leq \liminf_{k\to\infty}\mathbb{E}^u_{\sigma,\tau}[\lambda^k\mu^*(\pi_k) + (1-\lambda)\sum_{i=1}^{k}\lambda^{i-1}\mathtt{QR}_{\leq k-i}(\pi)]$. Since ε was arbitrary, we get the existence of the limit. Therefore, we have shown that $\mathbb{E}^u_{\sigma,\tau}[\mathtt{QR}(\pi)] \geq \mu^*(u)$ for every τ. The proof for Min is analogous, proving that μ^* is the value of the game. Since the value is unique, $\mathbb{O}^{\mathscr{G}}$ has a unique fixed point. Moreover, our proof shows the optimality of the strategies (σ, τ) constructed as in the statement of the theorem. □

We can now use stochastic quantitative reachability games to prove the positional determinacy of reachability games.

Proof of Theorem 72. Let $\mathscr{G}$ be a stochastic reachability game. By Theorem 73, $\mathscr{G}$ has a value and an optimal pure positional strategy for Min. We need to show that Max also has such a strategy. To do so, for $0 < \lambda < 1$, let $\mathscr{G}_\lambda$ be the quantitative stochastic reachability game with parameter λ played on the same arena as $\mathscr{G}$. By Theorem 74, $\mathscr{G}_\lambda$ is positionally determined. Furthermore, we have $0 \leq \mathrm{val}^{\mathscr{G}_\lambda}(u) \leq 1$ for every u. Therefore, we can find an increasing sequence $\lambda_1 < \lambda_2 < \dots$ such that $\lim_{k\to\infty} \lambda_k = 1$, $\mathrm{val}^{\mathscr{G}_{\lambda_k}}$ converges to some point, and every game $\mathscr{G}_{\lambda_k}$ has the same optimal pure positional strategy σ of Max. We will show that σ is optimal in $\mathscr{G}$. Let τ be any strategy of Min. Observe that for every u and every k we have

$$\begin{array}{rcl} \mathrm{val}^{\mathscr{G}_{\lambda_k}}(u) & \leq & \mathbb{E}^u_{\sigma,\tau}[\texttt{QuantitativeReach}(\pi)] = \sum_{i=0}^{\infty} \lambda^i \mathbb{P}^u_{\sigma,\tau}(\texttt{Reach}_{=i}(\mathrm{Win})) \\ & \leq & \mathbb{P}^u_{\sigma,\tau}(\texttt{Reach}(\mathrm{Win})), \end{array}$$

where $\texttt{Reach}_{=i}(\mathrm{Win}) = \{\pi\colon \pi_i = \mathrm{Win}, \pi_j \neq \mathrm{Win} \text{ for } j < i\}$. By taking τ to be the optimal pure positional strategy for Min in $\mathscr{G}$, we get $\lim_{k\to\infty} \mathrm{val}^{\mathscr{G}_{\lambda_k}}(u) \leq \mathrm{val}^{\mathscr{G}}(u)$. Furthermore, note that for every $\xi : V \to [0,1]$, we have $\|\mathbb{O}^{\mathscr{G}_\lambda}(\xi) - \mathbb{O}^{\mathscr{G}}(\xi)\|_\infty \leq (1-\lambda)$. Since $\mathrm{val}^{\mathscr{G}_{\lambda_k}}$ is the fixed point of $\mathbb{O}^{\mathscr{G}_\lambda}$ by Theorem 73, we get $\|\mathrm{val}^{\mathscr{G}_{\lambda_k}} - \mathbb{O}^{\mathscr{G}}(\mathrm{val}^{\mathscr{G}_{\lambda_k}})\|_\infty \leq (1-\lambda_k)$ for all k. Hence, the continuity of $\mathbb{O}^{\mathscr{G}}$ implies that $\mathrm{val}^{\mathscr{G}_{\lambda_k}}$ converges to a fixed point of $\mathbb{O}^{\mathscr{G}}$. Therefore, by Theorem 74, we get $\lim_{k\to\infty} \mathrm{val}^{\mathscr{G}_{\lambda_k}} = \mathrm{val}^{\mathscr{G}}$, which implies that $\mathbb{P}^u_{\sigma,\tau}(\texttt{Reach}(\mathrm{Win})) \geq \mathrm{val}^{\mathscr{G}}(u)$ for any τ. □

6.2 Normalisation: stopping, binary, simple

6.2.1 Normalised games

We say that σ is almost-surely winning from u is for all strategies τ of Min, we have $\mathbb{P}^u_{\sigma,\tau}(W) = 1$. The almost-sure winning region is the set of vertices from where Max has an almost-surely winning strategy. Similarly, σ is positively winning from u if for all strategies τ of Min, we have $\mathbb{P}^u_{\sigma,\tau}(W) > 0$, and the positive winning region is the set of vertices from where Max has a positively winning strategy. Let us write $W_{>0}(\mathscr{G})$ for the positively winning region, and $W_{=1}(\mathscr{G})$ for the almost surely winning region.

Theorem 75. *There exists an algorithm for computing the positively winning region of stochastic reachability games in time $O(m)$.*

Analogously to attractor computations in reachability games (cf. Section 2.1), we define a one-step *positive probability* predecessor operator $\mathrm{Pre}_{>0}$ as follows: for $X \subseteq V$ we put

$$\begin{array}{rcl} \mathrm{Pre}_{>0}(X) & = & \{u \in V_{\mathrm{Max}} : \exists u \to v \in E, v \in X\} \\ & \cup & \{u \in V_{\mathrm{Min}} : \forall u \to v \in E,\ v \in X\} \\ & \cup & \{u \in V_{\mathrm{Random}} : \exists u \to v \in E,\ v \in X \text{ and } \delta(u)(u \to v) > 0\}. \end{array}$$

Let us define an operator on subsets of vertices:

$$X \mapsto \mathrm{Win} \cup \mathrm{Pre}_{>0}(X).$$

We note that this operator is monotonic when equipping the powerset of vertices with the inclusion preorder: if $X \subseteq X'$ then $\mathrm{Pre}_{>0}(X) \subseteq \mathrm{Pre}_{>0}(X')$. Hence Theorem 4 applies: this operator has a least fixed point computed by the following sequence: we let $X_0 = \emptyset$ and $X_i = \mathrm{Win} \cup \mathrm{Pre}_{>0}(X_{i-1})$. This constructs a sequence $(X_i)_{i\in\mathbb{N}}$ of non-decreasing subsets of V. Hence the sequence stabilises after at most $n-1$ steps, let us write $\mathrm{Attr}_{>0}(\mathrm{Win})$ for the limit. We have the following simple characterization of the positively winning set:

Lemma 79 (Characterisation of the positively winning set)**.** *Let $\mathscr{G}$ a stochastic reachability game. The positively winning region is the least fixed point of the operator $X \mapsto Win \cup Pre_{>0}(X)$.*

Proof. We show two properties:

- For all i, we have $X_i \subseteq W_{>0}(\mathscr{G})$.
- $W_{>0}(\mathscr{G})$ is a pre-fixed point of $\mathrm{Pre}_{>0}$.

The first property implies that $\mathrm{Attr}_{>0}(\mathrm{Win}) \subseteq W_{>0}(\mathscr{G})$, and the second the converse implication.

We prove the first property by induction on i. The case $i = 0$ is clear. Let $v \in X_{i+1}$, either $v \in \mathrm{Win}$ and then it is in $W_{>0}(\mathscr{G})$, or v in $\mathrm{Pre}_{>0}(X_i)$. In each of the three cases ($v \in V_{\mathrm{Max}}, V_{\mathrm{Min}}, V_{\mathrm{Random}}$) we have a positive probability to land in X_i at the next step. By induction hypothesis, $X_i \subseteq W_{>0}(\mathscr{G})$. Hence we have constructed a strategy winning positively from v, implying that $v \in W_{>0}(\mathscr{G})$.

We now prove the second property: $W_{>0}(\mathscr{G}) \subseteq \mathrm{Win} \cup \mathrm{Pre}_{>0}(W_{>0}(\mathscr{G}))$. Let $v \in W_{>0}(\mathscr{G})$, by definition either $v \in \mathrm{Win}$ or there exists a move which yields a positive probability of winning after the next step, implying that $v \in \mathrm{Pre}_{>0}(W_{>0}(\mathscr{G}))$. □

Similarly as Section 2.1, we can compute the least fixed point of the operator $X \mapsto \mathrm{Win} \cup \mathrm{Pre}_{>0}(X)$ in time and space $O(m)$. To avoid repetitions, let us note that we can actually directly reduce to non-stochastic reachability games. Let $\mathscr{G}$ a stochastic reachability game, we construct a (non-stochastic) reachability game $\mathscr{G}'$ as follows. Max controls the vertices of Max and the random vertices, and Min the vertices of Min. This means that for each random vertex, Max chooses an outgoing edge with positive probability. We claim that Max has a positively winning strategy in $\mathscr{G}$ from u if and only if Max has a winning strategy in $\mathscr{G}'$ from u. Indeed, the attractor computation in $\mathscr{G}'$ is exactly the positive attractor computation in $\mathscr{G}$.

Theorem 76. *There exists an algorithm for computing the almost surely winning region of stochastic reachability games in time $O(n \cdot m)$.*

Lemma 80 (Fixed point characterisation of the almost surely winning region for stochastic reachability games)**.** *Let $\mathscr{G}$ be a stochastic reachability game.*

- *If $Attr_{>0}(Win) = V$, then $W_{>0}(\mathscr{G}) = V$.*
- *If $Attr_{>0}(Win) \neq V$, we define $\mathscr{G}' = \mathscr{G} \setminus Attr_{\mathrm{Min}}(V \setminus Attr_{>0}(Win))$, then $W_{>0}(\mathscr{G}) = \mathsf{W}_{>0}(\mathscr{G}')$.*

Proof. We prove the first item. Let σ be a positively winning pure and positional strategy from $\mathrm{Attr}_{>0}(\mathrm{Win}) = V$. We argue that σ wins almost surely everywhere. Note that there exists $N \in \mathbb{N}$ and $\eta > 0$ such that σ ensures to reach Win within N steps with probability at least η. A play consistent with σ can be divided into infinitely many finite plays of length N. Since each of them has probability at least η to reach Win, the infinite play has probability 1 to reach Win by the Borel-Cantelli lemma.

We now look at the second item. We first prove that $\mathrm{Attr}_{\mathrm{Min}}(V \setminus \mathrm{Attr}_{>0}(\mathrm{Win})) \subseteq \mathsf{W}_{=0}(\mathscr{G})$, the set of vertices where Min can ensure to reach Win with probability 0. Let τ_a denote an attractor strategy ensuring to reach $V \setminus \mathrm{Attr}_{>0}(\mathrm{Win})$ from $\mathrm{Attr}_{\mathrm{Min}}(V \setminus \mathrm{Attr}_{>0}(\mathrm{Win}))$, and τ_c a strategy ensuring reach Win with probability 0. We construct the strategy τ as the disjoint union of τ_a and τ_c:

$$\tau(v) = \begin{cases} \tau_a(v) & \text{if } v \in \mathrm{Attr}_{\mathrm{Min}}(V \setminus \mathrm{Attr}_{>0}(\mathrm{Win})) \setminus (V \setminus \mathrm{Attr}_{>0}(\mathrm{Win})), \\ \tau_c(v) & \text{if } v \in V \setminus \mathrm{Attr}_{>0}(\mathrm{Win}). \end{cases}$$

Any play consistent with τ is first consistent with τ_a until reaching $V \setminus \mathrm{Attr}_{>0}(\mathrm{Win})$ and then is consistent with τ_c and stays there forever. Thus we have proved that $\mathrm{Attr}_{\mathrm{Min}}(V \setminus \mathrm{Attr}_{>0}(\mathrm{Win})) \subseteq \mathsf{W}_{=0}(\mathscr{G})$, implying $W_{>0}(\mathscr{G}) \subseteq V \setminus \mathrm{Attr}_{\mathrm{Min}}(V \setminus \mathrm{Attr}_{>0}(\mathrm{Win}))$.

We now show that $\mathsf{W}_{>0}(\mathscr{G}') \subseteq W_{>0}(\mathscr{G})$, which implies the converse inclusion. Consider a positively winning strategy from $\mathsf{W}_{>0}(\mathscr{G}')$ in $\mathscr{G}'$, it induces a positively winning strategy in $\mathscr{G}$. □

The algorithm is presented in pseudocode in Algorithm 6.1. For the complexity analysis, the algorithm performs at most n recursive calls and each of them involves two positive attractor computations, implying the time complexity $O(n \cdot m)$.

Algorithm 6.1: The quadratic time algorithm for the almost-sure winning region in stochastic reachability games.

```
Data: A stochastic reachability game.
Function Solve(𝒢):
    X ← Attr_{>0}(Win)
    if X = V then
        return V
    else
        Let 𝒢' = 𝒢 \ Attr_Min(V \ X)
        return Solve(𝒢')
```

Another point of view on this algorithm is to directly reduce the problem to Büchi games. Let $\mathscr{G}$ a stochastic reachability game, we construct a (non-stochastic) Büchi game $\mathscr{G}'$ as follows. Max controls the vertices of Max, and Min the vertices of Min and the random vertices. For each random vertex, Min can either choose an outgoing edge, or let Max choose one herself. If he chooses, the edge has priority 2, and if she does, the edge has priority 1. We add a self-loop over Win with priority 2. We claim that Max has an almost-surely winning strategy in $\mathscr{G}$ from u if and only if Max has a winning strategy in $\mathscr{G}'$ from u. Indeed, a closer look at the fixed point computations reveals that the algorithm for Büchi games in $\mathscr{G}'$ performs the exact same steps as Algorithm 6.1.

Definition 20 (Normalised stochastic reachability games)**.** *We say that a stochastic reachability game is normalised if there is a unique almost-surely winning vertex Win and a unique vertex Lose which is not positively winning, and both are sinks.*

Lemma 81 (Reduction to normalised games)**.** *Let $\mathcal{G}$ a stochastic reachability game, we can compute a normalised game $\mathcal{G}'$ in time $O(n \cdot m)$ so that for all vertices u from $\mathcal{G}$, we have $val^{\mathcal{G}}(u) = val^{\mathcal{G}'}(u)$ (unless it is 0 or 1, and then it is merged into Win or Lose).*

We compute both the almost-surely and positively winning regions, and replace the set of almost-surely vertices by Win, and the complement of the set of positively winning vertices by Lose.

6.2.2 Simple stochastic games

Definition 21 (Binary stochastic games)**.** *We say that a stochastic game is binary if:*

- *it is normalised,*
- *every vertex except for Win and Lose have out-degree two,*
- *for every random vertex $u \in V_{\text{Random}}$ we have $\delta(u) = \frac{1}{2} \cdot v + \frac{1}{2} \cdot v'$ for some v, v'.*

Lemma 82 (Reduction from stochastic games to simple stochastic games)**.** *Let $\mathcal{G}$ a stochastic reachability game, we can compute a simple stochastic game in polynomial time such that for all vertices u from $\mathcal{G}$, we have $val^{\mathcal{G}}(u) = val^{\mathcal{G}'}(u)$. The game $\mathcal{G}'$ has $O(n \cdot (\log(n) + k))$ vertices, where k is the number of bits required to represent probabilities in $\mathcal{G}$.*

Proof. Let $v \in V_{\text{Random}}$ a random vertex with k outgoing edges, with probabilities $p_1, \ldots, p_k$, leading to $v_1, \ldots, v_k$. We first introduce intermediary vertices in order to build a binary tree, whose leaves are $v_1 \ldots v_k$, root is v, and probabilities are set at each level of the tree in order to recover $p_1, \cdots p_k$ on the respective branches. This introduces $O(\log(k))$ fresh vertices, and is illustrated on an example on Figure 6.3.

The same process can be applied to vertices in V_{Max} and V_{Min} (without probabilities) to reduce to out-degree 2.

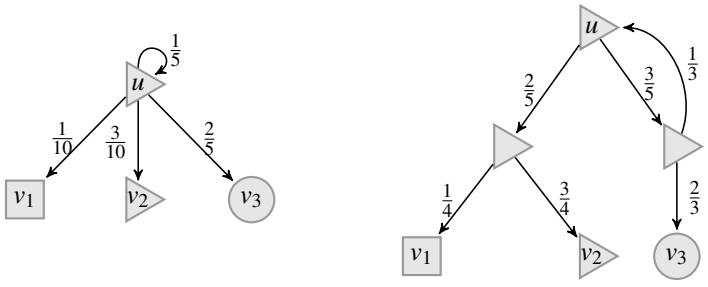


Figure 6.3: From general random vertices to binary ones.

It remains to explain how to simulate a distribution $\delta(u) = \frac{p}{q} \cdot v + \frac{q-p}{q} \cdot v'$ using only probabilities $\frac{1}{2}$. Let us denote the binary encodings (with most significant bit first) of p and $q-p$ as $a_1 \cdots a_t$ resp. $b_1 \cdots b_t$. We build the following gadget. The input vertex is u and for every $i \in [2, t+1]$, it has two exit edges with accumulated probabilities 2^{-i}. Now, if $a_i = 1$, one outgoing edge leads to v, and similarly if $b_i = 1$, then one outgoing edge leads to v'. The remaining edges are redirected to u.

The transformation is illustrated in Figure 6.4, with $p = 11$ and $q = 14$. The binary encodings are 1011 for p and 0011 for $p - q = 3$. For simplicity some vertices are represented several times to avoid intricate transitions. One can check that this gadget indeed simulates probabilities $\frac{p}{q}$ to v and $\frac{q-p}{q}$ to v'.

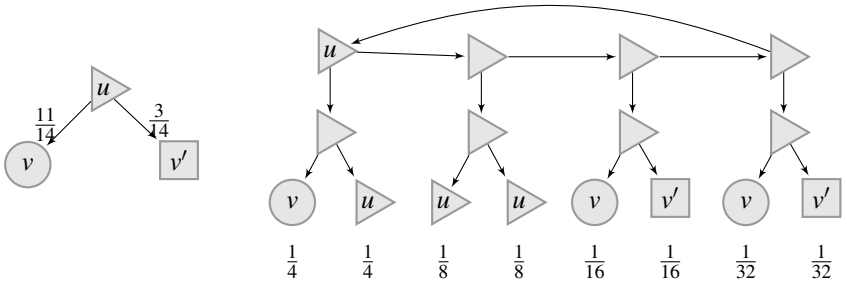


Figure 6.4: From binary random vertices to binary uniform ones.

□

6.2.3 Stopping games

The next lemmas will be useful in multiple proofs throughout the next sections. Note that if (σ, τ) is a pair of pure positional strategies of Max and Min, then the stochastic process induced on V by fixing (σ, τ) is a Markov chain in which the state Win is absorbing. We denote by $C^{\sigma,\tau}$ the set of recurrent states in this Markov chain and we put $C^{\sigma} = \bigcup_{\tau} C^{\sigma,\tau}$.

Lemma 83. *Let $\mathscr{G}$ be a stochastic reachability game and $\mu : V \to \mathsf{R}$ be any function. Take a pair of pure positional strategies (σ, τ) such that*

$$\forall u \in V_{\text{Max}} : \ \sigma(u) \in \mathit{argmax}\{\mu(v) : u \to v \in E\},$$
$$\forall u \in V_{\text{Min}} : \ \tau(u) \in \mathit{argmin}\{\mu(v) : u \to v \in E\}.$$

Then, we have the following two properties.

1. *If μ satisfies $\mu \leq \mathbb{O}^{\mathscr{G}}(\mu)$, $\mu(\mathit{Win}) = 1$, and $\mu(u) = 0$ for all $u \in C^{\sigma} \setminus \mathit{Win}$, then $\mathit{val}^{\mathscr{G}} \geq \mathit{val}^{\sigma} \geq \mu$.*
2. *If μ satisfies $\mu \geq \mathbb{O}^{\mathscr{G}}(\mu)$ and $\mu \geq 0$, then $\mathit{val}^{\mathscr{G}} \leq \mathit{val}^{\tau} \leq \mu$.*

Before giving a proof, we note that in this lemma the operator $\mathbb{O}^{\mathscr{G}}$ is defined not only on functions $\mu : V \to [0,1]$, but more generally on functions $\mu : V \to \mathsf{R}$.

Proof. We start by proving the first property. If $C^\sigma = V$, then the claim is trivial. Otherwise, let τ' be any pure positional strategy of Min and P be the transition matrix of the Markov chain obtained by fixing (σ,τ'). By definition of μ and σ we have $\mu \leq \mathbb{O}^{\mathscr{G}}(\mu) \leq P\mu$. Let Q denote the matrix obtained from P by removing the rows and columns indexed by $C^{\sigma,\tau'}$. Furthermore, let z be the vector defined as $z_u = P_{u,\mathrm{Win}}$ for all $u \in V \setminus C^{\sigma,\tau'}$. Then, the theory of absorbing Markov chains, see, e.g., [KS76, Chapter III], implies that $Q^N \to 0$, $(I-Q)^{-1} = I + Q + Q^2 + \ldots$, and $\mathbb{P}^u_{\sigma,\tau'}(\texttt{Reach}(\mathrm{Win})) = ((I-Q)^{-1}z)_u$ for every $u \in V \setminus C^{\sigma,\tau'}$. Let $\rho : (V \setminus C^{\sigma,\tau'}) \to \mathtt{R}$ denote the function obtained by restricting μ to the vertices in $V \setminus C^{\sigma,\tau'}$. Then, $\mu \leq P\mu$ implies that $\rho \leq z + Q\rho$ (note that here we use the fact that $\mu(\mathrm{Win}) = 1$ and $\mu(u) = 0$ for $u \in C^{\sigma,\tau'}$). Hence, for every $N \geq 1$ we get $\rho \leq (I + Q + \cdots + Q^N)z + Q^{N+1}\rho \to (I-Q)^{-1}z$. In particular, $\mathbb{P}^u_{\sigma,\tau'}(\texttt{Reach}(\mathrm{Win})) \geq \mu(u)$ for all u. Since τ' was arbitrary, positional determinacy implies that $\mathrm{val}^{\mathscr{G}} \geq \mathrm{val}^\sigma \geq \mu$.

The proof of the second property is analogous. Note that $\mu(\mathrm{Win}) \geq 1$ by definition. Let σ' be any pure positional strategy of Max and P be the transition matrix of the Markov chain obtained by fixing (σ',τ). If $C^{\sigma',\tau} = V$, then $\mathbb{P}^u_{\sigma',\tau}(\texttt{Reach}(\mathrm{Win})) = 0$ for all $u \neq \mathrm{Win}$ so $\mu(u) \geq \mathbb{P}^u_{\sigma',\tau}(\texttt{Reach}(\mathrm{Win}))$ for all u. Otherwise, let Q, z, ρ be defined as in the previous proof. By definition of τ, μ we get $\mu \geq P\mu$, which implies that $\rho \geq z + Q\rho$ (note that this implication only uses the fact that $\mu(\mathrm{Win}) \geq 1$ and $\mu \geq 0$). As previously, this inequality implies that $\mu(u) \geq \mathbb{P}^u_{\sigma',\tau}(\texttt{Reach}(\mathrm{Win}))$ for all u, which gives the claim. □

The second lemma characterises absorption probabilities in Markov chains with rational transition probabilities. We refer to [Sko21, AdMS21] for the proof.

Lemma 84. *Suppose that P is a square stochastic matrix with rational entries whose common denominator is D. Furthermore, let K denote the number of rows of P with at least two nonzero entries and suppose that $P_{vv} = 1$ for some index v. Consider a Markov chain with transition matrix given by P. Then, there exists a natural number $M \leq D^K$ such that for every state u the probability that the chain starting at u reaches v is a rational number of the form a_{uv}/M, where $a_{uv} \in \mathbb{N}$.*

To apply the lemma above to stochastic reachability games, we use the following notation. We suppose that D is the common denominator of all transition probabilities of $\mathscr{G}$ and that K is the number of *significant* random vertices, i.e., random vertices u such that $\delta(u)(v) > 0$ for at least two different v.

Corollary 18. *The value of the game $\mathscr{G}$ is a vector of rational numbers whose lowest common denominator is not greater than D^K.*

Proof. Let (σ,τ) be a pair of optimal pure positional strategies in $\mathscr{G}$. Then, $\mathrm{val}^{\mathscr{G}}(u) = \mathbb{P}^u_{\sigma,\tau}(\texttt{Reach}(\mathrm{Win}))$ for all u. Hence, the claim follows from Lemma 84. □

Remark 11. *We note that, as discussed in [Sko21, AdMS21], the bound $M \leq D^K$ in Lemma 84 and Corollary 18 can be improved in the following way: if we denote by d_u the lowest common denominator of the transition probabilities of a random vertex u, then $M \leq \prod_u d_u \leq D^K$. For the sake of simplicity, we use the bound $M \leq D^K$ in our proofs, but this can be replaced by $M \leq \prod_u d_u$.*

We now define the stopping stochastic reachability games. To do so, we suppose that the graph of the game is equipped with a special state Lose that is distinct from Win and that has only one outgoing edge, which is a loop going back to Lose. In particular, if a game reaches Lose, then it never leaves this state, and player Max cannot reach Win any more.

Definition 22 (Stopping stochastic reachability games)**.** *A stochastic reachability game is* stopping *if for every vertex $u \in V$ and every pair of pure positional strategies (σ, τ) we have $\mathbb{P}^u_{\sigma,\tau}(\texttt{Reach}(\{Win, Lose\})) = 1$.*

The following lemma shows that the condition of Definition 22 can be replaced by a seemingly weaker one.

Lemma 85. *A stochastic reachability game is stopping if for every vertex $u \in V$ and every pair of pure positional strategies (σ, τ) we have $\mathbb{P}^u_{\sigma,\tau}(\texttt{Reach}(\{Win, Lose\})) > 0$.*

Proof. Suppose that a Markov chain defined by (σ, τ) can reach the set $\{\text{Win}, \text{Lose}\}$ from any starting state u. Since the states Win, Lose are absorbing, this implies that they are the only recurrent states of the chain, so $\mathbb{P}^u_{\sigma,\tau}(\texttt{Reach}(\{\text{Win}, \text{Lose}\})) = 1$. □

Theorem 77 (Fixed point characterisation for stopping simple stochastic games)**.** *Let $\mathscr{G}$ be a stopping stochastic reachability game. Then, the operator $\mathbb{O}^{\mathscr{G}}$ has a unique fixed point μ such that $\mu(Lose) = 0$, namely $\mu = val^{\mathscr{G}}$. Moreover, pure positional strategies (σ, τ) of Max and Min are optimal if and only if they satisfy*

$$\forall u \in V_{\text{Max}} : \ \sigma(u) \in argmax\left\{val^{\mathscr{G}}(v) : u \to v \in E\right\},$$
$$\forall u \in V_{\text{Min}} : \ \tau(u) \in argmin\left\{val^{\mathscr{G}}(v) : u \to v \in E\right\}.$$

Proof. Let μ be any fixed point of $\mathbb{O}^{\mathscr{G}}$ such that $\mu(\text{Lose}) = 0$ (by Theorem 73, $\text{val}^{\mathscr{G}}$ is one such point). Let σ, τ be a pair of pure positional strategies such that $\sigma(u) \in \text{argmax}\{\mu(v) : u \to v \in E\}$ for all $u \in V_{\text{Max}}$ and $\tau(u) \in \text{argmin}\{\mu(v) : u \to v \in E\}$ for all $u \in V_{\text{Min}}$. Since the game is stopping, we have $C^\sigma = \{\text{Win}, \text{Lose}\}$. In particular, the pair (μ, σ) satisfies the conditions of Lemma 83 and therefore $\mu \leq \text{val}^{\mathscr{G}}$. Likewise, the pair (μ, τ) satisfies the conditions of Lemma 83, so $\mu \geq \text{val}^{\mathscr{G}}$. Thus, $\mu = \text{val}^{\mathscr{G}}$ and Lemma 83 show that σ, τ are optimal. Suppose now that τ is a pure positional strategy of Min such that $\tau(u) \notin \text{argmin}\left\{\text{val}^{\mathscr{G}}(v) : u \to v \in E\right\}$ for some $u \in V_{\text{Min}}$. Then, $\text{val}^{\mathscr{G}} \neq \mathbb{O}^{\mathscr{G}[\tau]}(\text{val}^{\mathscr{G}})$. Hence, by Theorem 73, $\text{val}^{\tau} \neq \text{val}^{\mathscr{G}}$ and τ is not optimal. The proof for player Max is analogous. □

Lemma 86 (Reduction to stopping games)**.** *Let $\mathscr{G}$ a stochastic reachability game. We can compute a stopping game $\mathscr{G}'$ such that any pair of optimal pure stationary strategies in $\mathscr{G}'$ is also optimal in $\mathscr{G}$. Moreover, for all vertices u in $\mathscr{G}$, we have $val^{\mathscr{G}}(u) > \frac{1}{2}$ if and only if $val^{\mathscr{G}'}(u) > \frac{1}{2}$.*

Proof. For every $0 < \varepsilon < 1$, we construct the game $\mathscr{G}^\varepsilon$ in the following way. First, we add a state Lose to $\mathscr{G}$. Then, for every edge $u \to v \in E$ such that $u \neq \text{Lose}$ we add a new random vertex w_{uv} to the graph. We remove the edge $u \to v$ and add the edges $u \to w_{uv}$, $w_{uv} \to v$, and $w_{uv} \to \text{Lose}$, as in Figure 6.5. We also define the transition function δ^ε

of the new game as follows. For every random state u of the original game we put $\delta^\varepsilon(u)(w_{uv}) = \delta(u)(v)$ for all v, and for every new state w_{uv} we put $\delta^\varepsilon(w_{uv})(\text{Lose}) = \varepsilon$ and $\delta^\varepsilon(w_{uv})(v) = 1-\varepsilon$.

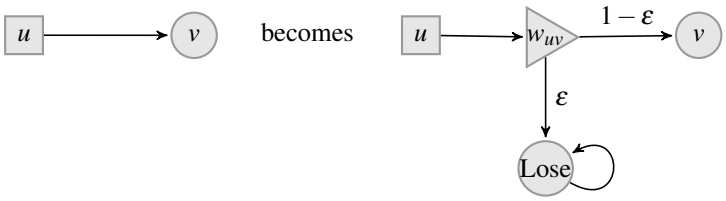


Figure 6.5: From reachability games to stopping games.

We also consider the game $\mathscr{G}^0$ obtained by putting $\varepsilon = 0$, i.e., a game that is identical to the original game $\mathscr{G}$ except for the fact that we added an isolated vertex Lose to the graph and we put a dummy vertex on every edge of the graph. In particular, the game $\mathscr{G}^0$ has the same optimal pure stationary strategies as $\mathscr{G}$. Furthermore, every vertex of $\mathscr{G}$ has the same value in $\mathscr{G}$ and in $\mathscr{G}^0$. Let D denote the lowest common denominator of all transition probabilities in $\mathscr{G}$. Furthermore, let K be the number of random vertices in $\mathscr{G}$. By Corollary 18, the values of the game $\mathscr{G}^0$ are rational numbers with lowest common denominator not higher than D^K. Let $\varepsilon = \min\{\frac{1}{4}, D^{-2K}\}$. We will show that $\mathscr{G}' = \mathscr{G}^\varepsilon$ satisfies the claim.

To begin, note that $\mathscr{G}'$ is stopping. Indeed, if we fix any pair of pure positional strategies (σ,τ) in $\mathscr{G}'$ and suppose that the resulting Markov starts at some state u, then this chain can reach the set $\{\text{Win}, \text{Lose}\}$, so $\mathscr{G}'$ is stopping by Lemma 85. Furthermore, note that if ξ is any vector with entries in $[0,1]$ and such that $\xi_{\text{Lose}} = 0$, then $\mathbb{O}^{\mathscr{G}'}(\xi) \leq \mathbb{O}^{\mathscr{G}^0}(\xi) \leq \varepsilon + \mathbb{O}^{\mathscr{G}'}(\xi)$, where the notation $\varepsilon + \mathbb{O}^{\mathscr{G}'}(\xi)$ means that we add ε to all coordinates of $\mathbb{O}^{\mathscr{G}'}(\xi)$. Let $\xi = \text{val}^{\mathscr{G}'}$ and suppose that σ, τ are optimal pure positional strategies of Max and Min in $\mathscr{G}'$. We want to show that (σ,τ) are also optimal in $\mathscr{G}^0$. By Theorem 77, we have $\sigma(u) \in \text{argmax}\{\xi(v) : u \to v \in E\}$ for all $u \in V_{\text{Max}}$ and $\tau(u) \in \text{argmin}\{\xi(v) : u \to v \in E\} = \text{argmin}\{\varepsilon + \xi(v) : u \to v \in E\}$ for all $u \in V_{\text{Min}}$. Since $\xi \geq 0$, Lemma 83 implies that $\text{val}^{\mathscr{G}^0} \leq \varepsilon + \xi$ and that the same is true if Min uses τ in $\mathscr{G}^0$. Suppose that u is any state in $\mathscr{G}^0$ different than Win and that there exists a pure positional strategy τ' of Min such that u is recurrent in the Markov chain induced by fixing (σ,τ') in $\mathscr{G}^0$. In this situation, the chain starting at u cannot reach Win. Hence, if we consider the Markov chain induced by fixing (σ,τ') in $\mathscr{G}'$, then this chain also cannot reach Win from u. By optimality of σ this implies that $\xi_u = 0$. Since $\xi_{\text{Win}} = 1$, Lemma 83 implies that $\text{val}^{\mathscr{G}'} \leq \text{val}^{\mathscr{G}^0}$ and the same is true if Max uses σ in $\mathscr{G}^0$. Hence, we established the inequality $\text{val}^{\mathscr{G}'} \leq \text{val}^{\mathscr{G}^0} \leq \varepsilon + \text{val}^{\mathscr{G}'}$. Let u be any state. Note that, by definition of ε, the interval $[\text{val}^{\mathscr{G}'}(u), \varepsilon + \text{val}^{\mathscr{G}'}(u)]$ can contain at most one rational number of denominator at most D^K. Moreover, $\text{val}^{\mathscr{G}^0}(u)$ is such a number. If we now consider the one-player game obtained from $\mathscr{G}^0$ by fixing τ, then Corollary 18 shows that the values of this game are also rational numbers of denominator at most D^K. Moreover, as shown above, these values are between $\text{val}^{\mathscr{G}^0}$ and $\varepsilon + \text{val}^{\mathscr{G}'}$. Hence,

they must be equal to $\mathrm{val}^{\mathscr{G}^0}$ and τ is optimal in $\mathscr{G}^0$. By the same reasoning, σ is optimal in $\mathscr{G}^0$. To show the last claim, note that $\mathrm{val}^{\mathscr{G}'}(u) > \frac{1}{2}$ implies $\mathrm{val}^{\mathscr{G}^0}(u) > \frac{1}{2}$. Conversely, since $\mathrm{val}^{\mathscr{G}^0}(u)$ has denominator at most D^K, if $\mathrm{val}^{\mathscr{G}^0}(u) > \frac{1}{2}$, then $\mathrm{val}^{\mathscr{G}^0}(u) > \varepsilon + \frac{1}{2}$, so $\mathrm{val}^{\mathscr{G}'}(u) > \frac{1}{2}$. □

6.3 A value iteration algorithm

We now present a value iteration algorithm for stochastic reachability games. This algorithm will rely on a appropriate rounding procedure. In this procedure, we want to approximate a given rational number from above by a number with bounded denominator. To make this more precise, given a positive natural number M, we denote by $\mathscr{Q}_M$ the set

$$\mathscr{Q}_M = \left\{ \frac{p}{q} : \, p \in \{0,1,\ldots,M\}, q \in \{1,2,\ldots,M\} \right\}.$$

Theorem 78. *Given a rational number $\alpha \in [0,1]$ and $M \geq 1$ we can find the smallest number $\beta \in \mathscr{Q}_M$ that satisfies $\alpha \leq \beta$ in $O(\log M)$ arithmetic operations.*

Proof. Let β be the smallest number in $\mathscr{Q}_M$ that satisfies $\alpha \leq \beta$. Note that if γ is any number in $\mathscr{Q}_M$, then it is trivial to decide if $\beta \leq \gamma$. Indeed, if $\alpha \leq \gamma$, then $\beta \leq \gamma$ and if $\alpha > \gamma$, then $\beta > \gamma$. Hence, the exact value of β can be found in $O(\log M)$ operations using the rational search technique described in [KM03]. □

We denote by $\textsc{RoundUp}(\mu, M)$ the procedure that takes as an input $M \geq 1$ and a function $\mu : V \to [0,1] \cap \mathbb{Q}$ and rounds up all values of μ using the procedure from Theorem 78. The pseudocode of the value iteration algorithm for stochastic reachability games is given in Algorithm 6.2.

Algorithm 6.2: The value iteration algorithm.

```
Data: A stochastic reachability game.
Choose μ such that μ ≤ val^𝒢 and μ ≤ 𝕆^𝒢(μ)
repeat
    μ ← RoundUp(𝕆^𝒢(μ), D^K)
until 𝕆^𝒢(μ) = μ
return μ
```

To analyze the algorithm, we use the following lemmas. To simplify the notation, we denote $\mathbb{B}(\mu) = \textsc{RoundUp}(\mathbb{O}^{\mathscr{G}}(\mu), D^K)$ for every function $\mu : V \to [0,1] \cap \mathbb{Q}$.

Lemma 87. *The operator $\mu \mapsto \mathbb{B}(\mu)$ is monotonic.*

Proof. Suppose that $0 \leq \alpha_1 \leq \alpha_2 \leq 1$ are two rational numbers. If $\beta_i \in \mathscr{Q}_M$ is the smallest number such that $\beta_i \geq \alpha_i$ for $i \in \{1,2\}$, then we have $\beta_2 \geq \alpha_2 \geq \alpha_1$, so $\beta_2 \geq \beta_1$ by definition. Hence, the operator $\mu \to \textsc{RoundUp}(\mu, D^K)$ is monotonic. Therefore, $\mathbb{B}(\mu)$ is monotonic as a composition of monotonic operators. □

Lemma 88. *If $\mu_0 \leq \mathbb{O}^{\mathscr{G}}(\mu)$, then the sequence $\mu_{k+1} = \mathbb{B}(\mu_k)$ produced by the value iteration algorithm is nondecreasing, $\mu_k \leq \mu_{k+1}$.*

Proof. We have $\mu_0 \leq \mathbb{O}^{\mathscr{G}}(\mu_0) \leq \mathbb{B}(\mu_0) = \mu_1$ by the assumption. Hence, the monotonicity of $\mathbb{B}$ gives $\mu_k \leq \mu_{k+1}$ for all k. □

Theorem 79 (Starting condition for value iteration)**.** *Suppose that $\mu_0 = 0$ or $\mu_0 = val^{\sigma}$ for some pure positional strategy σ of Max. Then, μ_0 satisfies the initial condition of value iteration, i.e., $\mu_0 \leq val^{\mathscr{G}}$ and $\mu_0 \leq \mathbb{O}^{\mathscr{G}}(\mu)$.*

Proof. Both claims are trivial if $\mu_0 = 0$. Moreover, $\mathrm{val}^{\sigma} \leq \mathrm{val}^{\mathscr{G}}$ by positional determinacy. To prove the last claim, note that $\mathrm{val}^{\sigma} = \mathbb{O}^{\mathscr{G}[\sigma]}(\mathrm{val}^{\sigma})$ by Theorem 73 and $\mathbb{O}^{\mathscr{G}[\sigma]}(\mathrm{val}^{\sigma}) \leq \mathbb{O}^{\mathscr{G}}(\mathrm{val}^{\sigma})$ by definition of $\mathbb{O}^{\mathscr{G}}$. □

Theorem 80 (Correctness of value iteration)**.** *The value iteration algorithm halts in at most nD^{2K} iterations and outputs a function μ such that $\mu = val^{\mathscr{G}}$.*

Proof. Let μ_0 be such that $\mu_0 \leq \mathrm{val}^{\mathscr{G}}$ and $\mu_0 \leq \mathbb{O}^{\mathscr{G}}(\mu_0)$. By Lemma 88, the sequence $\mu_{k+1} = \mathbb{B}(\mu_k)$ is nondecreasing. Furthermore, by Theorem 73, $\mathrm{val}^{\mathscr{G}}$ is a fixed point of $\mathbb{O}^{\mathscr{G}}$ and by Corollary 18 it is also a fixed point of $\textsc{RoundUp}(x, D^K)$, so it is a fixed point of $\mathbb{B}$. Hence, by the monotonicity of $\mathbb{B}$ we have $\mu_k \leq \mathrm{val}^{\mathscr{G}}$ for all k. In particular, if the algorithm halts, then it outputs a fixed point of $\mathbb{O}^{\mathscr{G}}$ that is not higher than $\mathrm{val}^{\mathscr{G}}$. Therefore, by Theorem 73, the algorithm outputs $\mathrm{val}^{\mathscr{G}}$ when it halts. To show that the algorithm halts, consider a sequence ξ_k defined as $\xi_0 = \mu_0$ and $\xi_{k+1} = \mathbb{O}^{\mathscr{G}}(\xi_k)$, i.e., a sequence produced by value iteration without rounding. The monotonicity of $\mathbb{O}^{\mathscr{G}}$ shows that ξ_k is nondecreasing, $\xi_k \leq \xi_{k+1}$, and Theorem 73 shows that $\xi_k \leq \mathrm{val}^{\mathscr{G}}$ for all k. In particular, the sequence μ_k converges to $\mu^* = \sup_k \mu_k$ (where we apply the supremum to all coordinates of the sequence), ξ_k converges to $\xi^* = \sup_k \xi_k$, and we have $\mu^* \leq \mathrm{val}^{\mathscr{G}}$ and $\xi^* \leq \mathrm{val}^{\mathscr{G}}$. Furthermore, by induction we have $\xi_k \leq \mu_k$ for all k. Indeed, $\xi_0 \leq \mu_0$ by definition, and if $\xi_k \leq \mu_k$, then $\xi_{k+1} = \mathbb{O}^{\mathscr{G}}(\xi_k) \leq \mathbb{O}^{\mathscr{G}}(\mu_k) \leq \mathbb{B}(\mu_k) = \mu_{k+1}$. Hence, we get $\xi^* \leq \mu^* \leq \mathrm{val}^{\mathscr{G}}$. Moreover, by continuity of $\mathbb{O}^{\mathscr{G}}$, the sequence ξ_k also converges to $\mathbb{O}^{\mathscr{G}}(\xi^*)$, which means that ξ^* is a fixed point of $\mathbb{O}^{\mathscr{G}}$. Since $\xi^* \leq \mathrm{val}^{\mathscr{G}}$, Theorem 73 shows that $\xi^* = \mu^* = \mathrm{val}^{\mathscr{G}}$. To finish the proof, note that μ_k is a vector of rational numbers in $[0,1]$ with denominators bounded by D^K. Since there are only finitely many such numbers, the sequence (μ_k) stabilizes after finitely many steps, i.e., there exists the smallest k^* such that $\mu_k = \mu^* = \mathrm{val}^{\mathscr{G}}$ for all $k \geq k^*$. Hence, the algorithm halts in k^* iterations. To bound k^* note that for $k < k^*$ we have $\mu_k \leq \mu_{k+1}$ and $\mu_k \neq \mu_{k+1}$. Therefore, at least one coordinate of μ_{k+1} is strictly larger than in μ_k. Since these coordinates are rational numbers of denominator at most D^K, they differ by at least D^{-2K}. Given that the coordinates of μ_k are in $[0,1]$, we conclude that $k^* \leq nD^{2K}$. □

6.4 A strategy improvement algorithm

Let us consider a stochastic reachability game $\mathscr{G}$ and set as a goal to construct an optimal strategy for Max. The key idea behind strategy improvement is to use val^{σ} to improve the strategy σ by *switching edges*, which is an operation that creates a new

strategy. This involves defining the notion of *improving edges*: let us consider a vertex $u \in V_{\text{Max}}$, we say that $e : u \to v$ is an *improving edge* if

$$\text{val}^{\sigma}(v) > \text{val}^{\sigma}(u).$$

Intuitively: according to val^{σ}, playing e is better than playing $\sigma(u)$.

Given a strategy σ and a set of improving edges S (for each $u \in V_{\text{Max}}$, S contains at most one outgoing edge of u), we write $\sigma[S]$ for the strategy

$$\sigma[S](u) = \begin{cases} e & \text{if there exists } e = u \to v \in S, \\ \sigma(v) & \text{otherwise.} \end{cases}$$

The difficulty is that an edge being improving does not mean that it is a better move than the current one in any context, but only according to the value function val^{σ}, so it is not clear that $\sigma[S]$ is better than σ. Strategy improvement algorithms depend on the following two principles:

- **Progress**: updating a strategy using improving edges is a strict improvement,
- **Optimality**: a strategy which does not have any improving edges is optimal.

The pseudocode of the algorithm is given in Algorithm 6.3. The algorithm is non-deterministic, in the sense that both the initial strategy and at each iteration, the choice of improving edge can be chosen arbitrarily. A typical choice, called the "greedy all-switches" rule, choosing for each $u \in V_{\text{Max}}$ a maximal improving edge, meaning

$$\text{argmax} \left\{ \text{val}^{\sigma}(v) : u \to v \in E \right\}.$$

Algorithm 6.3: The strategy improvement algorithm.

```
Choose an initial strategy σ_0 for Max
for i = 0, 1, 2, ... do
    Compute val^{σ_i} and the set of improving edges
    if σ_i does not have improving edges then
        return σ_i
    Choose a non-empty set S_i of improving edges
    σ_{i+1} ← σ_i[S_i]
```

Before proving the correctness of this algorithm, we show that computing val^{σ_i} at each iteration can be done in polynomial time. Note that the opposite problem (computing val^{τ} given τ) is the reachability problem in a MDP as discussed in Chapter 5. In particular, Theorem 67 gives a polynomial-time algorithm for this problem based on linear programming. Finding val^{σ} given σ is similar.

Lemma 89. *Let σ be a pure positional strategy of player Max. Then, val^σ is the solution of the linear program*

$$\begin{array}{lll} \textit{maximise} & \sum_{u\in V} x_u & \\ \textit{subject to} & x_u = 1 & \textit{if } u = \textit{Win}, \\ & x_u = x_{\sigma(u)} & \textit{if } u \in V_{\text{Max}}, \\ & x_u \le x_v & \textit{if } u \in V_{\text{Min}},\ u \to v \in E, \\ & x_u = \sum_v \delta(u)(v) x_v & \textit{if } u \in V_{\text{Random}}, \\ & x_u = 0 & \textit{if } u \in V \setminus W_{>0}(\mathscr{G}^{[\sigma]}). \end{array}$$

In particular, val^σ can be found in polynomial time.

Proof. By applying Theorem 73 to $\mathbb{O}^{\mathscr{G}[\sigma]}$ we get that val^σ is a feasible solution of the program. Moreover, note that every feasible solution x satisfies $x \le \mathbb{O}^{\mathscr{G}[\sigma]}(x)$, $x_{\text{Win}} = 1$, and $x_u = 0$ for all $u \in V \setminus W_{>0}(\mathscr{G}^{[\sigma]})$. Since $(C^\sigma \setminus \text{Win}) \subset (V \setminus W_{>0}(\mathscr{G}^{[\sigma]}))$, Lemma 83 shows that $x \le \text{val}^\sigma$. Hence, val^σ can be found by computing $W_{>0}(\mathscr{G}^{[\sigma]})$ (Theorem 75) and solving the linear program. □

Let us write $\sigma \le \sigma'$ if for all vertices u we have $\text{val}^\sigma(u) \le \text{val}^{\sigma'}(u)$, and $\sigma < \sigma'$ if additionally $\neg(\sigma' \le \sigma)$. We make the following observation.

Lemma 90. *Let σ be any strategy of Max and S a set of improving edges. We let $\sigma' = \sigma[S]$. Then, we have $val^\sigma \le \mathbb{O}^{\mathscr{G}[\sigma']}(val^\sigma) \le \mathbb{O}^{\mathscr{G}}(val^\sigma)$. Furthermore, the inequality $val^\sigma(u) \le \mathbb{O}^{\mathscr{G}}(val^\sigma)(u)$ is strict if and only if u is a vertex of Max that has at least one improving edge. Likewise, the inequality $val^\sigma(u) \le \mathbb{O}^{\mathscr{G}[\sigma']}(val^\sigma)(u)$ is strict if u has an outgoing edge in S. Moreover, if S is constructed by the greedy all-switches rule, then $\mathbb{O}^{\mathscr{G}[\sigma']}(val^\sigma) = \mathbb{O}^{\mathscr{G}}(val^\sigma)$.*

Proof. By Theorem 73 we have $\text{val}^\sigma = \mathbb{O}^{\mathscr{G}[\sigma]}(\text{val}^\sigma)$. Hence, the inequality $\text{val}^\sigma(u) \le \mathbb{O}^{\mathscr{G}}(\text{val}^\sigma)(u)$ follows from the definition of $\mathbb{O}^{\mathscr{G}}$ and this inequality is strict if and only if u is a vertex of Max that has at least one improving edge. Furthermore, the definition of σ' implies that $\mathbb{O}^{\mathscr{G}[\sigma']}(\text{val}^\sigma)$ is sandwiched between val^σ and $\mathbb{O}^{\mathscr{G}}(\text{val}^\sigma)$ and that it satisfies $\text{val}^\sigma(u) < \mathbb{O}^{\mathscr{G}[\sigma']}(\text{val}^\sigma)(u)$ for every u that has an outgoing edge in S. The last claim follows from the definition of the greedy all-switches rule. □

Theorem 81 (Progress property for the strategy improvement)**.** *Let σ a strategy and S a set of improving edges. We let $\sigma' = \sigma[S]$. Then $\sigma < \sigma'$.*

Proof. By Lemma 90, we have $\text{val}^\sigma \le \mathbb{O}^{\mathscr{G}[\sigma']}(\text{val}^\sigma)$ and this inequality is strict for at least one coordinate. Hence, by Theorem 73, $\text{val}^{\sigma'} \ne \text{val}^\sigma$. It remains to prove that $\text{val}^{\sigma'} \ge \text{val}^\sigma$. To do so, we use Lemma 83. Denote $\mu = \text{val}^\sigma$, let τ be any pure positional strategy of Min and let P be the transition matrix of the Markov chain induced by fixing (σ', τ). By definition, we have $\mu \le \mathbb{O}^{\mathscr{G}[\sigma']}(\mu) \le P\mu$. Suppose that $C \ne \{\text{Win}\}$ is a recurrent class of this Markov chain. Then, this class has a stationary distribution (see, e.g., [KS76, Chapter V]). In other words, there exists a nonnegative vector $\pi \ge 0$ such that $\pi_u > 0$ for $u \in C$, $\pi_u = 0$ for $u \notin C$, and $\pi^T P = \pi$. Since $\pi^T \mu \le \pi^T P \mu = \pi^T \mu$ we get $\mu(u) = \mathbb{O}^{\mathscr{G}[\sigma']}(\mu)(u) = (P\mu)_u$ for all $u \in C$. In particular, the strategies σ' and σ are the same on all vertices in $C \cap V_{\text{Max}}$. Hence, C is also a recurrent class in the

Markov chain obtained by fixing (σ,τ) and so $\mathrm{val}^{\sigma}(u) = 0$ for every $u \in C$. Since τ was arbitrary, we get $\mu(u) = 0$ for all $u \in C^{\sigma'} \setminus \{\mathrm{Win}\}$. We also have $\mu(\mathrm{Win}) = 1$ and by Lemma 83 we conclude that $\mu \leq \mathrm{val}^{\sigma'}$. □

Theorem 82 (Optimality property for the strategy improvement)**.** *Let σ be a strategy that has no improving edges, then σ is optimal.*

Proof. Let σ be a strategy that has no improving edges. Then, by Lemma 90 we have $\mathrm{val}^{\sigma} = \mathbb{O}^{\mathscr{G}}(\mathrm{val}^{\sigma})$, and so, by Theorem 73, $\mathrm{val}^{\sigma} \geq \mathrm{val}^{\mathscr{G}}$, which gives $\mathrm{val}^{\sigma} = \mathrm{val}^{\mathscr{G}}$. □

Theorem 83 (Comparison of strategy improvement and value iteration algorithms)**.** *Let us write $\sigma_0 < \sigma_1 < \sigma_2 < \cdots$ for the sequence of positional strategies in an execution of the strategy improvement algorithm with the greedy all-switches rule over $\mathscr{G}$ and $\mu_0 \leq \mu_1 \leq \mu_2 \leq \cdots$ for the sequence of functions computed by the corresponding value iteration algorithm over $\mathscr{G}$ initialised at val^{σ_0}: for all k, we have $\mu_0 = val^{\sigma_0}$ and $\mu_{k+1} = \mathbb{B}(\mu_k)$.*

Then for all k, we have $\mu_k \leq val^{\sigma_k}$.

Proof. For every k we have $\mathrm{val}^{\sigma_k} \leq \mathrm{val}^{\sigma_{k+1}}$. Hence, by Theorem 73 and the monotonicity of $\mathbb{O}^{\mathscr{G}[\sigma_{k+1}]}$ we get $\mathbb{O}^{\mathscr{G}[\sigma_{k+1}]}(\mathrm{val}^{\sigma_k}) \leq \mathrm{val}^{\sigma_{k+1}}$. Furthermore, by Lemma 90 we have $\mathbb{O}^{\mathscr{G}[\sigma_{k+1}]}(\mathrm{val}^{\sigma_k}) = \mathbb{O}^{\mathscr{G}}(\mathrm{val}^{\sigma_k})$. Thus $\mathbb{O}^{\mathscr{G}}(\mathrm{val}^{\sigma_k}) \leq \mathrm{val}^{\sigma_{k+1}}$ for all k. Moreover, by Corollary 18, val^{σ_k} is a fixed point of $x \mapsto \textsc{RoundUp}(x, D^K)$ for all k. Therefore, $\mathbb{B}(\mathrm{val}^{\sigma_k}) \leq \mathrm{val}^{\sigma_{k+1}}$ for all k. By induction and the monotonicity of $\mathbb{B}$, if $\mu_k \leq \mathrm{val}^{\sigma_k}$, then $\mu_{k+1} = \mathbb{B}(\mu_k) \leq \mathbb{B}(\mathrm{val}^{\sigma_k}) \leq \mathrm{val}^{\sigma_{k+1}}$. □

By combining Theorems 80 and 83, we get an $n \cdot D^{2K}$ bound on the number of iterations of the strategy improvement algorithm using the greedy all-switches rule. The following theorem improves this bound for *all* switching rules.

Theorem 84. *The strategy improvement algorithm stops in at most $|V_{\mathrm{Max}}| \cdot D^K$ iterations.*

Proof. Let $\sigma_0 < \sigma_1 < \sigma_2 < \cdots$ be the sequence of positional strategies in an execution of the strategy improvement algorithm. We have $\mathrm{val}^{\sigma_k} \leq \mathrm{val}^{\sigma_{k+1}}$ for all k. As in the proof of Theorem 83, by Theorem 73 and the monotonicity of $\mathbb{O}^{\mathscr{G}[\sigma_{k+1}]}$ we get $\mathbb{O}^{\mathscr{G}[\sigma_{k+1}]}(\mathrm{val}^{\sigma_k}) \leq \mathrm{val}^{\sigma_{k+1}}$. Furthermore, Lemma 90 shows that $\mathrm{val}^{\sigma_k} \leq \mathbb{O}^{\mathscr{G}[\sigma_{k+1}]}(\mathrm{val}^{\sigma_k})$ and that this inequality is strict for every vertex of Max that has an outgoing edge in S. Let u be such a vertex. We claim that $\mathbb{O}^{\mathscr{G}[\sigma_{k+1}]}(\mathrm{val}^{\sigma_k})(u) - \mathrm{val}^{\sigma_k}(u) \geq D^{-K}$. Indeed, by Corollary 18, val^{σ} is a vector of rational numbers with common denominator at most D^K. Since $\mathbb{O}^{\mathscr{G}[\sigma_{k+1}]}(\mathrm{val}^{\sigma_k})(u) = \mathrm{val}^{\sigma_k}(v)$ for some vertex v, we get $\mathbb{O}^{\mathscr{G}[\sigma_{k+1}]}(\mathrm{val}^{\sigma_k})(u) - \mathrm{val}^{\sigma_k}(u) \geq D^{-K}$. In particular, $\mathrm{val}^{\sigma_{k+1}}(u) \geq \mathrm{val}^{\sigma_k}(u) + D^{-K}$. Therefore, the strategy improvement algorithm increases the value of at least one vertex of Max by at least D^{-K}. Since the value is bounded by 1, the algorithm must stop within $|V_{\mathrm{Max}}| \cdot D^K$ iterations. □

6.5 Algorithms based on permutations of random vertices

We present two algorithms, both based on the same key idea: to order the random vertices and to restrict ourselves to strategies aiming at reaching the best random vertices (with respect to the order on random vertices). For the remainder of this section we fix a stochastic normalised reachability game $\mathscr{G}$.

Permutation of random vertices

Let us write $V_{\text{Random}} = \{v_1, \ldots, v_k\}$. We represents total orders on V_{Random} using permutations $\pi : V_{\text{Random}} \to V_{\text{Random}}$. We write $\pi_i = \pi^{-1}(v_i)$ for the i-th element in the order defined by π. Intuitively, π represents a preference order for Max on random vertices.

A permutation π induces what we call the π-regions:

$$\begin{cases} W_\pi^{k+1} = \{\text{Win}\} \\ W_\pi^i = \text{Attr}_{\text{Max}}(\{\pi_i, \ldots, \pi_k, \text{Win}\}) \setminus \bigcup_{j>i} W_\pi^j \quad i \in [1,k] \\ W_\pi^0 = V \setminus \bigcup_{j>0} W_\pi^j \end{cases}$$

The idea is that once we fix the order on random vertices, the rest of the game is deterministic: Max and Min only aim at the best (with respect to the permutation π) random vertex they can get. More precisely, W_π^i is the set of vertices where Max can ensure to reach $\{\pi_i, \ldots, \pi_k, \text{Win}\}$ before any other random vertex (but no subset $\{\pi_j, \ldots, \pi_k, \text{Win}\}$).

Fact 19. *We have $W_\pi^0 = \{Lose\}$.*

Indeed, recall that Lose is the unique vertex with value 0. We write $W_\pi^{\geq i}$ for $\bigcup_{j \geq i} W_\pi^j$. The π-regions induce the π-strategies σ_π (for Max) and τ_π (for Min):

- on W_π^i, σ_π is a pure and positional attractor strategy to $\{\pi_i, \ldots, \pi_k, \text{Win}\}$;
- on W_π^i, τ_π is a pure and positional counter-attractor strategy ensuring never to reach $\{\pi_{i+1}, \ldots, \pi_k, \text{Win}\}$.

We can then define for every $u \in V$:

$$\text{val}_\pi(u) \quad = \quad \mathbb{P}^u_{\sigma_\pi, \tau_\pi}(\texttt{Reach}(\text{Win})).$$

It can be easily computed using systems of linear equations.

The key property of permutation-based algorithms is stated below:

Theorem 85 (Existence of an optimal permutation)**.** *There exists a permutation π such that σ_π is optimal for Max and τ_π is optimal for Min.*

The remainder of this section is devoted to a proof of Theorem 85. We later explore how this yields algorithms.

Live and self-consistent permutations

Definition 23 (Live and self-consistent permutations)**.** *Let π a permutation.*

- *We say that π is* self-consistent *if*

$$val_\pi(\pi_1) \leq val_\pi(\pi_2) \leq \ldots \leq val_\pi(\pi_k).$$

- *We say that π is* live *if for every $i \in [1,k]$:*

$$\delta(\pi_i)\left(W_\pi^{\geq i+1}\right) > 0.$$

Intuitively, if π is self-consistent the order given by π coincides with the preference of Max, and if π is live there is a positive probability to reach a preferable (for Max) π-region. The following result directly implies Theorem 85.

Lemma 91. *The following properties hold.*

- *If a permutation π is live and self-consistent, then the π-strategies are optimal.*
- *There exists a live and self-consistent permutation.*

Live and self-consistent permutations induce optimal strategies

We prove the first item of Lemma 91.

Lemma 92 (Correctness of live and self-consistent permutations)**.** *If π is self-consistent, then val_π is a fixed point of the operator $\mathbb{O}$. Consequently, $val^{\mathscr{G}} \leq val_\pi$.*

Proof. We prove the following properties:

1. For $i \in [1,k]$ and $u \in W_\pi^i$ we have $\mathrm{val}_\pi(u) = \mathrm{val}_\pi(\pi_i)$.
2. For $u \in V_{\mathrm{Max}}$ we have $\mathrm{val}_\pi(u) = \mathrm{val}_\pi(\sigma_\pi(u)) = \max\{\mathrm{val}_\pi(v) : u \to v \in E\}$.
3. For $u \in V_{\mathrm{Min}}$ we have $\mathrm{val}_\pi(u) = \mathrm{val}_\pi(\tau_\pi(u)) = \min\{\mathrm{val}_\pi(v) : u \to v \in E\}$.

For $v \in W_\pi^i$, up to the first visit to a random vertex, the strategy profile (σ_π, τ_π) generates a unique path. So we can speak of the first random vertex encountered from u when applying (σ_π, τ_π). By definition of σ_π (attractor to $\{\pi_i, \ldots, \pi_k, \mathrm{Win}\}$) and τ_π (counter-attractor strategy avoiding $\{\pi_{i+1}, \ldots, \pi_k, \mathrm{Win}\}$), this random vertex can only be π_i. According values follow, proving the first item.

Assume $u \in V_{\mathrm{Max}} \cap W_\pi^i$. By definition of σ_π (being an attractor strategy), $\sigma_\pi(u) \in W_\pi^i \cup \{\pi_i, \ldots, \pi_k, \mathrm{Win}\}$. Dually, since $u \notin W_\pi^{\geq i+1}$, we have $\sigma_\pi(u) \notin \{\pi_{i+1}, \ldots, \pi_k, \mathrm{Win}\}$. Hence, $\sigma_\pi(u) \in W_\pi^i \cup \{\pi_i\} = W_\pi^i$, and we get that $\mathrm{val}_\pi(u) = \mathrm{val}_\pi(\pi_i) = \mathrm{val}_\pi(\sigma_\pi(u))$. Assume towards contradiction that there exists $u \to v \in E$ such that $\mathrm{val}_\pi(v) > \mathrm{val}_\pi(u)$. Since we know that $\mathrm{val}_\pi(u) = \mathrm{val}_\pi(\pi_i)$, by self-consistence this implies that $v \in W_\pi^j$ with $j > i$ (with $\mathrm{val}_\pi(v) = \mathrm{val}_\pi(\pi_j)$). But then, we can deduce that $u \in \mathrm{Attr}_{\mathrm{Max}}(v) \subseteq \mathrm{Attr}_{\mathrm{Max}}(\{\pi_j, \ldots, \pi_k, \mathrm{Win}\})$, which is not the case, since $u \notin W_\pi^{\geq i+1} \supseteq W_\pi^j$. Contradiction.

The reasoning is symmetric for $u \in V_{\mathrm{Min}}$. □

The converse inequality $\mathrm{val}^{\mathscr{G}} \geq \mathrm{val}_\pi$ is not true for general or self-consistent permutations, but it does hold when adding the liveness property. A key technical property of the π-strategies σ_π when π is live is that it induces a stopping MDP: Min cannot prevent the game converging to Lose or Win.

Lemma 93 (Live permutations imply stopping MDPs)**.** *Let π be a live permutation. Then, for every strategy τ of Min, for every u, we have $\mathbb{P}^u_{\sigma_\pi,\tau}(\texttt{Reach}(\{Lose, Win\})) = 1$.*

Proof. Let $\alpha = \min_{i\in[1,k]} \delta(\pi_i)\big(W_\pi^{\geq i+1}\big)$. By definition of a live permutation we have $\alpha > 0$. We write V_i for the random variable representing the i-th state of a run. By definition of α, for $i \in [1,k]$ and $\ell \geq 0$, we have $\mathbb{P}^u_{\sigma_\pi,\tau}\Big(V_{\ell+1} \in W_\pi^{\geq i+1} \mid V_\ell = \pi_i\Big) \geq \alpha$ and $\mathbb{P}^u_{\sigma_\pi,\tau}\Big(\exists h < |W_\pi^i|,\ V_{\ell+h} \in \{\pi_i,\ldots,\pi_k,\mathrm{Win}\} \mid V_\ell \in W_\pi^i\Big) = 1$, since σ_π plays according to attractor strategies in the corresponding π-regions, where n is the number of vertices in the game. This implies that $\mathbb{P}^u_{\sigma_\pi,\tau}\Big(V_{\ell+n} = \mathrm{Win} \mid V_\ell \neq \mathrm{Lose}\Big) \geq \alpha^k$, which we can rewrite as: $\mathbb{P}^u_{\sigma_\pi,\tau}\Big(\forall \ell' \in [\ell,\ell+n],\ V_{\ell'} \neq \mathrm{Win} \mid V_\ell \neq \mathrm{Lose}\Big) \leq 1-\alpha^k$. Iterating, we get that for every i,

$$\mathbb{P}^u_{\sigma_\pi,\tau}\Big(\forall \ell \in [0, i\cdot n],\ V_\ell \notin \{\mathrm{Win},\mathrm{Lose}\} \mid V_0 \neq \mathrm{Lose}\Big) \leq (1-\alpha^k)^i.$$

Hence by taking the limit $\mathbb{P}^u_{\sigma_\pi,\tau}(\forall \ell \geq 0,\ V_\ell \notin \{\mathrm{Win},\mathrm{Lose}\}) = 0$, which equivalently reads $\mathbb{P}^u_{\sigma_\pi,\tau}(\exists \ell \geq 0,\ V_\ell \in \{\mathrm{Win},\mathrm{Lose}\}) = 1$. □

We can now show that if π is a live and self-consistent permutation, then $\mathrm{val}_\pi = \mathrm{val}^{\mathscr{G}}$. Indeed, by Lemma 92 val_π is a fixed point of the operator $\mathbb{O}^{\mathscr{G}}$, hence of $\mathbb{O}^{\mathscr{G}[\sigma_\pi]}$. By Lemma 93, the MDP obtained when restricting to σ_π is stopping, hence (the special case for MDPs of) Theorem 77 implies that $\mathbb{O}^{\mathscr{G}[\sigma_\pi]}$ has a unique fixed point, which is $\mathrm{val}^{\mathscr{G}}$. Hence $\mathrm{val}_\pi = \mathrm{val}^{\mathscr{G}}$.

Existence of a live and self-consistent permutation

We now prove the second item of Lemma 91.

Lemma 94 (Live and value non-decreasing imply self-consistent)**.** *Let π be a live permutation such that $val^{\mathscr{G}}(\pi_1) \leq val^{\mathscr{G}}(\pi_2) \leq \cdots \leq val^{\mathscr{G}}(\pi_k)$. Then π is self-consistent.*

Proof. We show that for every vertex u we have $\mathrm{val}^{\mathscr{G}}(u) = \mathrm{val}_\pi(u)$, which implies the result. We first note that for every $i \in [1,k]$, for every $u \in W_\pi^i$ we have $\mathrm{val}^{\mathscr{G}}(u) = \mathrm{val}^{\mathscr{G}}(\pi_i)$.

Let us define a σ^* from u: it plays the attractor strategy to $\{\pi_i,\ldots,\pi_k,\mathrm{Win}\}$) until Win or a random vertex is reached, and in the latter case, switch to an optimal strategy. Clearly for every strategy τ for Min we have $\mathbb{P}^u_{\sigma^*,\tau}(\texttt{Reach}(\{\mathrm{Win}\})) \geq \min_{i\in[i,k]} \mathrm{val}^{\mathscr{G}}(\pi_j) = \mathrm{val}^{\mathscr{G}}(\pi_i)$. Hence $\mathrm{val}^{\mathscr{G}}(u) \geq \mathrm{val}^{\mathscr{G}}(\pi_i)$.

Conversely, let us define a strategy τ^* from u: it plays the counter-attractor strategy to $\{\pi_{i+1},\ldots,\pi_k,\mathrm{Win}\}$ until Lose or a random vertex is reached, and in the latter case, switch to an optimal strategy. It may never hit Lose or a random vertex, but this is

good to Min. Clearly for every strategy σ for Max we have $\mathbb{P}^u_{\sigma,\tau^*}(\texttt{Reach}(\{\text{Win}\})) \leq \max_{j\in[1,i]} \text{val}^{\mathscr{G}}(\pi_j) = \text{val}^{\mathscr{G}}(\pi_i)$. Hence $\text{val}^{\mathscr{G}}(u) \leq \text{val}^{\mathscr{G}}(\pi_i)$.

Now we can conclude: both $\text{val}^{\mathscr{G}}$ and val_π are fixed points of the operator $\mathbb{O}^{\mathscr{G}[\sigma_\pi]}$, which is unique because π is live (Lemma 93 and Theorem 77). □

It remains to show that there always exists a live permutation satisfying the hypothesis of Lemma 94. To do so, we show the following structural property of the game, which will help building an appropriate live permutation.

Lemma 95. *Let $X \subseteq V$ such that Win $\in X$, and $Y = V \setminus Attr_{\text{Max}}(X)$. Then either $Y = \{Lose\}$, or there exists a random vertex $u \in Y$ such that $val^{\mathscr{G}}(u) = \max\{val^{\mathscr{G}}(v) : v \in Y\}$ and $\delta(u)\Big(Attr_{\text{Max}}(X)\Big) > 0$.*

Proof. Let $m = \max\{\text{val}^{\mathscr{G}}(u) : u \in Y\}$, we write $Z = \big\{u \in Y : \text{val}^{\mathscr{G}}(u) = m\big\}$. We show that if there are no random vertices $u \in Z$ such that $\delta(u)\Big(\text{Attr}_{\text{Max}}(X)\Big) > 0$, then $Z = \{\text{Lose}\}$. To do so, we show that if $u \in Z$, then $\text{val}^{\mathscr{G}}(u) = 0$. Let us assume towards a contradiction that $\text{val}^{\mathscr{G}}(u) > 0$.

Let τ be a counter-attractor (pure and positional) strategy for Min to stay safe of $\text{Attr}_{\text{Max}}(X)$ from Y. The following properties are direct consequences of the definitions: for any $u \in Z$,

- if $u \in V_{\text{Min}}$ and $\tau(u) = u \to v \in E$, then $v \in Z$,
- if $u \in V_{\text{Max}}$ and $u \to v \in E$, then $v \in Y$,
- if $u \in V_{\text{Random}}$ and $\delta(u)(v) > 0$, then $v \in Z$.

We now define a strategy τ' which plays from v as τ until Z is left, and then switches to an optimal strategy for Min. Let σ a strategy for Max. Thanks to the properties above, a play consistent with σ and τ' from v either stays forever in Z, or reaches a vertex v such that $\text{val}^{\mathscr{G}}(v) < \text{val}^{\mathscr{G}}(u) = m$. Let us write $\beta = \max\{\text{val}^{\mathscr{G}}(v) \mid v \in Y \setminus Z\} < m$. We then have that $\mathbb{P}^v_{\sigma,\tau'}(\texttt{Reach}(\{\text{Win}\})) \leq \beta$, because the probability to reach Win if staying forever in Z is 0. Hence $\text{val}^{\mathscr{G}}(v) \leq \beta < \text{val}^{\mathscr{G}}(v)$, a contradiction. □

We can now prove the existence of a live permutation such that $\text{val}^{\mathscr{G}}(\pi_1) \leq \text{val}^{\mathscr{G}}(\pi_2) \leq \cdots \leq \text{val}^{\mathscr{G}}(\pi_k)$. We define the permutation π inductively, by repeatedly using Lemma 95. For every $i \in [k,1]$ we define π_i by applying Lemma 95 to $X = \{\pi_{i+1}, \ldots, \pi_k, \text{Win}\}$. By construction,

- $\text{val}^{\mathscr{G}}(\pi_i) = \max\{\text{val}^{\mathscr{G}}(v) \mid v \in V \setminus \text{Attr}_{\text{Max}}(\{\pi_{i+1}, \ldots, \pi_k, \text{Win}\})\}$;
- $\delta(\pi_i)\Big(\text{Attr}_{\text{Max}}(\{\pi_{i+1}, \ldots, \pi_k, \text{Win}\})\Big) > 0$.

It follows that π is live, and the hypothesis of Lemma 94 is satisfied. Hence π is self-consistent. This concludes the proof of the second item of Lemma 91.

Strategy enumeration algorithm

Theorem 85 induces a simple algorithm for computing the values and optimal strategies for both players in a stochastic reachability game: enumerate all permutations of random vertices, and for each of them, check whether it is live and self-consistent; stop when one is found. Note that indeed given π, it is easy to compute the π-regions and strategies and check for liveness and self-consistency, in polynomial time. However, as such, this requires to enumerate all permutations of random vertices, and there are $|V_{\text{Random}}|!$ of them. Hence the overall complexity of finding the values and the optimal strategies is exponential.

Strategy improvement algorithm

We will describe a strategy improvement algorithm, which may avoid enumerating all permutations. Note that there is nevertheless no guarantee that the overall complexity will be better than the strategy enumeration algorithm.

The algorithm consists in the following steps:

- Initialization step: Compute a live permutation π
- Improvement step: Given a live permutation π, compute a live and self-consistent permutation in $\mathscr{G}[\sigma_\pi]$.

We argue (not in full details) that the following properties are satisfied:

1. We can compute an initial live permutation in polynomial time.
2. For every live permutation π, we can compute in polynomial time a live and self-consistent permutation π' in $\mathscr{G}[\sigma_\pi]$.
3. The above mentioned permutation π' is live in $\mathscr{G}$ as well.
4. The improvement step is an improvement:
   - $\text{val}^{\mathscr{G}[\sigma_\pi]} \leq \text{val}^{\mathscr{G}[\sigma_{\pi'}]}$, and
   - If $\text{val}^{\mathscr{G}[\sigma_\pi]} = \text{val}^{\mathscr{G}[\sigma_{\pi'}]}$, then π' is self-consistent in $\mathscr{G}$.

The first property is based on the inductive construction following Lemma 95.

For the second property, we know as a consequence of Theorem 85 that there exists a live and self-consistent permutation π' in $\mathscr{G}[\sigma_\pi]$. Note that for this we need to argue that $\mathscr{G}[\sigma_\pi]$ is normalized.

For the third property, we note that the π'-regions in $\mathscr{G}[\sigma_\pi]$ are included in the π'-regions in $\mathscr{G}$, which implies the result.

The last property is harder to argue; it expresses the fact that the new permutation π' improves over π.

This last property ensures both termination of the algorithm: indeed, it is ensured by the finiteness of the number of permutations, and by the improvement characterization above. Note that it may be the case that in the worst-case, all permutations will be enumerated. No lower nor upper bound is known.

6.6 Reductions to simple stochastic games

The rest of this chapter focused on stochastic reachability games. As we will see in this section, they are powerful enough to encode other classes of games, such as non-stochastic discounted payoff games, and stochastic parity, mean payoff, and discounted payoff games.

6.6.1 From discounted payoff games to stochastic reachability games

Theorem 86 (Reducing discounted payoff games to stochastic reachability games)**.** *Solving (non-stochastic) discounted payoff games reduces in polynomial time to solving stochastic reachability games.*

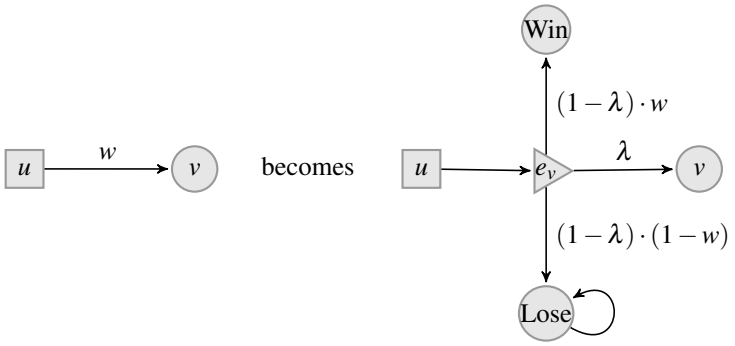


Figure 6.6: From discounted payoff games to stochastic reachability games.

Proof. Let $\mathscr{G}$ a discounted payoff game. Using a linear transformation we can assume that all weights are rational numbers in $[0,1]$. We construct $\mathscr{G}'$ a stochastic reachability game by adding to vertices v_{Win} and v_{Lose}, and each edge $e = u \xrightarrow{w} v$ in $\mathscr{G}$ is redirected to a new random vertex v_e with probabilistic distribution

$$\delta(v_e) = \lambda \cdot v + (1-\lambda)\cdot w \cdot \text{Win} + (1-\lambda)\cdot(1-w)\cdot \text{Lose}.$$

The reachability condition is $\texttt{Reach}(\text{Win})$. (Note that the game $\mathscr{G}'$ is stopping.)

We claim that for every $u \in V$, we have

$$\text{val}^{\mathscr{G}}(u) = \text{val}^{\mathscr{G}'}(u).$$

To this end, we rely on Lemma 38, which states that the discounted payoff values are the unique fixed point of the operator $\mathbb{O}^{\mathscr{G}} : F_V \to F_V$ defined by

$$\mathbb{O}^{\mathscr{G}}(\mu)(u) = \begin{cases} \max\left\{\lambda\cdot\mu(v) + (1-\lambda)\cdot w : u \xrightarrow{w} v \in E\right\} & \text{if } u \in V_{\text{Max}}, \\ \min\left\{\lambda\cdot\mu(v) + (1-\lambda)\cdot w : u \xrightarrow{w} v \in E\right\} & \text{if } u \in V_{\text{Min}}. \end{cases}$$

We show that $(\mathrm{val}^{\mathscr{G}'}(u))_{u\in V}$ is a fixed point of the operator $\mathbb{O}^{\mathscr{G}}$.

Thanks to Theorem 73, the values in $\mathscr{G}'$ satisfy the following:

$$\begin{cases} \mathrm{val}^{\mathscr{G}'}(\mathrm{Win}) &= 1, \\ \mathrm{val}^{\mathscr{G}'}(\mathrm{Lose}) &= 0, \\ \mathrm{val}^{\mathscr{G}'}(u) &= \max\left\{\mathrm{val}^{\mathscr{G}'}(v) : u \to v \in E\right\} & \text{if } u \in V_{\mathrm{Max}}, \\ \mathrm{val}^{\mathscr{G}'}(u) &= \min\left\{\mathrm{val}^{\mathscr{G}'}(v) : u \to v \in E\right\} & \text{if } u \in V_{\mathrm{Min}}, \\ \mathrm{val}^{\mathscr{G}'}(v_e) &= \lambda \cdot \mathrm{val}^{\mathscr{G}'}(v) \\ &+ (1-\lambda)\cdot w \cdot \mathrm{val}^{\mathscr{G}'}(\mathrm{Win}) \\ &+ (1-\lambda)\cdot(1-w)\cdot \mathrm{val}^{\mathscr{G}'}(\mathrm{Lose}) \end{cases}$$

Eliminating the values $\mathrm{val}^{\mathscr{G}'}(v_e)$ from these equations, we obtain

$$\begin{cases} \mathrm{val}^{\mathscr{G}'}(u) &= \max\left\{\lambda \cdot \mathrm{val}^{\mathscr{G}'}(v) + (1-\lambda)\cdot w : u \to v \in E\right\} & \text{if } u \in V_{\mathrm{Max}}, \\ \mathrm{val}^{\mathscr{G}'}(u) &= \min\left\{\lambda \cdot \mathrm{val}^{\mathscr{G}'}(v) + (1-\lambda)\cdot w : u \to v \in E\right\} & \text{if } u \in V_{\mathrm{Min}}. \end{cases}$$

Hence $(\mathrm{val}^{\mathscr{G}'}(u))_{u\in V}$ is a fixed point of the operator $\mathbb{O}^{\mathscr{G}}$. Since $\mathbb{O}^{\mathscr{G}}$ has a unique fixed point $\mathrm{val}^{\mathscr{G}}$, this implies the desired equality. □

We note that Theorem 86 easily extends to discounted stochastic games: from a discounted stochastic game, one can build a stochastic reachability game with the same values.

6.6.2 From stochastic mean payoff games to stochastic discounted payoff games

Both positional determinacy for mean payoff games and the reduction to discounted payoff games (presented in Theorem 47) can be lifted to stochastic games.

Theorem 87 (Stochastic mean payoff games are positionally determined)**.** *Stochastic mean payoff games are uniformly purely and positionally determined.*

Theorem 88 (Reducing stochastic mean payoff games to stochastic discounted payoff games)**.** *Let $\mathscr{G}$ a stochastic mean payoff game. Let $\lambda \in (0,1)$, we define $\mathscr{G}_\lambda$ the stochastic discounted payoff game obtained from $\mathscr{G}$. There exists a discount factor λ computable in polynomial time such that any pair of optimal pure positional strategies in the discounted payoff game with discount factor λ is also a pair of optimal strategies in the mean payoff game.*

6.6.3 From stochastic parity to stochastic mean payoff

Both positional determinacy for parity games and the reduction to mean payoff games (presented in Theorem 40) can be lifted to stochastic games.

Theorem 89 (Stochastic parity games are positionally determined)**.** *Stochastic parity games are uniformly purely and positionally determined.*

Theorem 90 (Reducing stochastic parity games to stochastic mean payoff games)**.** *Solving stochastic parity games reduces in polynomial time to solving stochastic mean payoff games.*

The reduction is similar as the one presented in Theorem 40, but requires additional technical care. In particular, it relies on the fact that we can compute the almost-surely and positively winning regions.

Theorem 91. *The following holds.*

- *There exists a polynomial time algorithm for computing the positively winning region of stochastic parity games.*
- *There exists a polynomial time algorithm for computing the almost-surely winning region of stochastic parity games.*

Let $\mathscr{G}$ a stochastic parity game. We let $p_{\min}$ the minimal non-zero probability that appears in $\mathscr{G}$, and n the number of vertices.

We construct $\mathscr{G}'$ a stochastic mean payoff game. As a first step, we compute the almost-sure and positive winning regions, we replace the almost-sure winning region with a sink vertex Win with a self-loop with weight 1, and the complement of the positive winning region with a sink vertex Lose with a self-loop with weight -1. For each edge $u \xrightarrow{p} v$ in $\mathscr{G}$, we have an edge $u \xrightarrow{\left(-2n \cdot p_{\min}^{-n}\right)^k} v$ in $\mathscr{G}'$.

We claim without proof that for all $u \in V$, we have

$$\mathrm{val}^{\mathscr{G}}(u) = \frac{1}{2} \cdot \left(\mathrm{val}^{\mathscr{G}'}(u) + 1\right).$$

Bibliographic references

The study of stochastic games started long before the algorithmic perspectives we develop here. This chapter took inspiration from the reference survey chapter by Kučera [Kuč11]. The introduction of stochastic games is due to Condon [Con92], where she proves (a weak form of) positional determinacy, and constructs strategy improvement algorithms. Moving back in time, positional determinacy for stochastic discounted payoff games was already proved (as a special case) by Shapley [Sha53], and for stochastic mean payoff games by Liggett and Lippman [LL69]. The Borel determinacy result for stochastic games was proved by Maitra and Sudderth [MS98]. The last section about reductions to stochastic reachability games is inspired by [AM09], which clarifies a lot the relationships between the different classes.

As for parity games, mean payoff games, and discounted payoff games, the major open question is whether stochastic reachability games can be solved in polynomial time. Strongly polynomial bound for strategy improvement have been obtained when the discount factor is fixed [HMZ13]. The best complexity to date is sub-exponential time randomised algorithms. A very general point of view on these algorithms is given by reducing them to LP-type problems [Hal07].

A more general point of view on strategy improvement algorithms was developed by Auger, de Montjoye, and Strozecki [AdMS21]. Constructing strategy improvement algorithms for more general classes of stochastic games is non-trivial, see for instance the case of stochastic mean payoff games [ACTDG13].

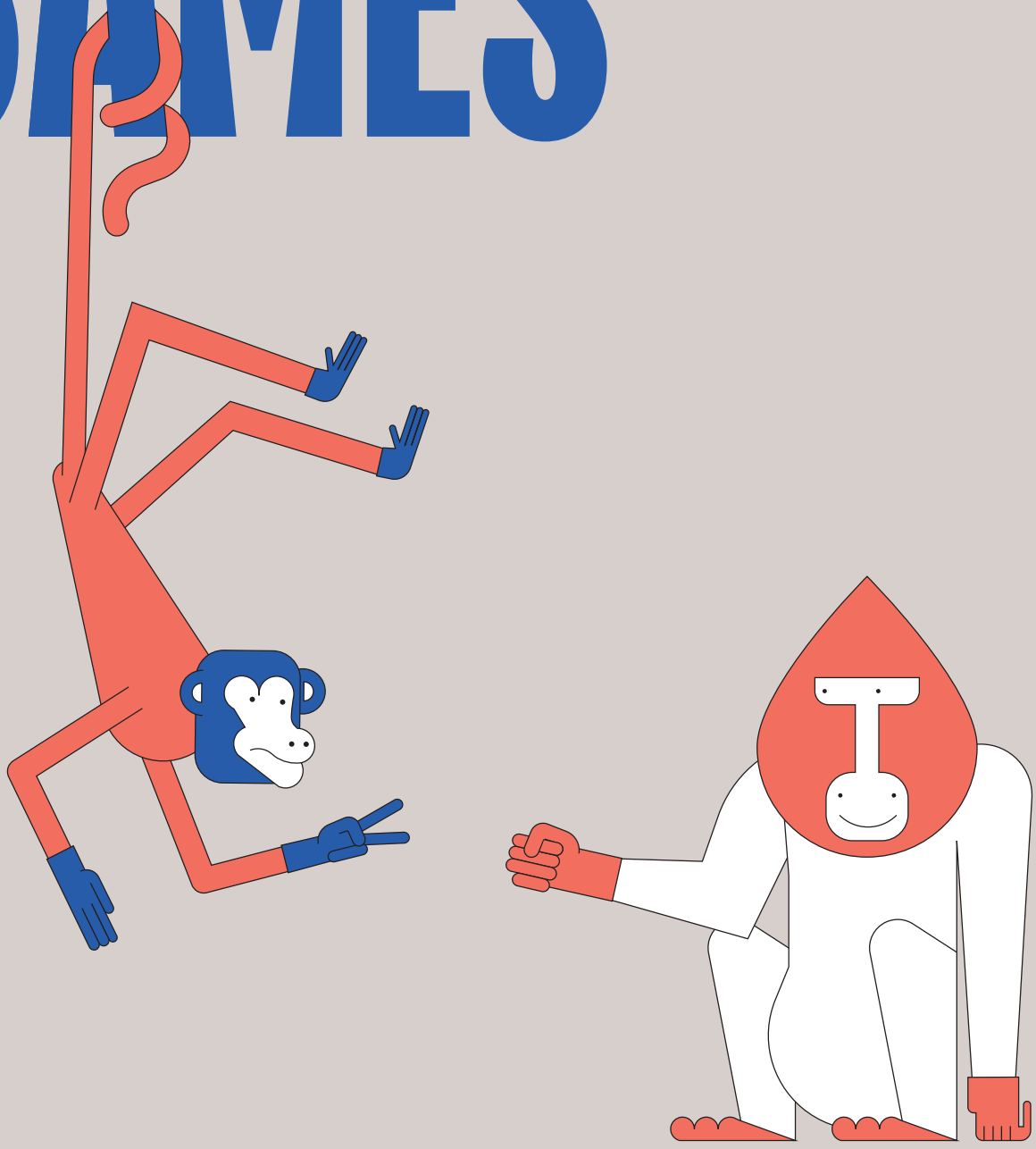
The permutation-based algorithm is due to Gimbert and Horn [GH08]. There is a wealth of algorithms we did not cover in this chapter, let us mention the accelerated value iteration for stochastic reachability games [IJM12], the value iteration algorithm for ergodic mean payoff games [AGKS22], and the pumping algorithm for stochastic mean payoff games [BEGM19].

Stochastic games have also been related to other models of computations, for instance constraint satisfaction problems [BM16], and complexity classes related to fixed point computations [EY10, FGMS20].

Part III

Information

CONCURRENT GAMES



Chapter 7

Concurrent Games

RASMUS IBSEN-JENSEN

This chapter considers concurrent games. The concurrent games we consider are extensions of the games considered in Chapter 2 and Chapter 4, but where the choice of which edge to choose in a round is determined not by the choice of the owner of the vertex (indeed the vertices in concurrent games have no owners), but by the outcome of a matrix game corresponding to the vertex and played in that round. A matrix game is in turn a generalization of rock-paper-scissors, where each player picks an action simultaneously and then their pair of actions determines the outcome.

We will consider concurrent discounted, reachability and mean payoff games and the definitions of the different objectives is as in the introduction. The chapter is divided into four sections:

1. Matrix games
2. Concurrent discounted payoff games
3. Concurrent reachability games
4. Concurrent mean payoff games

As we go through the sections in this chapter, the complexity of the strategies and the computational complexity of solving the games rises: indeed, since the games are generalizations of rock-paper-scissors, the strategies used requires randomness, but towards the end, no optimal or finite-memory ε-optimal strategies exists in general and even the principle of sunken cost does not apply! Even with all this, the related questions about values are solvable in polynomial space and thus also in exponential time even in the last section. The results we will focus on characterizes the complexity of the both the strategies as well as the computational complexity. In each section we first give some positive result and some number of negative results. Each negative result also applies to the classes of games considered in the latter sections and each positive

result applies to the classes considered in earlier sections (however, the positive results of latter sections will have worse complexity than the positive results from earlier sections). As mentioned, the strategies for this chapter require randomness and not too surprisingly, this implies that there is little difference between having stochastic or deterministic transition functions.

7.1 Notations

The definition of arena $\mathscr{A}$ in this chapter is $\mathscr{A} = (G, \Delta)$, where $G = (V, E)$ is a graph and $\Delta : V \times A \times A \to \mathscr{D}(E)$. In particular, we are not using the sets V_{Adam} and V_{Eve}. The games are played similarly to before and formally as follows: There is a token, initially on the initial vertex. Whenever the token is on some vertex v, Eve selects an action r in A and Adam selects an action c in A. The edge $e = (v, c, w)$ is then drawn from the distribution $\Delta(v, r, c)$ and the token is pushed from v to w. In general, the game continues like that forever.

We will use the following simplifying assumptions in this chapter:

1. We will assume that all colors are in $\{0, 1\}$, except for the section on Matrix games where we additionally also allow -1 (to be able to easily illustrate the game rock-paper-scissors). This simplifies some expressions, but generally, the dependency on the number of colors is not too bad comparatively.

2. To make illustrations easier, we assume that for any pair of edges e, e' in $\Delta(v, a, a')$ for any v, a, a', we have that $c(e) = c(e')$, i.e. the color does not depend on which edge is picked from $\Delta(v, a, a')$, but only v, a, a'. This assumption does not matter for the types of games considered.

We will overload the notation slightly for notational convenience, in that for any v, a, a', we will write $c(v, a, a')$ for $c(e)$ where $e \in \Delta(v, a, a')$ (note that the second assumption ensures that this is well-defined, i.e. there is only one such color).

A vertex v is absorbing if and only if each player has only 1 action in v and $\Delta(v, 1, 1) = v$.

To describe the complexity of good stationary strategies in concurrent games, we will use the notion of patience. Given a probability distribution $d \in \mathscr{D}$ the distribution has patience p if $p = \min_{i \in \text{supp}(d)} d(i)$ (i.e. the patience is the smallest, non-zero probability that an event may happen according to d). In essence, if you have low enough patience you can typically guess the strategy and check whether it is a good strategy (when you fix a strategy, the game becomes a Markov decision process, which are relative easy to work with), the game can solved in $\mathsf{NP} \cup \mathsf{coNP}$. However, some times the patience is huge and writing down a good strategy, in binary, cannot be done in polynomial space (it is quite surprising in some sense that even with this property, finding the values in the games remain in PSPACE).

We will illustrate a stochastic arena $\mathscr{A} = (G, \Delta)$ as follows: For each non-absorbing vertex v, there is matrix. Entry (i, j) of the matrix illustrating $v \in V$ describes what happens if, when the token is on v, Eve plays i and Adam j. The entry contains a color c, which is $c(v, i, j)$, and there is an arrow from entry (i, j) of v to w if there is

0	-1	1
1	0	-1
-1	1	0

Figure 7.1: Rock-paper-scissors. The color is 1 if Eve wins, 0 if they draw and -1 if Adam wins. Also, the actions are ordered as in the name of the game

an edge $e = (v, c, w)$ in $\Delta(v, i, j)$. The arrow corresponding to e is denoted with the probability $\Delta(v, i, j)(e)$. Especially, to simplify the illustrations we will do as follows: If $|\text{supp}(\Delta(v, i, j))| = 1$, we do not include the probability (which is 1). Also, in that case, let $e = (v, c, w) = \Delta(v, i, j)$ and if $v = w$, we omit the arrow and if w is absorbing we write c^* in entry i, j of v, where c is the color $c(w, 1, 1)$ (in this case, we omit the number $c(e)$ from the illustration, but in none of our illustrations does this number matter for what we try to illustrate).

7.2 Matrix games

A matrix game is a game defined from a $(R \times C)$-matrix M of numbers for some R, C. The game is played as follows: Eve picks a row r and Adam picks a column c simulations like in rock-paper-scissors. Adam then pays Eve $M[r, c]$, i.e. the content of the entry defined by being in row r and column c. A strategy in such a game for Eve (resp. Adam) consists of a distribution over the rows (resp. columns). There is an illustration of rock-paper-scissors as a matrix game in Figure 7.1.

The following theorem lists some known results for matrix games:

Theorem 92. *Each $(m \times n)$-matrix game M is determined and there exists optimal strategies for each player.*

- *The value and an optimal strategy for each player can be found in polynomial time and the problem is equivalent to linear programming.*
- *Let $c > 0$ be some constant. Consider the matrix cM where each entry of M has been multiplied by c. Then, the value of cM is cv.*
- *Let c be some constant. Consider the matrix $M + c$ where each entry of M is c larger (additively). Then, the value of $M + c$ is $v + c$.*
- *The value of matrix games are monotone in the entries.*

We will omit the proof of the existence of values, optimal strategies and the first bullet. The second bullet can be viewed as changing currency and clearly, this does not affect the optimal strategy. The third bullet can be viewed as getting a reward before playing the game, and again, clearly this does not affect how to play it. The last bullet

can be seen from that each pair of strategies must give a higher reward if the entries of the matrix is higher. This is especially true if you consider the optimal strategy for Eve in M together with an arbitrary strategy for Adam, which then shows that the value is higher.

Given a matrix M, we will by $\text{val}[M]$ denote the value of the matrix game M.

Perhaps interestingly, an illustration of a matrix M can be viewed as a game arena $\mathscr{A}$ (for concurrent games) with only one non-absorbing vertex. In each type of games considered in this section (apart from concurrent reachability games, where no game can be illustrated as a matrix with non-star entries different from 0), the value of the game with that arena matches $\text{val}[M]$ and the optimal strategies for each player is to play an optimal strategy from M in each round. One can also consider a game arena $\mathscr{A}^*$ with an illustration similar to M, but where there is a star in each entry (and $c(v,i,j) = 0$ for the unique non-absorbing state v and any pair of actions i, j). Again, the value is $\text{val}[M]$ (except for the case of discounted objectives, where the value is $(1-\delta)\text{val}[M]$) and the optimal strategies for each player is again to play an optimal strategy from M.

One could easily be lead to believe that in games (called repeated games with absorbing states) that can be illustrated as a single matrix M with some entries stared and others not, the value would be similar to $\text{val}[M]$ and the optimal strategy would again be to play the optimal strategy from M. However, this is very much not true and indeed, many of the games in this chapter, illustrating how complex concurrent games can be, are repeated games with absorbing states! In particular repeated games with absorbing states may (1) have irrational values and probabilities in optimal strategies (with any objective), (2) have no optimal strategies (for reachability and mean payoff objectives) and (3) have no ε-optimal finite-memory or ε-optimal Markov strategies (for mean payoff objectives)!

7.3 Concurrent discounted payoff games

In this section we focus on concurrent discounted payoff games. The key property of these games is that to a high degree, only the relative early part of the play matters. We will first argue that the value iteration algorithm works and especially converges to the value of the game and then that there are stationary optimal strategies in concurrent discounted payoff games. While the value iteration algorithm also works for the games considered in the latter sections, we will not explicitly show it there, since the proofs become much more complex. The argument here however will allow us to show quite a few more statements in essence as corollaries of the theorem that value iteration works.

The value iteration algorithm is based on the concept of finite-horizon (or time limited) games. It is also sometimes referred to as dynamic programming. Specifically, apart from the usual definition of a game, there is an additional integer T, denoting how many rounds are remaining initially, and a vector v, assigning a reward to each node if the game ends in that node with 0 rounds remaining. After round T the reward is 0. I.e. for $T = 0$, the outcome reward from node x is v_x in the first round and 0 in each later round. Let $\text{valOp}^T(v)$ be the vector that assigns to each node its value in the game with time-limit T with vector v.

In general this formulation leads to a simple dynamic algorithm that computes

$\text{valOp}^T(v)$ inductively in T. We have that $\text{valOp}^0(v) = v$ and given $\text{valOp}^{T-1}(v)$ it is easy to compute $\text{valOp}^T(v)$ because, if Eve selects row i and Adam column j in node x in the first round, the outcome is

$$\sum_{v \in V} \text{valOp}^{T-1}(v)\Delta(x,i,j)(v)$$

and thus $(\text{valOp}^T(v))_x$ is the value of the matrix $M^{T,x,v}$, where entry i,j is

$$\sum_{v \in V} \text{valOp}^{T-1}(v)\Delta(x,i,j)(v)$$

It is common to start with the all-0 vector for v when using the value iteration algorithm.

The following lemma shows many interesting properties of concurrent discounted payoff games.

Lemma 96 (Concurrent discounted payoff games)**.** *Concurrent discounted payoff games have the following properties:*

- *The value iteration algorithm converges for any initial vector v.*
- *The value iteration algorithm has an unique fixed point, independent of the initial vector v.*
- *There are optimal stationary strategies in concurrent discounted payoff games and the unique fixed point of the value iteration algorithm is the value (thus, the games are determined).*
- *The value of a concurrent discounted game can approximated in PPAD.*
- *There are ε-optimal stationary strategies with patience below $\frac{m\log(\varepsilon/2)}{\log(1-\gamma)\varepsilon}$.*

Proof. The first item comes from considering the vectors v and $\text{valOp}^1(v)$. We thus have that $\text{valOp}^{T+1}(v) \in [\text{valOp}^T(v) - (1-\gamma)^T, \text{valOp}^T(v) + (1-\gamma)^T]$ for all T. The statement then comes from that $\sum_{i=1}^{\infty}(1-\gamma)^i$ is a converging sum.

The second item comes from considering two fixed points, u, v. I.e., $\text{valOp}^1(v) = v$ and thus $\text{valOp}^T(v) = v$ for all T. Similar for u. But, $v = \text{valOp}^T(v) \in [\text{valOp}^T(u) - (1-\gamma)^T, \text{valOp}^T(u) + (1-\gamma)^T] = [u-(1-\gamma)^T, u+(1-\gamma)^T]$. Since it is true for all T, we have that $u = v$.

Claim 1. *Consider some T and the strategy for Eve that plays the first T steps following an optimal strategy in the finite-horizon game of length T with vector v, followed by playing arbitrarily. Then, the outcome is above $valOp^T(v) - (1-\gamma)^T \max_i v_i$.*

Proof. For any strategy for Adam, the expected reward for the first T rounds is at least the expected reward in the finite-horizon game. In each remaining round, the reward is at least 0 in the real game, but v_i in round T for some i followed by 0's in the finite-horizon game. Since the outcome is $\text{valOp}^T(v)$ in the finite-horizon game, the real outcome is then as described. □

One can show a similar statement for Adam. For any $\varepsilon > 0$ one can pick a big enough T such that $(1-\gamma)^T \max_i v_i \leq \varepsilon$.

Let v^* be the unique fixed point of the value iteration algorithm. Thus, $v^* = \text{valOp}^T(v^*)$ for all T. Pick some optimal strategies σ_x, τ_x in M^{T,x,v^*} for each x. Let σ^*, τ^* be the strategies that play σ_x, τ_x whenever in node x in each round. The strategy σ, τ are optimal in $\text{valOp}^T(v^*)$ for each T, because v^* is a fixed point. But, for each $\varepsilon > 0$, the strategy σ ensures outcome at least $v - \varepsilon$ and τ ensures outcome at most $v + \varepsilon$ using the claim. Hence, the third item follows.

The fourth item follows from that the value iteration algorithm is a contraction.

For the fifth item, consider the strategy used in the claim. Let T be $\log(\varepsilon/2)/\log(1-\gamma)$, i.e. T is such that

$$\gamma \sum_{i=T}^{\infty} (1-\gamma)^i = \varepsilon/2$$

or in words, T is such that the total outcome of each step after the T-th step is at most $\varepsilon/2$. Intuitively, if we modify the strategy very little, then the change is unlikely to come up in the first T steps. More precisely, we will modify our strategy so that the probability that change will matter is less than $\varepsilon/2$. That implies that the outcome differs by at most ε from the value. We will use this intuition together with the argument for the third item to give a bound on the patience of ε-optimal strategies. Fix some optimal stationary strategy σ for Eve and an arbitrary stationary strategy τ for Adam. Let σ' be a stationary strategy obtained from σ rounded greedily so that each probability is a rational number with denominator

$$q = mT/\varepsilon = \frac{m \log(\varepsilon/2)}{\log(1-\gamma)\varepsilon}.$$

We will argue that σ' is ε-optimal.

The rounding proceeds inductively as follows for each node x: The numbers p_i are the original probability and the numbers p'_i are the new probabilities. For each i, the number p'_i is defined as follows: If $\sum_{j=1}^{i-1}(p_i - p'_i) > 0$, then round up (i.e. p'_i is the smallest rational with denominator q so that $p_i < p'_i$) and otherwise round down, except the last number p'_ℓ, which is such that $\sum_{j=1}^{\ell} p'_i = 1$. Note that this ensures that $-1/q < \sum_{j=1}^{i-1}(p_i - p'_i) < 1/q$. It also ensures that $|p_i - p'_i| < 1/q$ for all i (including for $i = \ell$).

For all nodes x and rounds $T' \leq T$ we will define some random variables. Specifically, the random variables denote what happen in round T' if in node x. The random variable $X_{x,T'}$ (resp. $Y_{x,T'}$) denotes the action picked by Eve if Eve follows σ (resp. σ'). The random variable $Z_{x,T'}$ denotes the action picked by Adam. For each action pair (i,j) the random variable $W_{x,i,j,T'}$ denotes the node entered in round $T'+1$, if Eve picks i and Adam j. (As a side note: Each of the random variables are distributed the same way independent of T'). Each of these random variables are independent of each other, except that (as we will define later) the random variables $X_{x,T'}$ and $Y_{x,T'}$ for each x, T' are very much not independent of each other.

We see that we can view the first T steps of the play when Eve follows σ by only considering the outcome of $X_{x,T'}$, $Z_{x,T'}$ and $W_{x,i,j,T'}$ for all T' and x (even stronger: We only need to consider one x, i, j for each T', because the token is on only one node

at a time). Similarly, for σ', but using $Y_{x,T'}$ instead of $X_{x,T'}$. For this to work, note that each random variable should be independent, except that the random variables $X_{x,T'}$ and $Y_{x',T''}$ need not be independent of each other for any x', T''. This is precisely the property we had for them! For each x, T', we will then use a coupling $C_{x,T'} = (X'_{x,T'}, Y'_{x,T'})$, a distribution over $[m]^2$, such that $X'_{x,T}$ is distributed as $X_{x,T'}$ and $Y'_{x,T'}$ is distributed as $Y_{x,T'}$. We will use a classic result for distributions, called the Coupling Lemma.

To introduce the Coupling Lemma, first, we need the notion of total variation distance. Given two distributions, Δ and Δ' over a set S, the total variation distance t between Δ and Δ' is

$$t(\Delta, \Delta') = \frac{1}{2} \sum_{x \in S} |\Delta(x) - \Delta'(x)|$$

Lemma 97. *For any distributions Δ and Δ' over a set S, we have*

- *for all couplings (X, Y) of Δ and Δ', that*

$$t(\Delta, \Delta) \leq \Pr[X \neq Y]$$

- *that there is a coupling (X', Y') of Δ and Δ' satisfying that*

$$t(\Delta, \Delta) = \Pr[X' \neq Y']$$

Because of our rounding, we have that $t(X'_{x,T'}, Y'_{x,T'}) < \frac{m}{2q}$. Using that with the coupling lemma (the second part to be precise), lets us find a coupling $C_{x,T'} = (X'_{x,T'}, Y'_{x,T'})$ such that $\Pr[X'_{x,T'} \neq Y'_{x,T'}] < \frac{m}{2q}$.

Consider the plays π_1, π_2 for when Eve follows σ or σ' respectively. We can view the first T steps of these plays by considering $X'_{x,T'}$ instead of $X_{x,T'}$ and similar when Eve follows σ'. We can therefore see that the first T steps two plays are different with probability $= p < \frac{mT}{2q} = \varepsilon/2$ using union bounds.

We therefore see that the value for the path π_1 cannot differ from the value of π_2 with more than $p\gamma\sum_{i=1}^{T}(1-\gamma)^i = p$. I.e. in the worst case, if π_1 and π_2 differs, the reward is 1 in each step for π_1 but 0 in each step for π_2. Also, the rewards in the steps after step T can also differ by at most 1 and by our choice of T, we have that outcome contributed from these remaining steps are worth less than $\varepsilon/2$ as well. Hence, we see that σ' obtains the same as σ except for ε against any strategy τ and is thus ε-optimal. □

There is a classic problem in geometry called the sum-of-square-roots problem. The problem is defined as follows: Let $a, b_1, b_2, \ldots, b_n$ be natural numbers. Is $\sum_{i=1}^{n} \sqrt{b_i} > a$?

The problem comes up for decision problems about distances in Euclidean space. It is not known to be in P or NP for that matter, but is in the fourth level of the countering hierarchy, slightly inside PSPACE. The issue is in essence that it is not known how good an approximation of $\sqrt{b_i}$ is necessary to decide the strict inequality.

We will use the sum-of-square-roots problem to give an informal hardness argument, in that finding the exact value of a concurrent game is in general harder than solving the sum-of-square-roots problem.

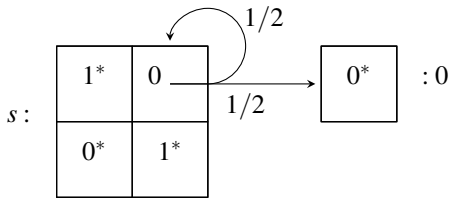


Figure 7.2: Concurrent discounted game with value $v_s = -2 + \sqrt{4+2(1-\gamma)}$

Consider the following game G: There are three vertices, $\{0,1,s\}$ where 0 and 1 are absorbing, with color 0 and 1 respectively. The vertex s is such that (1) $c(s,i,j) = 0$, (2) $\Delta(s,i,i) = 1$ (for $i \in \{1,2\}$), (3) $\Delta(s,2,1) = 0$ and (4) $\Delta(s,1,i)$ is the uniform distribution over s and 0. The game is illustrated in Figure 7.2.

Consider an optimal stationary strategy in G for Eve. Let p be the probability with which she plays the first action. If Adam knows that Eve will follow this strategy, the game devolves into a MDP. We know from that for such there exists optimal positional strategies and thus Adam is either going to play the left or right column always. Clearly, $0 < p < 1$ because $p = 0$ or 1 means that either playing the left or right column with probability 1 would ensure that no positive reward ever happens.

Let $v_0 = 0, v_1 = 1, v_s$ be the values of the three vertices. If he plays the left column, the outcome is $p(1-\gamma)$. If he plays the right column, the outcome is $p/2(1-\gamma)v_s + (1-p)(1-\gamma)$. Observe that the former is increasing in p and the latter is decreasing (since clearly, $0 < (1-\gamma)v_s < v_s$). Also, both are continues. Thus, the optimum is for $p(1-\gamma)$ to be equal to $p/2(1-\gamma)v_s + (1-p)(1-\gamma)$ and both equal to v_s. We will first isolate v_s in $v_s = p/2(1-\gamma)v_s + (1-p)(1-\gamma)$.

$$v_s = p/2(1-\gamma)v_s + (1-p)(1-\gamma) \Rightarrow (1-p/2(1-\gamma))v_s = (1-p)(1-\gamma) \Rightarrow$$
$$v_s = \frac{(1-p)(1-\gamma)}{1-p/2(1-\gamma)} \ .$$

Note that $p, \gamma < 1$ thus, $1 - p/2(1-\gamma) \neq 0$. We then have the equality

$$\begin{aligned}
\frac{(1-p)(1-\gamma)}{1-p/2(1-\gamma)} &= p(1-\gamma) \Rightarrow \\
(1-p)(1-\gamma) &= p(1-\gamma)(1-p/2(1-\gamma)) \Rightarrow \\
0 &= \frac{1-\gamma}{2}p^2 + 2p - 1 \Rightarrow \\
p &= \frac{-2 \pm \sqrt{4+2(1-\gamma)}}{1-\gamma} \ .
\end{aligned}$$

We see that $\frac{-2-\sqrt{4+2(1-\gamma)}}{1-\gamma} < 0$. Thus, $p = \frac{-2+\sqrt{4+2(1-\gamma)}}{1-\gamma}$. Also,

$$v_s = -2 + \sqrt{4+2(1-\gamma)} \ .$$

It is straightforward to modify the construction to get any square-root desired for a fixed γ.

s :			
	1^*	0	0^*
	1^*	0^*	0
	0^*	1^*	1^*

Figure 7.3: Alternate concurrent discounted game with value $v_s = -2 + \sqrt{4 + 2(1-\gamma)}$

By making such a construction for each number $\sqrt{b_i}$, we can make another vertex s^* that has the value of $(1-\gamma)\frac{\sum_{i=1}^{n}\sqrt{b_i}}{n}$ with a single action for each player and that goes to a uniformly random vertex. Observe that decreasing each reward by x, reduces the value of each vertex by x. Reduce each reward by $\frac{an}{1-\gamma}$. We can then decide the sum-of-square-roots problem by deciding whether the value of s^* is strictly above 0.

We get the following lemma.

Lemma 98. *The (exact) decision problem for the value is sum-of-square-root hard for concurrent discounted payoff games.*

We will use this game G as an example to illustrate how to make the Δ-function deterministic for concurrent games while having the same value and a similar optimal strategy.

Consider the following game G': There are three vertices, $\{0,1,s\}$ where 0 and 1 are absorbing, with color 0 and 1 respectively. The vertex s is such that (1) $c(s,i,j) = 0$ for all i, j, (2) $\Delta(s,i,j) = s$ for $i+1 = j$ (i.e. for $(i,j) \in \{(1,2),(2,3)\}$), (3) $\Delta(s,i,j) = 0$ for $i+j = 4$ (i.e. the "other" diagonal, $(i,j) \in \{(3,1),(2,2),(1,3)\}$) and (4) $\Delta(s,i,j) = 1$ otherwise (i.e. for $(i,j) \in \{(1,1),(2,1),(3,2),(3,3)\}$). The game is illustrated in Figure 7.3.

We will argue that the value of G' is equal to that of G. We clearly have that the value of s is in $(0,1)$. Consider a stationary strategy σ for Eve such that $\sigma(1) \neq \sigma(2)$. Let $p_i = \sigma(i)$ for $i \in \{1,2,3\}$. Let σ' be such that $\sigma'(3) = p_3$ and otherwise, $\sigma'(i) = \frac{p_1+p_2}{2}$ for $i \in \{1,2\}$. Let $p'_i = \sigma'(i)$ for $i \in \{1,2,3\}$.

Claim 2. *The strategy σ' is at least as good as σ.*

Proof. If Adam plays 1, then the expected outcome is $p_1 + p_2 = p'_1 + p'_2$ no matter if Eve plays σ or σ'. If he plays i for $i \in \{2,3\}$, the expected outcome is $\frac{p_{4-i}}{1-p_{i-1}}$ if Eve plays σ and otherwise, if she plays σ', the expected outcome is $\frac{p'_1}{1-p'_2} = \frac{p'_2}{1-p'_1}$. Note that $\frac{p'_1}{1-p'_2} > \min_{i\in\{2,3\}} \frac{p_{4-i}}{1-p_{i-1}}$ and thus, σ' is at least as good a strategy as σ. □

A similar argument shows that for any strategy τ for Adam the similar strategy τ' where $\tau'(1) = \tau(1)$ and $\tau'(i) = \frac{\tau(2)+\tau(3)}{2}$ for $i \in \{2,3\}$ is at least as good as τ. Consider

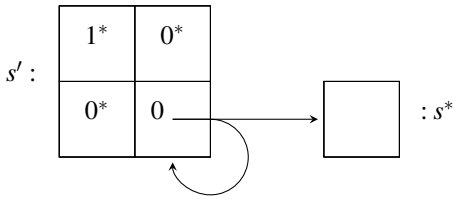


Figure 7.4: Concurrent discounted game that implies that if there is an exponential lower bound on patience, then the sum-of-square-roots problem is in P

that the players follows such stationary strategies σ' and τ'. Let σ be

$$\sigma(i) = \begin{cases} \sigma'(1) + \sigma'(2) & \text{if } i = 1 \\ \sigma'(3) & \text{if } i = 2 \end{cases}.$$

Similarly, let τ be

$$\tau(i) = \begin{cases} \tau'(1) & \text{if } i = 1 \\ \tau'(2) + \tau'(3) & \text{if } i = 2 \end{cases}.$$

But playing σ and τ in G gives the same outcome as playing σ' and τ' in G' as can be seen as follows: In either game, with probability

$$\sigma'(1)\tau'(2) + \sigma'(2)\tau'(3) = \frac{\sigma(1)\tau(2)}{2}$$

we play again with a reward of 0, with probability

$$\sigma'(1)\tau'(3) + \sigma'(2)\tau'(2) + \sigma'(3)\tau'(1) = \frac{\sigma(1)\tau(2)}{2} + \sigma(2)\tau(1)$$

we get absorbed in 0 after a reward of 0 and with probability

$$(\sigma'(1) + \sigma'(2))\tau'(1) + (\tau'(2) + \tau'(3))\sigma'(3)$$

we get absorbed in 1 after a reward of 0. But this is in particular the case if the players play optimally and thus, the value is the same in the two games.

Before, in Corollary Lemma 96, we argued that the patience of ε-optimal stationary strategies was $q = \frac{m\log(\varepsilon/2)}{\log(1-\gamma)\varepsilon}$. Giving a similar exponential bound for the optimal stationary strategies is harder than solving the sum-of-square-roots problem, as we will argue next. Assume that we had an exponential bound for optimal stationary strategies.

Consider an arbitrary yes-instance of the sum-of-square-roots problem, giving a vertex s^*. Reduce each reward by a and in the new game let s^*_a be the vertex corresponding to s^*. We will now create a game that uses the previous game as a sub-game. The game has 1 additional vertex s', which is a 2x2-matrix, such that $c(s', i, j) = 0$ and $\Delta(s, 1, 1) = 1$ and $\Delta(s, 2, 2) = s^*$ and $\Delta(s, i, j) = 0$ for $i \neq j$. There is an illustration in Figure 7.4, using the vertex s^* as above. Using an argument like above, we see that the probability p to play the top action in the vertex s' is such that $p(1-\gamma) =$

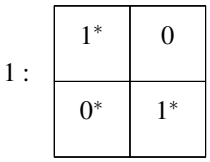


Figure 7.5: The snowball game or purgatory 1, in which no optimal strategy exists for Eve

$(1-p)(1-\gamma)x$, where x is the value of s^*. Thus, $x = \frac{p}{1-p}$. If p only needs to be exponential small, then x is exponentially small as well. This is true for any yes-instance of the sum-of-square-roots problem and thus, we only need polynomially many digits to decide the problem. We can find polynomially many digits of $\sqrt{b_i}$ for each i in polynomial time. We get the following lemma.

Lemma 99. *Giving an exponential lower-bound on patience for optimal stationary strategies in concurrent discounted payoff games implies that the sum-of-square-roots problem is in* P

7.4 Concurrent reachability games

In this section we consider concurrent reachability games. Intuitively, unlike concurrent discounted payoff games, these games cares only about the final part of the play. This, while perhaps not clear directly from the definitions, makes the games somewhat harder. For instance, the value iteration algorithm requires double-exponential time. Note that like for concurrent discounted payoff games, if we force Eve to follow some strategy, the game reduces to a MDP and there exists optimal positional strategies.

We will not prove the following known lemma. We will however, show the weaker statement that the decision problem for the value can be done in PSPACE.

Lemma 100 (Properties of concurrent reachability games)**.** *Concurrent reachability games are determined. Also, finding the value of a concurrent reachability game can be done in TFNP[NP].*

We will argue that there might not be optimal strategies for Eve in concurrent reachability games. The game we will use will later be a member of a family of games that requires high patience to play well.

The snowball game (or purgatory 1) is defined as follows: There are 3 vertices, the goal vertex Win, $\bot$ (an absorbing vertex) and a vertex 1, which has a 2x2 matrix, such that $\Delta(x,r,c)$ is a Dirac distribution over (1) Win for $r = c$, (2) 1 for $r < c$ and (3) $\bot$ for $r > c$. When we illustrate the game, we write view the goal vertex Win as being an absorbing vertex with color 1. There is an illustration of the snowball game in Figure 7.5.

For any $\varepsilon > 0$, consider the stationary strategy for Eve that plays the first action with probability $1-\varepsilon$. If Adam plays the left column always, play will reach Win with

pr. $1-\varepsilon$. If Adam plays the right column always, play reaches Win with pr. 1. Hence, the strategy for Eve is ε-optimal and the value of the vertex is 1.

On the other hand, if Adam plays the right column whenever Eve plays the first action with pr. 1, and otherwise plays the first action, then in the last round in the vertex there must be a positive pr. that play goes to vertex 0. Hence, Eve has no optimal strategy.

Lemma 101 (No optimal strategies in concurrent reachability games)**.** *Eve need not have an optimal strategy in a concurrent reachability game.*

The following lemma states some classical results for concurrent reachability games that we will not prove.

Lemma 102 (Classical results for concurrent reachability games)**.**

- *For any $\varepsilon > 0$, there are always ε-optimal stationary strategies for Eve and optimal stationary strategies for Adam.*
- *The value iteration algorithm converges to the optimal value and is defined exactly like for concurrent discounted payoff games, except that $\gamma = 0$ and the target vertex has value 1 in the first iteration.*
- *The values v are the least fixed point (i.e., every other fixed point v' is such that for all i $v_i \leq v'$) of the value iteration operator valOp.*

(Note that the games are not symmetric, in that Eve tries to reach a node and Adam tries to stay away from it, and in particular, even though Eve need not have an optimal strategy in a concurrent reachability game, Adam always has one.)

The decision problem for the existential first order theory over the reals is the following decision problem: Given a function $F : \mathbb{R}^n \to \{\text{true}, \text{false}\}$, is there an vector v such that $F(v)$ is true? The function F must be an well-formed (i.e. connected with logical 'and', 'or' and 'not') quantifier-free formula over polynomial inequalities. E.g. $x^2y + z \geq 5 \wedge \neg(xz \leq 3)$ would be such a function.

Lemma 103. *The decision problem for the existential theory over the reals is in PSPACE*

We will now, given a number c, encode the problem whether the value in a concurrent reachability game is $< c$, starting from some vertex x. The idea is that we can describe a fixed point of the value iteration operator, i.e. we can describe that $v_i = \text{val}[M^i(v)]_i$. Since we know that the values are the least fixed point, we can then just add the condition that $v_x < c$. We can describe the value of a matrix game by guessing a strategy for each player, σ for Eve and τ for Adam, and then checking that these strategies are optimal by showing that the outcome obtained by following Eve's strategy is equal to what is obtained by following Adam's. I.e. we check that for Eve's strategy, the least outcome obtained when Adam plays any given column is v_i and similar for Adam. We describe that as $v_i = \min_{a \in [m]}(\sigma M^i_{*,a}(v))$ and $v_i = \max_{a \in [m]}(\tau M^i_{a,*}(v))$, where $M^i_{*,j}(v)$ is the j-th column of $M^i(v)$ and $M^i_{j,*}(v)$ is the j-th row for all j. We can express that $x = \min(a_1, a_2, \ldots, a_n)$ for any number n and any polynomials a_1, a_2, a_n, in the first order theory of the reals by stating that

$$x \leq a_1 \wedge x \leq a_2 \wedge \cdots \wedge x \leq a_n \wedge (x = a_1 \vee x = a_2 \vee \cdots \vee x = a_n) \ .$$

Similarly, one can also describe that $x = \max(a_1, a_2, \ldots, a_n)$ for any number n and any polynomials a_1, a_2, a_n.

We can similarly make a statement that $v_x \leq c$. Using that PSPACE is equal to co-PSPACE, we also get that we can find if $v_x \geq c$ and $v_x > c$ and therefore also $v_x = c$ and $v_x \neq c$, all in PSPACE.

Lemma 104. *Decision problems for the values in a concurrent reachability game is in PSPACE.*

The set of vertices that have value 0 can be found in polynomial time. This is because, the set of vertices that have value 0 in the time limited game of length n has also value 0 in all other time limited games. That this is so is easy to see by considering that Eve plays an ε-optimal strategy for $v > \varepsilon$, where v is the value of the vertex with the lowest value. The game then devolves to a MDP for Adam, and the statement is true for such.

Lemma 105 (Value 0 vertices in concurrent reachability games)**.** *The set of vertices of value 0 in a concurrent reachability game can be found in polynomial time.*

Next, we argue that we can find the set of vertices S_1 that have value 1 in polynomial time as well. For notational convenience, for stationary strategies σ, τ we will, for a set x and a set of vertices S use

$$F^{\sigma,\tau}(x,S) := \sum_{r \in A_1} \sum_{c \in A_2} \sum_{x' \in S} \sigma(x)(r)\tau(x)(c)\delta(x,r,c)(x') \ ,$$

i.e. the probability when the players follows σ, τ to go from x directly to a vertex in S.

For a set S, containing Win and a non-empty subset $S' \subseteq (S \setminus \{\text{Win}\})$ and a vertex $x \in S'$, let the *subset property* be the following: For each $\varepsilon > 0$, there is a strategy σ for Eve, such that for any strategy τ for Adam, $F^{\sigma,\tau}(x, S \setminus S') \cdot \varepsilon > F^{\sigma,\tau}(x, V \setminus S)$.

For a set S, containing Win, let the *value-1-property* be that the subset property is satisfied for each $S' \subseteq (S \setminus \{\text{Win}\})$ (with some $x \in S'$). For a set S satisfying the value-1-property, we will define some subsets. Let S^0 be the set consisting of Win. Let S^i, for each $i \geq 1$ be the set of vertices such that for each $x \in S^i$, the vertex x satisfies the subset property for S' and $S = \bigcup_{j=0}^{i-1} S^j$. Let ℓ be the largest number such that S^ℓ is non-empty (note that S^i, for $i > \ell$ must then be empty).

We will be using the following lemma.

Lemma 106. *The set of vertices S_1 of value 1 satisfies the value-1-property.*

Proof. The proof is by contradiction. Thus, there is an S (not containing Win) such that S_1 and S does not satisfy the subset property for any $x \in S$. I.e. for some constant $\varepsilon > 0$, $F^{\sigma,\tau}(x, S_1 \setminus S)\varepsilon \leq F^{\sigma,\tau}(x, V \setminus S_1)$ for any strategy σ for Eve and some strategy τ_σ for Adam and for every $x \in S$. But then, all vertices in S have value $\leq (1-\varepsilon) + \varepsilon V_{\max}$ where $v_{\max} < 1$ is the largest value of a vertex in $V \setminus S_1$. This is because to get to Win from S, it must leave S and in that step, the probability to go to a vertex in $V \setminus S_1$ (from which one cannot obtain more than $v_{\max}$) is at least the constant ε. □

We will argue that this is a precise characterization of S_1 next.

Lemma 107. *Consider a set S' satisfying the value-1-property, then each vertex of S' has value 1.*

Proof. We will for all i and any $\varepsilon > 0$ construct a strategy $\sigma_{i,\varepsilon}$ for Eve that starting from a vertex in $\bigcup_{j=i}^{n} S^j$ will eventually get to S^{i-1} with probability at least $1-\varepsilon$ (especially, the strategy $\sigma_{1,\varepsilon}$ is ε-optimal). We will do it using backwards induction in i and thus start from $i = \ell$.

Note that the base case for $i = \ell$ follows directly from the condition. We now for any $\varepsilon > 0$ will find a strategy $\sigma_{i,\varepsilon}$, given that we have a strategy $\sigma_{i+1,\varepsilon'}$ for any $\varepsilon' > 0$. The idea is that the subset property gives a strategy σ such that for all τ, $F^{\sigma,\tau}(x, S' \setminus S)\varepsilon/2 > F^{\sigma,\tau}(x, V \setminus S')$, for $S = \bigcup_{j=i}^{n} S^j$. Note that in expectation, following this strategy, we go to some other vertex in S with at least some fixed probability p (that could be quite close to 1). Hence, in expectation, we need to be in such a vertex $1/(1-p)$ times before entering either $S' \setminus S$ or $V \setminus S'$. We therefore follow the strategy $\sigma_{i+1,\varepsilon'}$ in $\bigcup_{j=i+1}^{n} S^j$, where $\varepsilon' = \frac{\varepsilon}{2(1-p)}$. The inductive construction then follows by applying union bound over the $1/(1-p)$ times we are in S^i. □

Note that the lemmas together show that S_1 is the largest set satisfying the value-1-property.

Our algorithm, $\mathscr{A}_1$, for finding S_1 is then as follows: Assign to each vertex x a rank rk_x, a value in $0, 1, \ldots, n, \bot$, starting with $\mathrm{rk}_x = 0$. Let $\overline{S}^i$ be the set of vertices of rank i. Let $\overline{S}_1 = V \setminus S^{\bot}$. We increment (where a rank is less than another if it is a smaller number. Also, all other ranks are below $\bot$) the rank of a non-goal vertex $x \in \overline{S}^i$, whenever it does not satisfy the subset property for $S = \overline{S}_1$ and $S' = \bigcup_{j=i}^{n} \overline{S}^j$. Note that no vertex can satisfy the subset property for rank 0, since S' is all vertices. Whenever a stable configuration is reached, output $\overline{S}_1$.

Lemma 108. *The output of the $\mathscr{A}_1$ algorithm is correct.*

Proof. The idea is that we want $S^i = \overline{S}^i$ at termination. Note that the subset property is harder to satisfy for a vertex x if we remove vertices from S or from $(S \setminus S')$. Thus, if at some time we have that x does not satisfy the subset property for $S = \overline{S}_1$ and $S' = \bigcup_{j=i}^{n} \overline{S}^j$, then it does not do so for S being any subset of $\overline{S}_1$ or $(S \setminus S')$ being any subset of $\bigcup_{j=0}^{i-1} \overline{S}^j$. However, initially $S_1 \supseteq \overline{S}_1$ (being all vertices) and $\bigcup_{j=i}^{n} S^j \supseteq \bigcup_{j=0}^{i-1} \overline{S}^j$ for all i. But we must have that $S_1 \supseteq \overline{S}_1$ and $\bigcup_{j=i}^{n} S^j \supseteq \bigcup_{j=0}^{i-1} \overline{S}^j$ for all i, at all latter points as well, since in the last iteration it was satisfied, for all i and all $x \in S^i$, we have that the subset property is satisfied for $S = \overline{S}_1$ and $S' = \bigcup_{j=i}^{n} \overline{S}^j$, because we have that $S \supseteq S_1$ and $(S \setminus S') = \bigcup_{j=1}^{i-1} \overline{S}^j \supseteq \bigcup_{j=1}^{i-1} S^j$. On the other hand, eventually no vertex gets it rank incremented (since there are a finite number of ranks and vertices) and the algorithm terminates with a set $\overline{S}_1 \supseteq S_1$ satisfying the value-1-property. Since S_1 is the largest such set, we have that $\overline{S}_1 = S_1$. □

We will now consider the running time of the algorithm. We will consider that we can decide whether a pair of sets S and S' satisfies the subset property for a vertex x can be solved in $O(k)$ time. Let S_x be the set of vertices that can be visited in a play

immediately after x. Observe that $|S_x|$ is a lower bound on the number of arrows from matrix x in our illustration of the game. Consider a vertex x and a rank i, and we want to find an upper bound on the computation we do on x while it has rank i. Clearly, we only need to consider incrementing the rank of x whenever a vertex in S_x has changed its value and only if it is changed to either i (from $i-1$) or to $\perp$ (from n), because otherwise, the sets S and S' have not changed. Thus, we only do at most $2|S_x|+1$ checks whether x satisfies the subset property. There are $n+1$ ranks, so in total for x, we use at most $(n+1)(2|S_x|+1)$ checks. Hence, in total over all x, we do $O(n\sum_x |S_x|)$ checks.

Lemma 109. *The runtime of the $\mathscr{A}_1$ algorithm is $O(nk\sum_x |S_x|)$.*

We will then finally consider how to check whether x satisfies the subset property for a pair of sets S and S'. We will do so by constructing a strategy $\sigma = \sigma(\varepsilon)$ for Eve satisfying the property for any fixed $\varepsilon > 0$. We will construct the strategy σ from some sequence of pairs of sets (of rows and columns) $(R_1,C_1),(R_2,C_2),\ldots,(R_\ell,C_\ell)$. We will let $C^*_i = \bigcup_{j=1}^{i} C_j$ and similar for R^*_i. For convenience, we also define C^*_0 as the empty set of columns. We will define R_i from C^*_{i-1} as each row $r \notin R^*_{i-1}$ such that, for all $c \notin C^*_{i-1}$, we have that $F^{r,c}(x, V \setminus S) = 0$. We will define $C_i \notin C^*_{i=1}$ from R_i as each column c such that there is a $r \in R_i$ such that $F^{r,c}(x, S \setminus S') > 0$. The set R^*_ℓ is the first set such that $R^*_{\ell+1}$ is empty (clearly, by construction all sets R_i, C_i for $i > \ell$ would also be empty).

Lemma 110. *There is a strategy $\sigma(\varepsilon)$ for all $\varepsilon > 0$ if and only if C^*_ℓ is the set of all columns.*

Proof. We will first argue that if C^*_ℓ is not all columns C, then the strategy τ that plays uniformly over $C' = (C \setminus C^*_\ell)$ shows that no strategy $\sigma(\varepsilon)$ exists for small enough $\varepsilon > 0$. This is because any row r such that $F^{r,c}(x, S \setminus S') > 0$ for some $c \in C'$ is also such that $F^{r,c'}(x, V \setminus S) > 0$ for some column $c' \in C'$. This is because otherwise r would be in R^i for some i and then c would be in C^*_ℓ. Hence, the probability $F^{r,c'}(x, V \setminus S)$ cannot be more than a constant factor smaller than $F^{r,c}(x, S \setminus S')$.

Otherwise, if $C^*_\ell = C$, then let $\sigma(\varepsilon)$ be the strategy that picks an i from the distribution $\mathscr{D}$ and then plays an action in R^i uniformly at random. The distribution $\mathscr{D}$ is such that for all $j \in \{1,\ldots,\ell-1\}$ we have $\Pr^{\mathscr{D}}[i = j]\varepsilon\delta_{\min}/m = \Pr^{\mathscr{D}}[i > j]$.

To argue that $\sigma = \sigma(\varepsilon)$ satisfies the subset property for x, S, S' consider each column c. We have that $c \in C^j$ for some j. Let p be the pr. with which a row in R^j is played by σ and thus, $F^{\sigma,c}(x, S \setminus S') \geq p\delta_{\min} > 0$. Any row r such that $F^{r,c}(x, V \setminus S) > 0$ must be outside R^*_j by construction (and if such a row exists $j < \ell$). We play such rows with pr. $\leq \Pr^{\mathscr{D}}[i > j]$ and thus $\Pr^{\mathscr{D}}[i > j] \geq F^{\sigma,c}(x, V \setminus S)$. We have that $pm > \Pr^{\mathscr{D}}[i = j]$ (strict because $j < \ell$) and thus,

$$F^{\sigma,c}(x, S \setminus S')\varepsilon \geq p\delta_{\min}\varepsilon > \delta_{\min}\varepsilon/m \Pr^{\mathscr{D}}[i = j] \geq \Pr^{\mathscr{D}}[i > j] \geq F^{\sigma,c}(x, V \setminus S) \ .$$

This completes the proof of the lemma. □

Our algorithm for checking if a vertex x satisfies the subset property for sets S, S' is as follows: We will construct the sequence of sets $(R_1,C_1),(R_2,C_2),\ldots,(R_\ell,C_\ell)$. To do

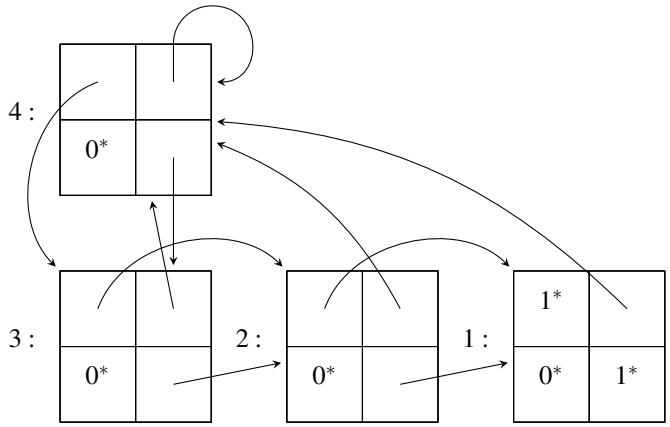


Figure 7.6: Purgatory 4. For clarity, the colors are omitted, except that 0^* corresponds to an edge to an absorbing vertex different from Win and 1^* corresponds to an edge to Win

so we will use a data structure. The data structure has the following properties: Initially, for each column c, we will make a list L_c of the rows r such that $F^{r,c}(x, V \setminus S) > 0$. We will also have a counter for each row r that initially contains how many such columns there are. Finally for each row r, there is a list L_r of columns such that $F^{r,c}(x, S \setminus S') > 0$.

The algorithm then uses the data structure as follows: Let $i \leftarrow 1$. Add all the rows with the counter at 0 to R^i. If R^i is the empty set, return whether C^*_{i-1} is all columns. For each row r in R^i go through $c \in L_r$ and subtract 1 from the counter of each row in L_c. If a counter reach 0, add it to R^{i+1}. Increment i. Go to line 3.

The total running time the algorithm is $O(\sum_{r,c} |\text{supp}(\Delta(x,r,c))|)$.

Lemma 111. *We can check whether a vertex x satisfies the subset property for sets S and S' in time*

$$O(\sum_{r,c} |supp(\Delta(x,r,c))|).$$

We therefore get that

Lemma 112. *We can compute the set of vertices of value 1 in time*

$$O\left(n \sum_x |S_x| \sum_{r,c} |supp(\Delta(x,r,c))|\right).$$

Next, we will give a lower bound for patience: we prove that in some games, the patience for every ε-optimal stationary strategy must be high. For a number k, let purgatory k be the following game: There are $2+k$ vertices, Win (which is vertex 0), one vertex $\perp$ which is absorbing and each other vertex $i \in \{1, \ldots, k\}$ has a 2x2 matrix, such that $\Delta(x,r,c)$ is a Dirac distribution over (1) $i-1$ for $r = c$, (2) k for $r < c$ and (3) $\perp$ for $r > c$. There is an illustration of Purgatory 4 in Figure 7.6.

It is easy to see that all vertices but $\perp$ are in S_1 using the value 1 property.

Lemma 113 (Properties of the purgatory game)**.** *For any $0 < \varepsilon < 1/2$ and any $k \geq 1$, there is a unique strategy for Eve with least patience which is ε-optimal in purgatory k. That strategy has patience $\varepsilon^{-2^{k-1}}$*

Proof. We will find the best strategy with patience $1/p$ for any number $0 < p < 1/2$. It is clear that the best strategy with patience $1/p$ is to play the strategy that maximizes the pr. of eventually reaching i from vertex j for all $j > i$ while having patience $1/p$. Let $x_k = 1 - p$ and let $x_i = 1 - \sqrt{1 - x_{i+1}}$. We will argue that x_i is the probability of eventually reaching vertex $i-1$ from vertex k for all $i \in \{1, \ldots, k\}$ and that there is a unique best strategy with patience $1/p$.

The unique best strategy in vertex k is to play the top action with pr. $1-p$ and the bottom action with pr. p. This ensures that the pr. x_k of reaching $k-1$ from k is $1-p$ if Adam plays the left column.

Consider now vertex $i \in \{1, \ldots, k-1\}$. For the purpose of finding good strategies in i, we can view vertex i, when Eve plays her best strategy in $j > i$ and Adam plays a best response, as a smaller reachability game with 3 vertices, i.e. $i-1$ (as Win), $\perp$ and i, where $\Delta(i, r, c)$ is (1) a Dirac distribution over $i-1$ for $r = c$, (2) a Dirac distribution over $\perp$ for $r > c$ and (3) a distribution that goes to $\perp$ with pr. $1 - x_{i-1}$ and to i with the remaining pr. for $r < c$. See Figure 7.6.

Let p_i be the probability with which a strategy σ plays the top row in vertex i. We can then consider the game as a MDP, since we have fixed a stationary strategy for one of the players. It is clear that the pr. to reach $i-1$ from i if Adam plays the right column is strictly increasing in p_i and the pr. to reach $i-1$ from i if Adam plays the left column is strictly decreasing in p_i. We will consider the strategy such that the pr. of reaching $i-1$ is equal no matter which column Adam plays (by the previous statement, this strategy must then be optimal). Observe that the pr. of reaching $i-1$ if Adam always plays the left column is p_i (which is then also the pr. to reach $i-1$ from i) and if he always plays the right column it is $p_i x_i p_i + (1 - p_i)$ (using that the pr. of reaching $i-1$ from i is p_i). Thus, we have that

$$p_i = p_i x_i p_i + (1 - p_i) \Rightarrow 0 = p_i^2/2x_i - p_i + 1/2 \Rightarrow p_i = \frac{1 \pm \sqrt{1 - x_i}}{x_i} \ .$$

We see that $\frac{1+\sqrt{1-x_i}}{x_i} > 1$ and thus the solution is $p_i = \frac{1-\sqrt{1-x_i}}{x_i}$ or that $(x_{i-1} =) x_i p_i = 1 - \sqrt{1 - x_i}$.

Consider now that the strategy is exactly ε-optimal, implying that $x_1 = 1 - \varepsilon$. We will argue that $p = \varepsilon^{2^{k-1}}$. We will do so by arguing using induction in i that $x_i = 1 - \varepsilon^{2^{k-1}}$, since $x_k = 1 - p$ (this also shows that the strategy is indeed using patience $1/p$ since the probabilities in the other vertices, which are $p_i = \frac{x_{i-1}}{x_i}$, are strictly above $1/p$). We have already noted that $x_1 = 1 - \varepsilon = 1 - \varepsilon^{2^0}$. We will next argue that $x_i = 1 - \varepsilon^{2^{k-1}}$, for $i \geq 2$ using that $x_{i-1} = 1 - \varepsilon^{2^{k-2}}$. We have that

$$1 - \varepsilon^{2^{k-2}} = x_{i-1} = 1 - \sqrt{1 - x_i} \Rightarrow \sqrt{1 - x_i} = \varepsilon^{2^{k-2}} \Rightarrow 1 - \varepsilon^{2^{k-1}} = x_i \ .$$

This completes the proof of the lemma. □

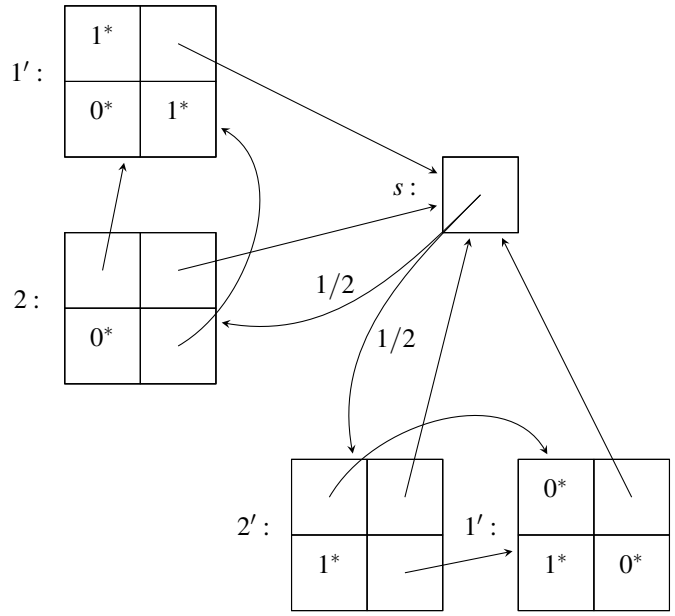


Figure 7.7: Purgatory duel 2. For clarity, the colors are omitted, except that 0^* corresponds to an edge to an absorbing vertex different from Win and 1^* corresponds to an edge to Win

Concurrent reachability games are not symmetric in the players. E.g. Adam always have an optimal strategy but Eve might not. We will next argue that Adam still requires double exponential patience to play well.

Consider the following game called purgatory duel k which can be viewed as a symmetric version of purgatory k. There are $3+2k$ vertices, Win (which is vertex 0), one vertex $\perp$ which is absorbing (and is also vertex $0'$), and the start vertex s and each other vertex $\{1,\dots,k,1',\dots,k'\}$ has a 2x2 matrix. Each vertex $x \in \{1,\dots,k\}$ is such that $\Delta(x,r,c)$ is a Dirac distribution over (1) $x-1$ for $r=c$, (2) s for $r<c$ and (3) $\perp$ for $r>c$. Each vertex $x' \in \{1',\dots,k'\}$ is such that $\Delta(x',r,c)$ is a Dirac distribution over (1) $x-1'$ for $r=c$, (2) s for $r<c$ and (3) Win for $r>c$. The start vertex is 1x1 matrix and is such that $\Delta(s,r,c)$ is a uniform distribution over k and k'. There is a illustration of Purgatory Duel 2 in Figure 7.7.

We will say that a strategy σ for Eve mirrors a strategy τ for Adam, if $\sigma(i)=\tau(i')$ and $\sigma(i')=\tau(i)$ for all i. Similarly, τ mirrors σ.

Lemma 114. *The value of vertex s is $1/2$. Also, for any $\varepsilon>0$, any $(1/2-\varepsilon)$-optimal strategy τ for Adam does not follow a Dirac distribution in i' for any $i' \in \{1',\dots,k'\}$. Finally, every ε-optimal strategy is a mirror of an ε-optimal strategy*

Proof. First, the value of vertex s is at most $1/2$. This is because Adam can mirror any strategy σ for Eve. This ensures that any play reaching Win is mirrored by an equally likely play reaching $\perp$. Thus, then the players follows these strategies, the pr. to reach

Win is equal to the pr. to reach $\perp$ (there might also be some positive pr. to not reach neither, but Adam also wins those plays).

Fix $\varepsilon > 0$ and consider a strategy τ for Eve that plays a Dirac distribution in $i' \in \{1', \ldots, k'\}$. Then τ is not $(1/2-\varepsilon)$-optimal. We can see that as follows: Let σ be the strategy for Eve that plays $r = 1$ when in $j' \in \{(i+1)', \ldots, k'\}$ and the action which is not equal to $\tau(i')$ when in i'. This ensures that the play will always reach either Win or s from k'. But then Eve can play an $(\varepsilon/2)$-optimal strategy for purgatory k in $\{1, \ldots, k\}$, ensuring that Win is reached with pr. at least $1-\varepsilon/2$. But then τ is not $(1/2-\varepsilon)$-optimal.

Consider that Adam is following a strategy τ that is not playing a Dirac distribution in $i' \in \{1', \ldots, k'\}$ and Eve is playing an arbitrary strategy σ. Then, eventually play reaches Win or $\perp$ with pr. 1, because in every $k+1$ steps, either s is visited or either Win or $\perp$ is reached, and after s has been visited, k' is next half the time and from k' $\perp$ is reached with positive pr.

Consider an ε-optimal strategy τ for Adam, for $\varepsilon < 1/2$. Then, let Eve's mirror strategy to τ be σ_τ. Now, either Win or $\perp$ is reached and because the strategies mirrors each other, the pr. to reach Win is equal to that of reaching $\perp$. Thus, we see that the value is at most $1/2$, implying that it is exactly $1/2$.

It also follows that the strategies that are ε-optimal mirrors each other. □

We will now argue that Eve's (and thus Adam's) $\frac{1}{4}$-optimal strategies requires high patience.

To do so we will use the following lemma, showing that you can sometimes modify a concurrent game (or any of its special cases) and get a game with less value. While the proof is explicitly for concurrent reachability games, the proof is basically identical for concurrent discounted and mean payoff games. In a game G, for a vertex v and a duration T, let v_G^T be the value of the time-limited game with duration T.

Lemma 115. *Consider a concurrent reachability game G and a pair of vertices u, v, such that for all T, we have that $u_G^T \geq v_G^T$. Consider a vertex w such that for a pair of actions, (r,c) we have that $v \in supp(\Delta(w,r,c))$. Consider an alternate game G' equal to G, except that some of the probability mass is moved from v to u when playing (r,c) in w, i.e. $0 < \Delta(G,w,r,c)(v) - \Delta(G',w,r,c)(v) = \Delta(G',w,r,c)(u) - \Delta(G,w,r,c)(u)$. Then for all vertices z we have that $z_G^T \leq z_{G'}^T$*

Proof. The proof is by induction in T. The proof is trivial for $T = 0$, because $z_G^T = 0 = z_{G'}^T$ for all non-goal vertices (and the goal vertex Win has value 1). For $T \geq 1$, we have that $z_G^{T-1} \leq z_{G'}^{T-1}$. But, matrix games are monotone in their entries, so it follows directly that for $z \neq w$ we have that $z_G^T \leq z_{G'}^T$. Consider the matrix for w_G^T compared to $w_{G'}^T$. All entries but the one for (r,c) are smaller directly by induction. We also have that $v_G^T \leq u_G^T \leq u_{G'}^T$, the first inequality by definition and the second by induction. We thus see that all entries in w_G^T are smaller than in $w_{G'}^T$. The lemma follows. □

We are now ready to find the patience in concurrent reachability games.

Lemma 116. *Any $(1/4)$-optimal strategy, for either player, in purgatory duel k has patience at least $(3/4)^{-2^{k-1}}$ for each k*

Proof. We will show that the lemma is true for Eve's strategies and that it is true for Adam's follows from Lemma 115. Consider an $(1/4)$-optimal strategy σ for Eve. Fixing this strategy for Eve, we get a MDP for Adam. Clearly, in this MDP G', we have that $0 = \bot^T_{G'} \leq s^T_{G'}$ for all T. We can thus apply Lemma 115 to changing $\Delta(1',1,1)$ from $\bot$ to s. In the resulting game G'', we still have that $0 = \bot^T_{G''} \leq s^T_{G''}$ and thus, we can change $\Delta(1',2,2)$ from $\bot$ to s. Let the next game be G^*. Thus, for any $i' \in \{1',\ldots,k'\}$, the plays from i' to $\bot$ in G^* all goes through s. Note that Adam can ensure that the play reaches s from i' and thus, when he plays optimally, do so. Thus, whenever s is entered and Adam plays optimally, k is enter eventually with pr. 1. For the purpose of the value, we can thus disregard s and vertices in $\{1',\ldots,k'\}$ and just view each edge going to s as going to k instead. But the resulting game is purgatory k (in which Eve has fixed his strategy) and Eve is playing a strategy that gives value at least $1/4$, which requires at least $(3/4)^{-2^{k-1}}$ patience, by Lemma 113. □

7.5 Concurrent mean payoff games

In this section we consider concurrent mean payoff games. We will show that in general, any ε-optimal strategy in some concurrent mean payoff games are quite complex. We will first, however, show that finding the value of a concurrent mean payoff game can be done in polynomial space.

Lemma 117 (Properties of concurrent mean payoff games)**.** *Concurrent mean payoff games are determined and the value is the limit of the value of the corresponding time-limited game as well as the limit of the corresponding discounted game, for the discount factor going to 0 from above. There is an polynomial time algorithm similar to Lemma 112, for finding the set of vertices where a finite memory strategy suffice to ensure $1-\varepsilon$ (recall that all rewards are in $\{0,1\}$). For any fixed number n, there is a polynomial time algorithm for approximating the value in a concurrent mean payoff game with n vertices (i.e. the running time is polynomial in the number of actions)*

We will not show this lemma, but simply note that the ε-optimal strategies known for general concurrent mean payoff games can be viewed as playing the corresponding discounted game with a variable discount factor that depends on how 'nice' the rewards has been up to now. Basically, in each round you play the optimal strategy in the corresponding discounted game with a discount factor γ. Whenever your rewards are close to or better than the value, you decrease γ towards 0 and in each round your rewards are much worse than the value you let γ increase, except not bigger than the initial γ in the first round. Much of this section will argue that many natural candidates for simpler types of strategies does not work.

We will show that approximating the value, however, can, as mentioned, be done in polynomial space. The proof relies on Proposition 22 from [HKL$^+$11], stating the following:

Proposition 1. *Let $\varepsilon = 2^{-j}$, where j is some positive integer, and the probabilities be rational numbers where the numerator and denominator have bitsize at most τ. Also, let $\lambda = \varepsilon^{\tau m^{O(n^2)}}$. Consider some state s and let the value of that state in the λ discounted game be v_λ and the value in mean payoff game be v, then $|v - v_\lambda| < \varepsilon$.*

We will use that to again reduce to the existential theory over the reals. For a fixed discount factor γ, we can easily express the value of the corresponding discounted game, like we expressed the value of a concurrent reachability game. We have that the value v is then $v = \lim_{\gamma \to 0^+} f(\gamma)$, where f is the found expression. I.e. for any ε, there is a γ' such that for all $\gamma < \gamma$, we have that $|f(\gamma) - v| \leq \varepsilon$. Also, that $v > c$ means that there is ε, such that $v - \varepsilon > c$.

The problem is thus to come up with a polynomial sized formula to express that λ is $\varepsilon^{\tau m^{O(n^2)}} = 2^{-j\tau m^{O(n^2)}}$.

That can be done as follows, using $\ell = O(n^2) \cdot \log(m) + \log(j\tau)$ many variables, $v_0, v_1, \ldots v_{\ell-1}$:

$$v_0 = 1/2$$

and for all $0 < i < \ell$, we have that

$$v_i = v_{i-1} \cdot v_{i-1}.$$

Using induction, we see that $v_i = 2^{-2^i}$, i.e., $v_1 = 1/2 = 2^{-2^0}$ and

$$v_i = v_{i-1} \cdot v_{i-1} = 2^{-2^{i-1}} \cdot 2^{-2^{i-1}} = 2^{-2^{i-1}-2^{i-1}} = 2^{-2^i}$$

In particular,

$$v_{\ell-1} = 2^{-2^\ell} = 2^{-2^{O(n^2)\cdot\log(m)+\log(j\tau)}} = 2^{-j\tau m^{O(n^2)}}$$

is the value we wanted for λ. Thus, for a given number v, we can test if the value of a concurrent λ-discounted game is above $v + \varepsilon$, which, using the proposition above, implies that v is below the value of the corresponding concurrent mean payoff game. On the other hand, the proposition also implies that if the value of the concurrent λ-discounted game is below $v - \varepsilon$, then the value of the concurrent mean payoff game is below v. Being able to answer these questions lets you easily approximate the value of a concurrent mean payoff using binary search.

We get the following lemma.

Lemma 118. *Approximating the value of a concurrent mean payoff game can be in done in polynomial space*

We will now consider a specific, well-studied example of a concurrent mean payoff game, since it shows that many natural kinds of strategies do not suffice in general. The game is called the big match and is defined as follows: There are 3 vertices, $\{0, s, 1\}$, where the vertices in $\{0, 1\}$ are absorbing, and with value equal to their name. The last vertex s has a 2x2-matrix and for all i, j for $i \neq j$, we have that $c(s, 1, 1) = 1$, and for $i \neq 1 \neq j$ we have that $c(s, 1, 1) = 0$. Also, $\Delta(s, 1, i) = s$ for each i, $\Delta(s, 2, 1) = 0$ and $\Delta(s, 2, 2) = 1$. There is an illustration in Figure 7.8. The value of the Big Match is $1/2$.

Consider a finite-memory strategy σ for Eve. We will argue that σ cannot guarantee ε (any strategy can guarantee -1, since the colors are between 0 and 1) for any $0 < \varepsilon$. Let τ be the stationary strategy for Adam that plays 1 with pr. $\varepsilon/2$. Then playing σ against τ, we get an Markov chain, where the vertex space is pairs of memory states and game vertices. In Markov chains, eventually, with pr. 1, a set of vertices S is

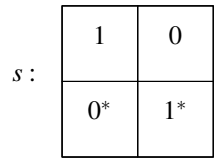


Figure 7.8: The Big Match

reached such that the set of vertices visited infinitely often is S. Such a set is called ergodic. The set S can clearly only contain 1 game vertex, since whenever s is left, it is never entered again. Hence, if S contains s, the pr. that play will ever reach $\{0,1\}$ is 0. In the MC we get from the players playing σ and τ, let $T_{\varepsilon/2}$ be such that with pr. $\varepsilon/2$ some ergodic set has been reached. Let τ' be the strategy that plays τ for $T_{\varepsilon/2}$ and afterwards plays 2.

When σ is played against τ', either we reach $\{0,1\}$ and Adam plays 1 only finitely many times, while in s (the latter because there are only finitely many numbers below $T_{\varepsilon/2}$). Thus, for Eve to win a play, the play needs to reach vertex 1. There are two ways to do so, either Eve stops before $T_{\varepsilon/2}$ or after. In the former case, the pr. to reach 1 is only $\varepsilon/2$ (because Adam needs to play 2 at the time, which is only done with pr. $\varepsilon/2$). The latter only happens with pr. $\varepsilon/2$ by definition of $T_{\varepsilon/2}$ (because, Adam could play 2 for an arbitrary number of steps while following τ but s would not be left anyway).

We get the following lemma.

Lemma 119 (Properties of the Big Match)**.** *No finite memory strategy can guarantee more than* 0 *in the Big Match.*

The principle of sunken cost states that, when acting rationally, one should disregard cost already paid. We will next argue that this does not apply (naively) to the Big Match. A strategy following the principle of sunken cost would not depend on past cost paid and thus, in each step T, there is a pr. p_T of stopping for Eve. Such strategies are called Markov strategies in the Big Match. Fix some Markov strategy σ for Eve. We will argue, like before, that σ cannot guarantee more than ε for any $\varepsilon > 0$. Note that Eve does not depend on the choices of Adam and thus, either she stops with pr. 1 or she does not. In the former case, Adam just plays 1 forever. When Eve stops, the vertex reached is thus -1. Alternately, if Eve does not stop with pr. 1, there must be a time T, such that she only stops with pr. ε after T (this is actually also the case even if she stops with pr. 1). Adam's strategy is then to play 1 for T steps and 2 thereafter. Observe that the pr. to reach 1 is thus at most ε, in that it must be that Eve stops after T. If she does not stop (or stops in 0), there will be only finitely many 1s.

We see the following:

Lemma 120 (Properties of the Big Match)**.** *No Markov strategy can guarantee more than* 0 *in the Big Match*

Bibliographic references

We will now give the references for this chapter, split into a few paragraphs, each corresponding to a section in the chapter.

John von Neumann's work on matrix games [vNM44] (also called normal form games), showing that they have a value and there exists optimal stationary strategies, is typically considered the founding work in game theory. Besides that paper, Dantzig [Dan65] showed the equivalence to linear programming, and thus that they can be solved in polynomial time using e.g. Khachiyan's [Kha79] work on the ellipsoid method. There are also some results on how complex the strategies for matrix games are: For any $\varepsilon > 0$, there exists an ε-optimal strategy that plays uniformly over a multi-set of actions of size $\lceil (\ln n)/\varepsilon^2 \rceil$ as shown by Lipton and Young [LY94] (this is a stronger requirement than patience). Also, as shown by Feder, Nazerzadeh and Saberi [FNS07] there exists games such that any ε-optimal strategy has support at least $\Omega(\frac{\log n}{\varepsilon^2})$ (note that if, for some x, the support is $\Omega(x)$ then patience is also $\Omega(x)$). Finally, as shown by Hansen, Ibsen-Jensen, Podolskii and Tsigaridas [HIJPT13], there is an optimal strategy in any matrix game with patience less than $(n+2)^{\frac{n+2}{2}}/2^{n+1}$ and for each k there exists games with $n = m = 2^k$ such that any optimal strategy has patience at least $n^{n/2}/2^{n(1+o(1))}$ (there are also results for m and n not equal to 2^k for some k, but not quite as tight to the upper bound).

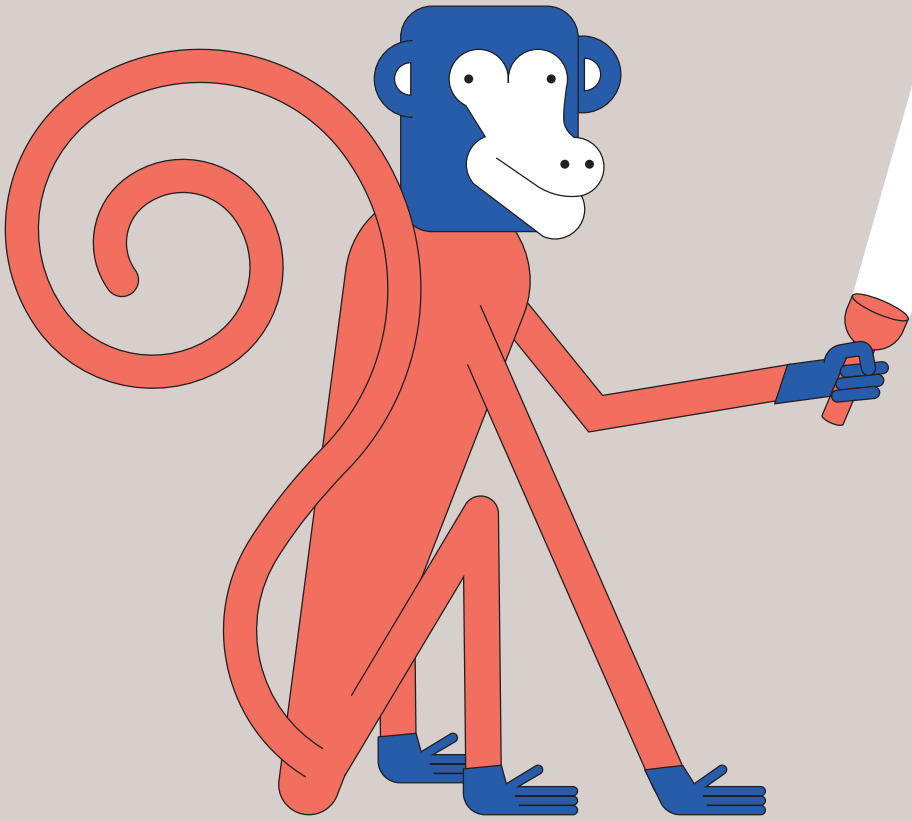
Shapley [Sha53] first considered concurrent games and focused on the class of concurrent discounted payoff games. For these, he showed that they have a value and that there are optimal stationary strategies, using in essence the proof we used for the first 3 items of Lemma 96. The proof of the fourth item, i.e. that the value can be approximated in PPAD, comes from the work of Etessami and Yannakakis [EY10]. The proof of the fifth item, an upper bound on the patience of ε-optimal strategies appears in [IJ12].

Everett [Eve57] was the first to consider concurrent reachability games (formally, he considered a slight generalization). In that paper, he showed that the games have a value and ε-optimal stationary strategies (i.e. the first part of Lemma 100 and Lemma 102). He also used the snowball game to show that Eve does not always have an optimal strategy (i.e. Lemma 101). Finally, he introduced the notion of patience for strategies. It was shown by Himmelberg, Parthasarathy, Raghavan and Vleck [Par73, HPRV76] that Adam has an optimal strategy. Frederiksen and Miltersen [FM13] showed that for each vertex x and action a (except for one action for each vertex), there is a number $c_{x,a}$ and an integer $d_{x,a}$, such that for any $\varepsilon > 0$, the strategy that plays each action a' in vertex x' with pr. $c_{x',a'}\varepsilon^{d_{x',a'}}$ is ε-optimal (the last action in each vertex is played with the remaining pr.). They used that to show that approximating the value (since the value can be irrational, it seems reasonable to approximate) can be done in TFNP[NP], slightly inside PSPACE. Finding the set of vertices of value 0, i.e. Lemma 105, is folklore. Finding the set of vertices of value 1, on the other hand, i.e. Lemma 112, is by de Alfaro, Henzinger and Kupferman [dAHK98] (their proof is different). Hansen, Koucký and Miltersen [HKM09] showed that purgatory k requires patience $\varepsilon^{2^{k-1}}$ for any $1 > \varepsilon > 0$ for Eve. Later, Chatterjee, Hansen and Ibsen-Jensen [CHIJ17] showed that purgatory duel k requires patience $(3/4)^{2^{k-1}}$

for any $0 \leq \varepsilon < 1/4$ for either player.

Gillette [Gil57] was the first to consider concurrent mean payoff games and introduced the Big Match game we use as an example. He showed that the Big Match does not have stationary strategies ensuring more than 0. Later, Blackwell and Ferguson [BF68] showed that the Big Match have a value and that value is $1/2$ by showing that some strategies that depends on the full history is ε-optimal. They also showed that no optimal Markov strategy (i.e. Lemma 120) can ensure more than 0 in that game. Next, Kohlberg [Koh74] extended this to show that all repeated games with absorbing states have a value. Finally, Mertens and Neyman [MN81] showed that all concurrent mean payoff games have a value (i.e the first part of Lemma 117). The strategies employed in all these papers kept track on the sum of over the rounds of the values of the vertex in that round minus the color in that round (the strategy by Mertens and Neyman set the memory to 0 if it should have been negative though). Finding the set of vertices where for every $\varepsilon > 0$, a finite memory strategy can ensure value $1 - \varepsilon$ was done by Chatterjee and Ibsen-Jensen [CIJ15] (the middle part of Lemma 117). Furthermore, finding the values in a game with a fixed number of vertices in polynomial time was done by Hansen, Koucký, Lauritzen, Miltersen and Tsigaridas [HKL$^+$11], informally speaking by doing binary search for the values. That finite-memory strategies cannot ensure more than 0 in the Big Match, i.e. Lemma 119 seems to be folklore. Hansen, Ibsen-Jensen and Koucký [HIJK16] considered extending Markov strategies with a finite amount of space and showed that if the memory is a deterministic function of the history, then no such strategy can ensure more than -1 in the Big Match. They also showed, for any fixed $\varepsilon > 0$, that in round T one only needs $O(\log \log T)$ bits of memory to play ε-optimal in any absorbing game. Finally, Hansen, Ibsen-Jensen and Neyman [HIJN18] showed that Markov strategies extended with a single bit of space suffice to play the Big Match ε-optimally, for any $\varepsilon > 0$ (naturally, the strategy used randomisation to update the memory state).

GAMES WITH SIGNALS



Chapter 8

Games with Signals

HUGO GIMBERT

Imperfect information. This chapter presents a few results about zero-sum games with imperfect information. Those games are a generalization of concurrent games in order to take into account the possibility that players might be imperfectly informed about the current state of the game and the actions taken by their opponent, or even their own action. We will also discuss situations where players may forget what they used to know.

Before providing formal definitions of games with imperfect information, we give several examples.

Simple poker. Our first example is a finite duration game which is a simplified version of poker, inspired by Borel and von Neumann simplified poker [FF03]. This game is played with 4 cards $\{\spadesuit, \heartsuit, \clubsuit, \diamondsuit\}$.

- The goal of Eve and Adam is to win the content of a pot in which, initially, they both put 1 euro.
- Eve receives a private random card, unknown by Adam.
- Eve decides whether to check or raise. If she checks then she wins the pot iff her card is $\spadesuit$.
- If Eve raises then Adam has two options: fold or call. If Adam folds then Eve receives the pot. If Adam raises then both player add two euros in the pot and Eve wins the pot iff her card is $\spadesuit$.

A natural strategy for Eve is to raise when she has a spade and otherwise check. Playing so, she reveals her card to Adam, and we will see that the optimal behaviour for her consists in bluffing from time to time, i.e. raise although her card is not a spade.

The distracted logician. Our second example is another finite duration game. A logician is driving home. For that he should go through two crossings, and turn left at the first one and right at the second one. This logician is very much absorbed in his thoughts, trying to prove that $P \neq NP$, and is thus pretty distracted: upon taking a decision, he cannot tell whether he already saw a crossing or not.

This simple example is useful to discuss the observability of actions and make a distinction between mixed strategies and behavioral strategy.

Network controller. The following example is inspired from collision regulation in Ethernet protocols: the controller of a network card has to share an Ethernet layer with another network card, controller by another controller, possibly malicious.

When sending a data packet, the controller selects a delay in microseconds between 1 and 512 and transmits this delay to the network card. The other controller does the same. The network cards try to send their data packet at the chosen dates. Choosing the same date results in a data collision, and the process is repeated until there is no collision, at that time the data can be sent.

The chosen delay has to be kept hidden from the opponent. This way, it can be chosen randomly, which ensures that the data will eventually be sent with probability 1, whatever does the opponent.

Guess my set. Our fourth example is an infinite duration game, parametrized by some integer n. The play is divided into three phases.

- In the first phase, Eve secretly chooses a subset $X \subsetneq \{1, \dots, 2n\}$ of size n among the $\binom{2n}{n}$ possibilities.
- In the second phase, Eve discloses to Adam $\frac{1}{2}\binom{2n}{n}$ pairwise distinct sets of size n which are all different from X.
- In the third phase, Adam aims at guessing X by trying up to $\frac{1}{2}\binom{2n}{n}$ sets of size n. If Adam succeeds in guessing X, the game restarts from the beginning. Otherwise, Eve wins.

Clearly Adam has a strategy to prevent forever Eve to win: try up one by one all those sets that were not disclosed by Eve. This strategy uses a lot of memory: Adam has to remember the whole sequence of $\frac{1}{2}\binom{2n}{n}$ sets disclosed by Eve. We will see that a variant of this game can be represented in a compact way, using a number of states polynomial in n. As a consequence, playing optimally a game with imperfect-information and infinite duration might require a memory of size doubly-exponential in the size of the game.

8.1 Notations

We consider *stochastic games with signals*, that are a standard tool in game theory to model imperfect information in stochastic games [RSV06, Ren09]. When playing a stochastic game with signals, players cannot observe the actual state of the game,

nor the actions played by themselves or their opponent: the only source of information of a player are private signals they receive throughout the play. Stochastic games with signals subsume standard stochastic games [Sha53], repeated games with incomplete information [Aum64], games with imperfect monitoring [RSV06], concurrent games [dAH00] and deterministic games with imperfect information on one side [Rei84, CDHR07].

Like in previous chapters, V, C and A denote respectively the sets of vertices, colors and actions.

Definition 24. *An imperfect information arena $\mathscr{A}$ is a tuple (S,Δ) where*

- *S is the set of* signals
- *$\Delta : V \times A \times A \to \mathscr{D}(V \times S \times S \times C)$ maps the current vertex and a pair of actions to a probability distribution over vertices, pairs of signals and colors.*

Initially, the game is in a state $v_0 \in V$ chosen according to a probability distribution $\delta_0 \in \mathscr{D}(V)$ known by both players; the initial state is v_0 with probability $\delta_0(v_0)$. At each step $n \in \mathbb{N}$, both players simultaneously choose some actions $a,b \in A$ They respectively receive signals $s,t \in S$, and the game moves to a new state v_{n+1}. This happens with probability $\Delta(v_n,a,b)(v_{n+1},c,d)$. This fixed probability is known by both players, as well as the whole description of the game.

A *play* is a sequence $(v_0,a_0,b_0,s_0,t_0,c_0),(v_1,a_1,b_1,s_1,t_1,c_1),(v_2\dots$ such that for every n, the probability $\Delta(v_n,a_n,b_n)(v_{n+1},s_n,t_n,c_n)$ is positive.

A sequence of signals for a player is *realisable* for Eve if it appears in a play, we denote $R_E \subseteq S^*$ the set of these sequence. Similarly for Adam.

An example. The simplified poker can be modelled as a stochastic game with signals. Actions of players are *public signals* sent to both players. Also their the payoff of Eve is publicly announced, when non-zero. Upon choosing whether to call or fold, Adam cannot distinguish between states ♠`Raised` and ■`Raised`, in both cases he received the sequence of signals $\circ$, `raise`. A graphical representation is provided on Figure 8.1.

The game is played with 4 cards $\{\spadesuit,\heartsuit,\clubsuit,\diamondsuit\}$. We exploit the symmetry of payoffs with respect to $\{\heartsuit,\clubsuit,\diamondsuit\}$ and identify these three colours as a single one, denoted ■, received initially by Eve with probability $\frac{3}{4}$. The set of vertices is an initial vertex `Start`, a terminal vertex `End` plus the four states

$$\{\spadesuit,\blacksquare\} \times \{\texttt{Play},\texttt{Raised}\} \ .$$

The set of colors are possible payoffs $C = \{0,-1,+1,-3,+3\}$.

The set of actions A is the union of actions of Eve $A_E = \{\cdot,\texttt{check},\texttt{raise}\}$ and actions of Adam $A_A = \{\cdot,\texttt{call},\texttt{fold}\}$.

The set of signals is $\{\circ,\spadesuit,\blacksquare\}$ plus $\{\texttt{check},\texttt{raise},\texttt{call},\texttt{fold}\} \times \{0,-1,+1,-3,+3\}$.

The rules of the game, are defined by the set of *legal* transitions. Let $c \in \{\spadesuit,\blacksquare\}$.

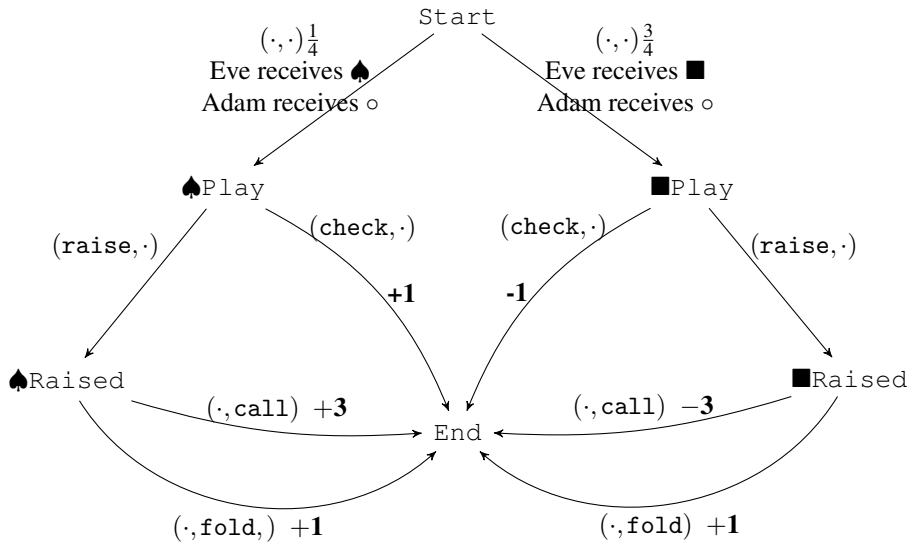


Figure 8.1: The simplified poker game.

The following transitions are legal.

$$\Delta(\texttt{Start},\cdot,\cdot)((c,\texttt{Play}),c,\circ,0) = \begin{cases} \frac{1}{4} & \text{if } c = \spadesuit \\ \frac{3}{4} & \text{if } c = \blacksquare \end{cases}.$$

$$\Delta((c,\texttt{Play}),\texttt{check},\cdot)(\texttt{End},\texttt{check}_x,\texttt{check}_x,x) = 1 \text{ where } x = \begin{cases} +1 & \text{if } c = \spadesuit \\ -1 & \text{if } c = \blacksquare. \end{cases}$$

$$\Delta((c,\texttt{Play}),\texttt{raise},\cdot)((c,\texttt{Raised}),\texttt{raise}_0,\texttt{raise}_0,0) = 1$$

$$\Delta((c,\texttt{Raised}),\cdot,\texttt{call})(\texttt{End},\texttt{call}_x,\texttt{call}_x,x) = 1 \text{ where } x = \begin{cases} +3 & \text{if } c = \spadesuit \\ -3 & \text{if } c = \blacksquare. \end{cases}$$

$$\Delta((c,\texttt{Raised}),\cdot,\texttt{fold})(\texttt{End},\texttt{fold}_1,\texttt{fold}_1,+1) = 1$$

state `End` is absorbing with payoff 0.

To simplify the notations, we assumed in the general case that players share the same set of actions and signals. As a consequence, other transitions than the legal ones are possible. One can use a threat to guarantee that Eve plays `check` and `raise` after receiving her card, by setting a heavy loss of -10 if she plays another action instead. Same thing to enforce that Adam plays `call` or `fold` after receiving the signal `raise`. When targeting applications, legal moves should be explicitly specified, typically using an automaton to compute the set of legal actions depending on the sequence of signals.

Strategies: behavioral, mixed and general. Intuitively, players make their decisions based upon the sequence of signals they receive, which is formalised with strate-

gies. There are several natural classes of strategies to play games with signals, as discussed in [CDH10b] and Section 4 in [BGG17].

A behavioural strategy of Eve associates with every realisable sequence of signals a probability distribution over actions:

$$\sigma : R_E \to \mathscr{D}(A) \ .$$

When Eve plays σ, after having received a sequence of signals $s_0, \dots, s_n$ she chooses action a with probability $\sigma(s_0, \dots, s_n)(a)$. Strategies of Adam are the same, except they are defined on R_A.

Remark that in general a player may not observe which actions he actually played, for example S might be a singleton in which case the players only knows the number of steps so far.

A game has *observable actions* if there exists a mapping $\mathrm{Act} : S \to A$ such that

$$\Delta(v,a,b)(w,s,t) > 0 \implies (a = \mathrm{Act}(s) \wedge b = \mathrm{Act}(t)) \ .$$

In [BGG17, Lemma 4.6 and 4.7] it was shown that without loss of generality, one can consider games where actions are observable and players play behavioural strategies. The discussion is technical and beyond the scope of this book.

8.2 Finite duration

We start with some results on the very interesting class of game with finite duration.

A game has *finite duration* if there is a set of absorbing vertices L, called *leaves*, such that every play eventually reaches L. In other words, the directed graph (V,E) induced by all pairs (v,w) such that $\exists a,b \in A, s,t \in S, \Delta(v,a,b)(w,s,t) > 0$ is acyclic, except for self loops on leaves.

Moreover, C is the set of real numbers, colours are called *payoffs*. At the moment the play π reaches a leaf $\ell \in L$ for the first time, the game is essentially over: Eve receives the sum of payoffs seen to far, denoted $\mathtt{pay}(\pi)$ and all future payoffs are 0. Such plays are called *terminal plays*.

Once a terminal play occurs, the game is over. For this reason, in this section we restrict realisable sequences of signals to the ones occurring in terminal plays and their prefixes. This guarantees finiteness of R_E and R_A since

$$R_E \cup R_A \subseteq S^{\leq n} \ .$$

An initial distribution δ_0 and two strategies σ and τ of Eve and Adam naturally induce a probability distribution $\mathbb{P}_{\delta_0}^{\sigma,\tau}$ on the set of terminal plays starting in one of the vertices $v_0, \delta_0(v_0) > 0$. Players have opposite interests: Eve seeks to maximize her expected payoff

$$\mathbb{E}_{\delta_0}^{\sigma,\tau} = \sum_{\text{terminal plays } \pi} \mathbb{P}_{\delta_0}^{\sigma,\tau}(\pi) \cdot \mathtt{pay}(\pi) \ ,$$

while Adam wants to minimize it.

8.2.1 Existence and computability of the value

Next theorem gathers several folklore results.

Theorem 93 (Finite duration games)**.** *A game with finite duration and imperfect information has a value: for every initial distribution δ_0,*

$$\sup_{\sigma}\inf_{\tau}\mathbb{E}_{\delta_0}^{\sigma,\tau} \;=\; \inf_{\tau}\sup_{\sigma}\mathbb{E}_{\delta_0}^{\sigma,\tau} \quad .$$

This value is denoted val(δ_0) and is computable [1]*. Both players have optimal strategies.*

Reduction to normal form. The main ingredient for proving this theorem is a transformation of the game into a matrix game called its *normal form*.

The intuition is that a player, instead of choosing progressively her actions as she receives new signals, may choose once for all at the beginning of the game how to react to every possible sequence of signals she might receive in the future.

Fix an initial distribution δ_0. In the normal form version the game, Eve picks a *deterministic* strategy $\sigma : R_E \to A$ while simultaneously Adam picks $\tau : R_A \to A$. Then the game is over and Eve receives payoff $\mathbb{E}_{\delta_0}^{\sigma,\tau}$. There are finitely many such deterministic strategies, thus the normal form game is a *matrix game*. See Section 7.2 for more details about matrix games.

An example. In the simplified poker example, the reduction is as follows.

We rely on the formal description of the game at the end of Section 8.1 and perform two simplifications. First, we only consider strategies playing moves according to the rules, other strategies are strategically useless.

Deterministic strategies of Eve are mappings $\sigma : \{\spadesuit, \blacksquare\} \to \{\texttt{check}, \texttt{raise}\}$. Adam has only two deterministic strategies: after the sequence $\circ\texttt{Raised}$, he should choose between actions `call` and `fold`.

The normal form is

	`call`	`fold`
$\spadesuit \to \texttt{check}, \blacksquare \to \texttt{check}$	-0.5	-0.5
$\spadesuit \to \texttt{raise}, \blacksquare \to \texttt{check}$	0	-0.5
$\spadesuit \to \texttt{raise}, \blacksquare \to \texttt{raise}$	-1.5	$+1$
$\spadesuit \to \texttt{check}, \blacksquare \to \texttt{raise}$	-2	$+1$

The first line corresponds to Eve never raising, thus her odds are +1 euro at 25% and -1 at 75% thus an expected payoff of -0.5. The third line corresponds to Eve always raising. If Adam calls then her odds are +3 at 25% and -3 at 75%, on average -1.5. If Adam folds, she gets payoff +1.

Remark that the rows where Eve checks with $\spadesuit$ are dominated by the corresponding row where Eve does not. Thus checking with $\spadesuit$ (slow playing) has no strategic interest,

[1] provided payoffs are presented in a way compatible with linear solvers, typically rational values.

and by elimination of weakly dominated strategies, the normal form game is equivalent to:

	`call`	`fold`
♠ → `raise`, ■ → `check`	0	−0.5
♠ → `raise`, ■ → `raise`	−1.5	1

The value of this game is $-\frac{1}{4}$. Eve has a unique optimal strategy which consists in playing the top row with probability $\frac{5}{6}$. In other words, she should bluff with probability $\frac{1}{6}$ when she receives ■. Adam has a unique optimal strategy which consists in calling or folding with equal probability $\frac{1}{2}$.

Proof of Theorem 93. The example illustrates the correspondence between behavioural strategies in the finite-duration game on one side and mixed strategies in the normal form game on the other. In the general case, the correspondence can be stated as follows.

Lemma 121. *Denote Strat the set of behavioural strategies, $Strat_d$ the subset of deterministic strategies and $\mathscr{D}(Strat_d)$ the set of strategies in the normal form game.*

1. *There is a mapping $\Phi : Strat \to \mathscr{D}(Strat_d)$ which preserves payoffs:*

$$\forall \sigma, \tau \in Strat, \mathbb{E}_{\delta_0}^{\sigma,\tau} = \sum_{\sigma',\tau' \in Strat_d} \Phi(\sigma)(\sigma') \cdot \Phi(\tau)(\tau') \cdot \mathbb{E}_{\delta_0}^{\sigma',\tau'} \quad .$$

2. *Since actions are observable, there is a mapping $\Phi' : \mathscr{D}(Strat_d) \to Strat$ which preserves payoffs:*

$$\forall \Sigma, T \in \mathscr{D}(Strat), \sum_{\sigma',\tau' \in Strat_d} \Sigma(\sigma') T(\tau') \mathbb{E}_{\delta_0}^{\sigma',\tau'} = \mathbb{E}_{\delta_0}^{\Phi'(\sigma),\Phi'(\tau)} \quad .$$

3. *$\Phi' \circ \Phi$ is the identity.*

We assumed earlier that each player can observe its own actions. This hypothesis is necessary for ii) and iii) to hold in general.

Proof. We start with i). Intuitively, all random choices of actions performed by a behavioural strategy σ of Eve can be done at the beginning of the play. Playing σ is equivalent to playing each deterministic strategy σ' with probability

$$\Phi(\sigma)(\sigma') = \Pi_{u \in R_E} \sigma(u)(\sigma'(u)) \quad .$$

We prove ii). Let $\Sigma \in \mathscr{D}(\text{Strat})$. The definition of the behavioural strategy $\sigma = \Phi'(\Sigma)$ is as follows. Let $s_0 \dots s_k$ be a finite sequence of signals. Since actions are observable, this defines unambiguously the sequence of corresponding actions $a_0 \dots a_k$ where $a_i = \text{Act}(s_i)$. We set $\sigma(s_0 \dots s_k)(a)$ to be the probability that a deterministic strategy chosen with Σ chooses action a after signals $s_0 \dots s_k$, conditioned on the fact that it has already chosen action $a_0 \dots a_k$:

$$\sigma(s_0 \dots s_k)(a) = \Sigma\left(\sigma'(s_0 \dots s_k) = a \mid \forall 0 \le i \le k, \sigma'(s_0 \dots s_{i-1}) = \text{Act}(s_i)\right) \quad ,$$

where the vertical pipe denotes a conditional probability. □

We proceed with the proof of Theorem 93. According to Theorem 92, the normal form has a value and optimal strategies for each player. Denote val_N the value and $\Sigma^\sharp$ and $T^\sharp$ the optimal strategies. Let $\sigma^\sharp = \Phi'(\Sigma^\sharp)$. Then $\sigma^\sharp$ ensures a payoff of at least val_N in the imperfect information game, because for every strategy τ,

$$\mathbb{E}_{\delta_0}^{\sigma^\sharp,\tau} = \mathbb{E}_{\delta_0}^{\Phi'(\Sigma^\sharp),\Phi'(\Phi(\tau))} = \sum_{\sigma',\tau' \in \mathrm{Strat}_d} \Sigma^\sharp(\sigma')\Phi(\tau)(\tau')\mathbb{E}_{\delta_0}^{\sigma',\tau'} \geq \mathrm{val}_N \ ,$$

where the first equalities are applications of Lemma 121 and the inequality is by optimality of $\Sigma^\sharp$. Symmetrically, $\tau^\sharp = \Phi'(T^\sharp)$ guarantees $\forall \sigma, \mathbb{E}_{\delta_0}^{\sigma,\tau^\sharp} \leq \mathrm{val}_N$. Thus the value of the game with finite duration is val_N and $\sigma^\sharp$ and $\tau^\sharp$ are optimal. □

8.2.2 The Koller-Meggido-von Stengel reduction to linear programming

The reduction of a finite-duration game with imperfect information to its normal form proves that the value exists and is computable. However the corresponding algorithm is computationally very expensive, it requires solving a linear program of size roughly doubly-exponential in the size of the original game, since the normal form is a matrix index by $A^{R_E} \times A^{R_A}$ and the set of signal sequences might contain all sequences of S of length $\leq n$.

Koller, Meggido and von Stengel did provide a more efficient direct reduction to linear programming. Strategies of Eve in the normal form game live in $\mathsf{R}^{A^{R_E}}$ while her strategies in the game with imperfect information live in a space with exponentially fewer dimensions, namely $\mathsf{R}^{R_E \times A}$. The direct reduction avoids this dimensional blowup.

Theorem 94. *The value of a game with imperfect information can be computed by a linear program with $|R_E| + |R_A|$ variables.*

As a consequence, in the particular case where the game graph is a tree then $|R_E| \leq n$ and $|R_A| \leq n$ and the value can be computed in polynomial time, like stated in [KMvS94].

Proof. The construction of the linear program relies on three key ideas.

First, representing a behavioral strategy $\sigma : R_E \to \mathscr{D}(A)$ of Eve as a *plan* $\pi : R_E \to [0,1]$ recursively defined by $\pi(\varepsilon) = 1$ and for every $s_0 \cdots s_n \in R_E, s \in S$,

$$\pi(s_0 \cdots s_n \cdot s) = \pi(s_0 \cdots s_n) \cdot \sigma(s_0 \cdots s_n)(\mathrm{Act}(s)) \ .$$

Remind that actions are observable and $\mathrm{Act}(s)$ denotes the action that Eve has just played before receiving signal s. In the linear program, plans are represented by variables $(p_r)_{r \in R_E}$. Valuations corresponding to plans can be characterized by the following equalities. First, $p_\varepsilon = 1$. Second, for every $s_0 \ldots s_{n-1} s, s_0 \ldots s_{n-1} s' \in R_E$,

$$(\mathrm{Act}(s) = \mathrm{Act}(s')) \implies \left(p_{s_0 \ldots s_{n-1} s} = p_{s_0 \ldots s_{n-1} s'}\right) \ .$$

We denote $p_{s_0\dots s_{n-1}a}$ the common value of all $p_{s_0\dots s_{n-1}s}$ with $a = \mathrm{Act}(s)$. The third equality is $p_{s_0\dots s_{n-1}} = \sum_{a\in A} p_{s_0\dots s_{n-1}a}$.

The second key idea is to introduce variables evaluating the contribution of a (realisable) sequence of signals of Adam to the total expected payoff Eve. These contributions are represented by variables $(v_r)_{r\in R_A}$.

The third key idea is to aggregate the product of transition probabilities along a play. For every play $(v_0,a_0,b_0,s_0,t_0,c_0),\dots,(v_k,a_k,b_k,s_k,t_k,c_k)$ we denote $\mathbb{E}(\pi)$ the product of all transition probabilities of π and $r_E(\pi)$ the sequence of signals of Eve in this play:

$$\mathbb{E}(\pi) = \delta_0(v_0)\cdot\Delta(v_0,a_0,b_0,s_0,t_0,c_0)\cdots\Delta(v_k,a_k,b_k,s_k,t_k,c_k)$$
$$r_E(\pi) = s_0,s_1,\dots,s_k \ .$$

We show that the following linear program with variables $(p_r)_{r\in R_E}$, $(v_r)_{r\in R_A}$ has an optimal solution which equals to $\mathrm{val}(\delta_0)$. For every sequences of signals $r\in R_A$ we denote $T_A(r)$ the (possibly empty) set of terminal plays whose sequence of signals for Adam is r.

Maximise v_ε subject to

$(p_r)_{r\in R_E}$ is a plan of Eve

$$\forall r\in R_A, \forall a\in A,\quad v_r \le \sum_{\substack{rs\in R_A\\ s\in S,\mathrm{Act}(s)=a}} v_{rs} + \sum_{\pi\in T(r)} \mathbb{E}(\pi)\cdot \mathtt{pay}(\pi)\cdot p_{r_E(\pi)} \qquad (8.1)$$

For our purpose, it is enough to establish that the optimal solution of the LP is

$$\mathrm{val}(\delta_0) = \sup_\sigma \min_{\tau \text{ deterministic}} \mathbb{E}_{\delta_0}^{\sigma,\tau} \ .$$

The reason is that in a matrix game, for every fixed strategy of Eve, Adam can minimize the payoff by playing a single action with probability 1. Thus, according to the reduction to normal form seen in the previous chapter, for every strategy σ of Eve, there is a *deterministic* strategy τ of Adam which minimizes $\mathbb{E}_{\delta_0}^{\sigma,\tau}$.

We show first that for every feasible solution $(p_r)_{r\in R_E}$, $(v_r)_{r\in R_A}$ of the linear program, the strategy σ corresponding to the plan $(p_r)_{r\in R_E}$ guarantees that for every *deterministic* strategy τ, $\mathbb{E}_{\delta_0}^{\sigma,\tau} \ge v_\varepsilon$. Since τ is deterministic then $\mathbb{E}_{\delta_0}^{\sigma,\tau}$ is the sum of all $\mathbb{E}(\pi)\cdot\mathtt{pay}(\pi)\cdot p_{r_E(\pi)}$ over plays π played according to τ thus a trivial induction shows $\mathbb{E}_{\delta_0}^{\sigma,\tau} \ge v_\varepsilon$.

We show now that to every strategy σ of Eve, and to every deterministic optimal answer τ of Adam, corresponds a feasible solution of the program such that $v_\varepsilon = \mathbb{E}_{\delta_0}^{\sigma,\tau}$. Let $(p_r)_{r\in R_E}$ the plan corresponding to σ. For every $r\in R_A$ define v_r be the expected payoff of Eve in an auxiliary game where she plays σ and Adam plays τ and the payoff

of Eve is turned to 0 whenever Adam signals sequence does not start with r. We show that the linear constraint Equation (8.1) holds for every $r \in R_A$ and action a. Since τ is deterministic then Equation (8.1) is an equality whenever $a = \tau(r)$. And since τ is an optimal answer to σ, it is locally optimal in the sense where playing an action different from $\tau(r)$ after r cannot be profitable to Adam, hence Equation (8.1) holds. Finally, $(p_r)_{r\in R_E}$, $(v_r)_{r\in R_A}$ is a feasible solution. □

An example. The following linear program computes the value of the simplified poker example.

Maximise v_ε subject to

$$\begin{aligned}
\forall r \in R_E,\ 0 \le p_r \le 1 \\
p_{\spadesuit,\texttt{check}} + p_{\spadesuit,\texttt{raise}} &= 1 \\
p_{\blacksquare,\texttt{check}} + p_{\blacksquare,\texttt{raise}} &= 1 \\
v_\varepsilon \le v_\circ &\le v_{\circ,\texttt{check}} + v_{\circ,\texttt{raise}} \\
v_{\circ,\texttt{check}} &\le \frac{1}{4}\cdot p_{\spadesuit,\texttt{check}} \cdot (+1) + \frac{3}{4}\cdot p_{\blacksquare,\texttt{check}} \cdot (-1) \\
v_{\circ,\texttt{raise}} &\le \frac{1}{4}\cdot p_{\spadesuit,\texttt{raise}} \cdot (+1) + \frac{3}{4}\cdot p_{\blacksquare,\texttt{raise}} \cdot (+1) \\
v_{\circ,\texttt{raise}} &\le \frac{1}{4}\cdot p_{\spadesuit,\texttt{raise}} \cdot (+3) + \frac{3}{4}\cdot p_{\blacksquare,\texttt{raise}} \cdot (-3)
\end{aligned}$$

Setting $x = p_{\spadesuit,\texttt{check}}$ and $y = p_{\blacksquare,\texttt{check}}$, the solution is

$$\begin{aligned}
&\frac{1}{4}\max_{(x,y)\in[0,1]^2} (x - 3y + \min((1-x)+3(1-y), 3(1-x) - 9(1-y))) \\
=&\frac{1}{4}\max_{(x,y)\in[0,1]^2} \min(4-6y, -6-2x+6y) = \frac{1}{4}\max_{y\in[0,1]} \min(4-6y, -6+6y) \quad ,
\end{aligned}$$

which is maximal when $y = \frac{5}{6}$ and the solution is $-\frac{1}{4}$.

Nose scratch variant. Assume now that Eve does not have the perfect poker face: whenever she has ♠ she scratches her nose with probability $\frac{1}{2}$ whereas in general it happens only with probability $\frac{1}{6}$. Only Adam is aware of this sign, which he receives as a private signal s (scratch) or n (no scratch).

Compared to the perfect poker face situation, the situation is slightly better for Adam: the value drops from $-\frac{1}{4}$ to $(-\frac{1}{4} - \frac{1}{10})$. The optimal bluff frequency of Eve decreases from $\frac{1}{6}$ to $\frac{1}{10}$. Computation details follow.

Maximise v_ε subject to

$$
\begin{aligned}
&\forall u \in R_E,\ 0 \le p_u \le 1 \\
&\qquad p_{\spadesuit,\mathtt{c}} + p_{\spadesuit,\mathtt{r}} = 1 \quad p_{\blacksquare,\mathtt{c}} + p_{\blacksquare,\mathtt{r}} = 1 \\
&\qquad v_\varepsilon \le v_{\mathtt{s}} + v_{\mathtt{n}} \quad v_{\mathtt{s}} \le v_{\mathtt{sc}} + v_{\mathtt{sr}} \quad v_{\mathtt{n}} \le v_{\mathtt{nc}} + v_{\mathtt{nr}} \\
&\qquad v_{\mathtt{sc}} \le \frac{1}{4}\cdot\frac{1}{2}\cdot p_{\spadesuit,\mathtt{c}}\cdot(+1) + \frac{3}{4}\cdot\frac{1}{6}\cdot p_{\blacksquare,\mathtt{c}}\cdot(-1) \\
&\qquad v_{\mathtt{nc}} \le \frac{1}{4}\cdot\frac{1}{2}\cdot p_{\spadesuit,\mathtt{c}}\cdot(+1) + \frac{3}{4}\cdot\frac{5}{6}\cdot p_{\blacksquare,\mathtt{c}}\cdot(-1) \\
&\qquad v_{\mathtt{sr}} \le \frac{1}{4}\cdot\frac{1}{2}\cdot p_{\spadesuit,\mathtt{r}}\cdot(+1) + \frac{3}{4}\cdot\frac{1}{6}\cdot p_{\blacksquare,\mathtt{r}}\cdot(+1) \\
&\qquad v_{\mathtt{sr}} \le \frac{1}{4}\cdot\frac{1}{2}\cdot p_{\spadesuit,\mathtt{r}}\cdot(+3) + \frac{3}{4}\cdot\frac{1}{6}\cdot p_{\blacksquare,\mathtt{r}}\cdot(-3) \\
&\qquad v_{\mathtt{nr}} \le \frac{1}{4}\cdot\frac{1}{2}\cdot p_{\spadesuit,\mathtt{r}}\cdot(+1) + \frac{3}{4}\cdot\frac{5}{6}\cdot p_{\blacksquare,\mathtt{r}}\cdot(+1) \\
&\qquad v_{\mathtt{nr}} \le \frac{1}{4}\cdot\frac{1}{2}\cdot p_{\spadesuit,\mathtt{r}}\cdot(+3) + \frac{3}{4}\cdot\frac{5}{6}\cdot p_{\blacksquare,\mathtt{r}}\cdot(-3)
\end{aligned}
$$

Set $y = p_{\blacksquare,\mathtt{c}}$. Some elementary simplifications lead to the equivalent program:

$$\max_{0\le y\le 1} \frac{1}{8}\left(\min\left(8-12y, -10+8y, 6-8y, -12+12y\right)\right)$$

The optimum is reached when $8y-10 = 8-12y$ i.e. when $p_{\blacksquare,\mathtt{c}} = \frac{9}{10}$ and is equal to $-\frac{7}{20} = -\frac{1}{4} - \frac{1}{10}$.

8.3 Infinite duration

Games with infinite duration and imperfect information are a natural model for applications such as synthesis of controllers of embedded systems. This is illustrated by the example of the network controller. Whereas in the previous section games of finite-duration were equipped with real-valued payoffs, here we focus on Büchi conditions.

8.3.1 Playing games with infinite duration and imperfect information

Notations used for games of finite duration are kept. On top of that we need to define how probabilities are measured and the winning conditions.

Measuring probabilities. The choice of an initial distribution $\delta_0 \in \mathscr{D}(V)$ and two strategies $\sigma : R_E \to \mathscr{D}(A)$ and $\tau : R_A \to \mathscr{D}(A)$ for Eve and Adam defines a Markov chain on the set of all finite plays. This in turn defines a probability measure $\mathbb{P}_{\delta_0}^{\sigma,\tau}$ on the Borel-measurable subsets of Δ^ω. The random variables V_n, A_n, B_n, S_n and T_n denote respectively the n-th state, action of Eve, action of Adam, signal received by Eve and Adam, and we denote π_n the finite play $\pi_n = V_0, A_0, B_0, S_0, T_0, V_1, \ldots, S_n, T_n, V_{n+1}$.

The probability measure $\mathbb{P}_{\delta_0}^{\sigma,\tau}$ is the only probability measure over Δ^ω such that for every $v \in V$, $\mathbb{P}_{\delta_0}^{\sigma,\tau}(V_0 = v) = \delta_0(v)$ and for every $n \in \mathbb{N}$,

$$\mathbb{P}_{\delta_0}^{\sigma,\tau}(V_{n+1}, S_n, T_n \mid \pi_n) \\ = \sigma(S_0 \cdots S_{n-1})(A_n) \cdot \tau(T_0 \cdots T_{n-1})(B_n) \cdot \Delta(V_n, A_n, B_n)(V_{n+1}, S_n, T_n) \ ,$$

where we use standard notations for conditional probability measures.

Winning conditions. The set of colours is $C = \{0,1\}$. The reachability, safety, Büchi and coBüchi condition condition are defined as follows:

$$\begin{aligned} \texttt{Reach} &= \{\exists n \in \mathbb{N}, C_n = 1\} \\ \texttt{Safety} &= \{\forall n \in \mathbb{N}, C_n = 0\} \\ \texttt{Buchi} &= \{\forall m \in \mathbb{N}, \exists n \geq m, C_n = 1\} \\ \texttt{CoBuchi} &= \{\exists m \in \mathbb{N}, \forall n \geq m, C_n = 0\} \ . \end{aligned}$$

When the winning condition is `Win`, Eve and Adam use strategies σ and τ and the initial distribution is δ_0, then Eve wins the game with probability:

$$\mathbb{P}_{\delta_0}^{\sigma,\tau}(\texttt{Win}) \ .$$

Eve wants to maximise this probability, while Adam wants to minimise it.

8.3.2 The value problem.

The value problem is computationally intractable for games with infinite duration and imperfect information. This holds even for the very simple case of blind one-player games with reachability conditions. Those are games where the set of signals is a singleton and actions of Adam have no influence on the transition probabilities. These games can be seen as probabilistic automata, hence the undecidability result of Paz applies.

Theorem 95 (Undecidability for blind one-player games)**.** *Whether Eve has a strategy to win with probability $\geq \frac{1}{2}$ is undecidable, even in blind one-player games.*

Actually, the value might not even exist.

Proposition 2 (The value may not be defined)**.** *There is a game with infinite duration imperfect information and Büchi condition in which*

$$\sup_\sigma \inf_\tau \mathbb{P}_{\delta_0}^{\sigma,\tau}(\texttt{Buchi}) = \frac{1}{2} < 1 = \inf_\tau \sup_\sigma \mathbb{P}_{\delta_0}^{\sigma,\tau}(\texttt{Buchi}) \ .$$

The value however exists for games with reachability condition. Although the value problem is not decidable, there are some other interesting decision problems to consider.

8.3.3 Winning with probability 1 or > 0

Winning almost-surely or positively. A strategy σ for Eve is *almost-surely winning* from an initial distribution δ_0 if

$$\forall \tau, \mathbb{P}_{\delta_0}^{\sigma,\tau}(\texttt{Win}) = 1 \ .$$

When such an almost-surely strategy σ exists, the initial distribution δ_0 is said to be almost-surely winning (for Eve).

A less enjoyable situation for Eve is when she only has a positively winning strategy. A strategy σ for Eve is *positively winning* from an initial distribution δ_0 if

$$\forall \tau, \mathbb{P}_{\delta_0}^{\sigma,\tau}(\texttt{Win}) > 0 \ .$$

When such a strategy σ exists, the initial distribution δ is said to be positively winning (for Eve). Symmetrically, a strategy τ for Adam is positively winning if it guarantees $\forall \sigma, \mathbb{P}_{\delta_0}^{\sigma,\tau}(\texttt{Win}) < 1$.

The worst situation for Eve is when her opponent has an almost-surely winning strategy τ, which thus ensures $\mathbb{P}_{\delta_0}^{\sigma,\tau}(\texttt{Win}) = 0$ whatever strategy σ is chosen by Eve.

Qualitative determinacy.

Theorem 96 (Qualitative determinacy)**.** *Stochastic games with signals and reachability, safety and Büchi winning conditions are qualitatively determined: either Eve wins almost-surely winning or Adam wins positively. Formally, in those games,*

$$\left(\forall \tau, \exists \sigma, \mathbb{P}_{\delta_0}^{\sigma,\tau}(\texttt{Win}) = 1\right) \implies \left(\exists \sigma, \forall \tau, \mathbb{P}_{\delta_0}^{\sigma,\tau}(\texttt{Win}) = 1\right) \ .$$

The proof of this result is given in the next section.

Since reachability and safety games are dual, a consequence of Theorem 96, is that in a reachability game, every initial distribution is either almost-surely winning for Eve, almost-surely winning for Adam, or positively winning for both players. When a safety condition is satisfied almost-surely for a fixed profile of strategies, it trivially implies that the safety condition is satisfied by all consistent plays, thus for safety games winning *surely* is the same than winning almost-surely.

By contrast, co-Büchi games are *not* qualitatively determined:

Lemma 122. *There is a co-Büchi game in which neither Eve has an almost-surely winning strategy nor Adam has a positively winning strategy.*

Proof. In this game, Eve observes everything, Adam is blind (he only observes his own actions), and Eve's objective is to visit only finitely many times the ☹-state. The initial state is ☹. The set of actions is $\{a,b,c,d\}$. All transitions are deterministic.

On one hand, no strategy Σ is almost-surely winning for Eve for her co-Büchi objective. Since both players can observe their actions, it is enough to prove that no behavioral strategy $\sigma \in C^* \to \Delta(I)$ of Eve is almost-surely winning. Fix strategy σ and assume towards contradiction that σ is almost-surely winning. We define

a strategy τ such that $\mathbb{P}^{\sigma,\tau}_{☹}(\texttt{Buchi}) > 0$. Strategy τ starts by playing only c. The probability to be in state ☹ at step n is $x^0_n = \mathbb{P}^{\sigma,c^\omega}_{☹}(V_n = ☹)$ and since σ is almost-surely winning then $x^0_n \to_n 0$ thus there exists n_0 such that $x^0_{n_0} \leq \frac{1}{2}$. Then τ plays d at step n_0. Assuming the state was 2 when d was played, the probability to be in state ☹ at step $n \geq n_0$ is $x^1_n = \mathbb{P}^{\sigma,c^{n_0}dc^\omega}_{☹}\left(V_n = ☹ \mid V_{n_0} = ☹\right)$ and since σ is almost-surely winning there exists n_1 such that $x^1_{n_1} \leq \frac{1}{4}$. Then τ plays d at step n_1. By induction we keep defining τ this way so that $\tau = c^{n_0-1}dc^{n_1-n_0-1}dc^{n_2-n_1-1}d\cdots$. and for every $k \in \mathbb{N}$, $\mathbb{P}^{\sigma,\tau}_{☹}\left(V_{n_{k+1}} = ☹ \text{ and } V_{n_{k+1}-1} = 2 \mid V_{n_k} = ☹\right) \geq 1 - \frac{1}{2^{k+1}}$. Thus finally $\mathbb{P}^{\sigma,\tau}_{☹}(\texttt{Buchi}) \geq \Pi_k(1 - \frac{1}{2^{k+1}}) > 0$ which contradicts the hypothesis.

On the other hand, Adam does not have a positively winning strategy either. Intuitively, Adam cannot win positively because as time passes, either the play reaches state 1 or the chances that Adam plays action d drop to 0. When these chances are small, Eve can play action c and she bets no more d will be played and the play will stay safe in state 2. If Eve loses her bet then again she waits until the chances to see another d are small and then plays action c. Eve may lose a couple of bets but almost-surely she eventually is right and the `CoBuchi` condition is almost-surely fulfilled.

Finally neither Eve wins almost-surely nor Adam wins positively. □

Decidability

Theorem 97. *Deciding whether the initial distribution of a Büchi games, is almost-surely winning for Eve is* 2EXP*-complete. For safety games, the same problem is* EXP*-complete.*

Concerning winning positively a *safety or co-Büchi game*, one can use Theorem 96 and the determinacy property: Adam has a positively winning strategy in the above game if and only if Eve has no almost-surely winning strategy. Therefore, deciding when Adam has a positively winning strategy can also be done, with the same complexity.

Theorem 98. *For reachability and Büchi games where either Eve is perfectly informed about the state or Adam is better informed than Eve, deciding whether the initial distribution is almost-surely winning for Eve is* EXP*-complete. In safety games Eve is perfectly informed about the state, the decision problem is in* P.

8.3.4 Qualitative determinacy: proof of Theorem 96

Beliefs. The *belief* of a player is the set of possible states of the game, according to the signals received by the player.

Definition 25 (Belief)**.** *Let $\mathscr{A}$ be an arena with observable actions. From an initial set of states $L \subseteq V$, the belief of Eve after having received signal s is:*

$$\mathscr{B}_{Eve}(L,s) = \{v \in V \mid \exists l \in L, t \in S \text{ such that } \Delta(l,s,t)(v,Act(s),Act(t)) > 0\} \ .$$

Remark that in this definition we use the fact that actions of Eve are observable, thus when he receives a signal $s \in C$ Eve can deduce he played action $a_1(c) \in I$. The

belief of Eve after having received a sequence of signals $s_1, \ldots, s_n$ is defined inductively by:

$$\mathscr{B}_{Eve}(L, s_1, s_2, \ldots, s_n) = \mathscr{B}_{Eve}(\mathscr{B}_{Eve}(L, s_1, \ldots, s_{n-1}), s_n).$$

Beliefs of Adam are defined similarly. Given an initial distribution δ, we denote $\mathscr{B}^n_{Eve}$ the random variable defined by

$$\mathscr{B}^0_{Eve} = supp(\delta)$$
$$\mathscr{B}^{n+1}_{Eve} = \mathscr{B}_{Eve}(supp(\delta), C_1, \ldots, C_{n+1}) = \mathscr{B}_{Eve}(\mathscr{B}^n_{Eve}, C_{n+1}) \ .$$

We will also rely on the notion of *belief of belief*, called here *2-belief*, which, roughly speaking, represents for one player the set of possible beliefs for his (or her) adversary, as well as the possible current state.

Definition 26 (2-Belief)**.** *Let $\mathscr{A}$ be an arena with observable actions. From an initial set $\mathscr{L} \subseteq V \times \mathscr{P}(V)$ of pairs composed of a state and a belief for Adam, the 2-belief of Eve after having received signal c is the subset of $V \times \mathscr{P}(V)$ defined by:*

$$\mathscr{B}^{(2)}_{Eve}(\mathscr{L}, s) = \{(v, \mathscr{B}_{Adam}(L,t)) \mid \exists (\ell, L) \in \mathscr{L}, t \in S, \Delta(v,s,t)(\ell, Act(s), Act(t)) > 0\} \ .$$

From an initial set $\mathscr{L} \subseteq V \times \mathscr{P}(V)$ of pairs composed of a state and a belief for Adam, the 2-belief of Eve after having received a sequence of signals $s_1, \ldots, s_n$ is defined inductively by:

$$\mathscr{B}^{(2)}_{Eve}(\mathscr{L}, s_1, s_2, \ldots, s_n) = \mathscr{B}^{(2)}_{Eve}\left(\mathscr{B}^{(2)}_{Eve}(\mathscr{L}, s_1, \ldots, s_{n-1}), s_n\right) \ .$$

There are natural definitions of 3-beliefs (beliefs on beliefs on beliefs) and even k-beliefs however for our purpose, 2-beliefs are enough, in the following sense: in Büchi games the positively winning sets of Adam can be characterised by fixed point equations on sets of 2-beliefs, and some positively winning strategies of Adam with finite-memory can be implemented using 2-beliefs.

Supports positively winning supports. Note that whether an initial distribution δ_0 is almost-surely or positively winning depends only on its support, because $\mathbb{P}^{\sigma,\tau}_{\delta_0}(\texttt{Win}) = \sum_{v \in V} \delta_0(v) \cdot \mathbb{P}^{\sigma,\tau}_{\delta_0}(\texttt{Win} \mid V_0 = v)$. As a consequence, we will say that a support $L \subseteq V$ is almost-surely or positively winning for a player if there exists a distribution with support L which has the same property.

In the sequel, we will denote $\mathscr{L}_{\text{Eve},=1}$ the set of supports almost-surely winning for Eve and $\mathscr{L}_{\text{Adam},>0}$ those positively winning for Adam.

Then the qualitative determinacy theorem is a corollary of the following lemma.

Lemma 123. *In every Büchi game, every non-empty support which does not belong to $\mathscr{L}_{Adam,>0}$ belongs to $\mathscr{L}_{Eve,=1}$.*

The proof of this lemma relies on the definition of a strategy called the maximal strategy. We prove that this strategy is almost-surely winning from any initial distribution which is not positively winning for Adam.

Definition 27 (Maximal strategy)**.** *For every non-empty support $L \subseteq V$ we define the set of L-safe actions for Eve as*

$$\texttt{ISafe}(L) = \left\{a \in A \mid \forall s \in S, (Act(s) = a) \implies (\mathscr{B}_{Eve}(L,s) \notin \mathscr{L}_{Adam,>0})\right\} \ ,$$

in other words these are the actions which Eve can play without taking the risk that her belief is positively winning for Adam.

The maximal strategy *is the strategy of Eve which plays the uniform distribution on* $\texttt{ISafe}(\mathscr{B}_{Eve})$ *when it is not empty and plays the uniform distribution on A otherwise. It is denoted* σ_{max}*.*

To play her maximal strategy at step n, Eve only needs to keep track of her belief $\mathscr{B}^n_{\text{Eve}}$, thus $\sigma_{\max}$ can be implemented by Eve using a finite-memory device which keeps track of the current belief. Such a strategy is said to be *belief-based*. We will use several times the following technical lemma about belief-based strategies.

Lemma 124. *Fix a Büchi game. Let $\mathscr{L} \subseteq \mathscr{P}(V)$ and σ a strategy for player* 1*. Assume that σ is a belief strategy, $\mathscr{L}$ is downward-closed (i.e. $L \in \mathscr{L} \wedge L' \subseteq L \implies L' \in \mathscr{L}$) and for every $L \in \mathscr{L} \setminus \{\emptyset\}$ and every strategy τ,*

$$\mathbb{P}^{\sigma,\tau}_{\delta_L}(\texttt{Reach}) > 0 \ , \tag{8.2}$$

$$\mathbb{P}^{\sigma,\tau}_{\delta_L}(\forall n \in \mathbb{N}, \mathscr{B}^n_{Eve} \in \mathscr{L}) = 1 \ . \tag{8.3}$$

Then σ is almost-surely winning for the Büchi game from any support $L \in \mathscr{L} \setminus \{\emptyset\}$.

Proof. Since $\mathscr{L}$ is downward-closed then $\forall L \in \mathscr{L}, \forall l \in L, \{l\} \in \mathscr{L}$ thus Section 8.3.3 implies

$$\forall L \in \mathscr{L}, \forall l \in L, \mathbb{P}^{\sigma,\tau}_{\delta_L}(\texttt{Reach} \mid V_0 = l) > 0 \ . \tag{8.4}$$

Once σ is fixed then the game is a one-player game with state space $V \times 2^V$ and imperfect information and Equation (8.4) implies that in this one-player game,

$$\forall L \in \mathscr{L}, \forall l \in L, \forall \tau, \mathbb{P}^{\tau}_{\delta_L}(\texttt{Reach} \mid V_0 = l) > \varepsilon \ , \tag{8.5}$$

where $N = |K| \cdot 2^{|K|}$ and $\varepsilon = p_{\min}^{|K| \cdot 2^{|K|}}$ and $p_{\min}$ is the minimal non-zero transition probability. Moreover Equation (8.3) implies that in this one-player game the second component of the state space is always in $\mathscr{L}$, whatever strategy τ is played by player 2. Remind the definition

$$\texttt{Reach} = \{\exists n \in \mathbb{N}, C_n = 1\} \ .$$

As a consequence, in this one-player game for every $m \in \mathbb{N}$, and every behavioral strategy τ and every $l \in V$,

$$\mathbb{P}^{\tau}_{\delta_L}(\exists m \leq n \leq m+N, C_n = 1 \mid K_m = l) \geq \varepsilon, \tag{8.6}$$

whenever $\mathbb{P}^{\tau}_{\delta_L}(V_m = l) > 0$.

We use the Borel-Cantelli Lemma to conclude the proof. According to Equation (8.6), for every τ, $L \in \overline{\mathscr{L}}$, $m \in \mathbb{N}$,

$$\mathbb{P}^{\tau}_{\delta_L}(\exists n, mN \leq n < (m+1)N, C_n = 1 \mid V_{mN}) \geq \varepsilon, \tag{8.7}$$

which implies for every behavioral strategy τ and $k,m \in \mathbb{N}$,

$$\mathbb{P}^{\tau}_{\delta_L}(\forall n, ((m \cdot N) \leq n < ((m+k) \cdot N) \implies C_n \neq 1)) \leq (1-\varepsilon)^k \ .$$

Since $\sum_k (1-\varepsilon)^k$ is finite, we can apply Borel-Cantelli Lemma for the events $(\{\forall n, m \cdot N \leq n < (m+k) \cdot N \implies C_n \neq 1\})_k$ and we get $\mathbb{P}^{\tau}_{\delta_L}(\forall n, m \cdot N \leq n \implies C_n \neq 1) = 0$ thus

$$\mathbb{P}^{\tau}_{\delta_L}(\texttt{Buchi}) = 1 \ . \tag{8.8}$$

As a consequence σ is almost-surely winning for the Büchi game. □

An important feature of the maximal strategy is the following.

Lemma 125. *In a Büchi game with observable actions, let $\delta \in \Delta(K)$ be an initial distribution which is not positively winning for Adam, i.e. $supp(\delta) \notin \mathscr{L}_{Adam,>0}$. Then for every strategy τ of Adam*

$$\mathbb{P}^{\sigma_{max},\tau}_{\delta}(\forall n \in \mathbb{N}, \mathscr{B}^n_{Eve} \notin \mathscr{L}_{Adam,>0}) = 1 \ . \tag{8.9}$$

Proof. We only provide a sketch of proof. The proof is an induction based on the fact that for every non-empty subset $L \subseteq V$,

$$(L \notin \mathscr{L}_{\text{Adam},>0}) \implies (\texttt{ISafe}(L) \neq \emptyset) \ .$$

Assume a contrario that $\texttt{ISafe}(L) = \emptyset$ for some $L \notin \mathscr{L}_{\text{Adam},>0}$. Then for every action $a \in A$ there exists a signal $s_a \in S$ such that $\mathscr{B}_{\text{Eve}}(L, s_a) \neq \emptyset$ and $\mathscr{B}_{\text{Eve}}(L, s_a) \in \mathscr{L}_{\text{Adam},>0}$. Since $\mathscr{B}_{\text{Eve}}(L, s_a) \neq \emptyset$, the definition of the belief operator implies:

$$\exists v_a \in L, w_a \in V, t_a \in T, \text{ such that } \Delta(w_a, s_a, t_a)(v_a, \text{Act}(s), \text{Act}(t_a)) > 0 \ .$$

But then Adam can win positively from L with the following strategy. At the first round, Adam plays randomly any action in A. At the next round, Adam picks up randomly a belief in $\mathscr{L}_{\text{Adam},>0}$ and plays forever the corresponding positively winning strategy. Remark that this strategy of Adam is not described as a behavioural strategy but rather as a finite-memory strategy. Since actions are observable, such a finite-memory strategy can be turned into a behavioural one, see [BGG17, Lemma 4.6 and 4.7].

Why is Adam strategy positively winning from L? Whatever action $a \in A$ is played by Eve, with positive probability she will receive signal s_a, because Adam might play the action $\text{Act}(t_a)$. Since $\mathscr{B}_{\text{Eve}}(L, s_a) \in \mathscr{L}_{\text{Adam},>0}$ then there Adam might with positive probability play a strategy positively winning when the initial belief of Eve is $\mathscr{B}_{\text{Eve}}(L, s_a)$. Thus whatever action Eve chooses, she might lose with positive probability.

□

The notion of maximal strategy being defined, we can complete the proof of Theorem 96. For that, we show that σ_{max} is almost-surely winning from every support not in $\mathscr{L}_{\mathrm{Adam},>0}$.

Reachability and safety conditions can be easily encoded as Büchi conditions, thus it is enough to consider Büchi games.

The first step is to prove that for every $L \in \mathscr{L}_{\mathrm{Eve},=1}$, for every strategy τ of Adam,

$$\mathbb{P}_{\delta_L}^{\sigma_{\mathsf{max}},\tau}(\texttt{Safety}) < 1 \ . \tag{8.10}$$

We prove Equation (8.10) by contradiction. Assume Equation (8.10) does not hold for some $L \in \mathscr{L}_{\mathrm{Eve},=1}$ and strategy τ:

$$\mathbb{P}_{\delta_L}^{\sigma_{\mathsf{max}},\tau}(\texttt{Safety}) = 1 \ . \tag{8.11}$$

Under this assumption we use τ to build a strategy positively winning from L, which will contradict the hypothesis $L \in \mathscr{L}_{\mathrm{Adam},>0}$. Although τ is surely winning from L against the particular strategy σ_{max}, there is no reason for τ to be positively winning from L against all other strategies of player 1.

However we can rely on τ in order to define another strategy τ' for Adam positively winning from L. The strategy τ' is a strategy which gives positive probability to play τ all along the play, as well as any strategy in the family of strategies $(\tau_{n,B})_{n\in\mathbb{N},B\in\mathscr{L}_{\mathrm{Adam},>0}}$ defined as follows. For every $B \in \mathscr{L}_{\mathrm{Adam},>0}$ we choose a strategy τ_B positively winning from B. Then $\tau_{n,B}$ is the strategy which plays the uniform distribution on A for the first n steps then forgets past signals and switches definitively to τ_B.

A possible way to implement the strategy τ' is as follows. At the beginning of the play player 2 tosses a fair coin. If the result is head then he plays τ. Otherwise he keeps tossing coins and as long as the coin toss is head, player 2 plays randomly an action in J . The day the coin toss is tail, he picks up randomly some $B \in \mathscr{L}_{\mathrm{Adam},>0}$ and starts playing τ_B.

Remark that this strategy of Adam is not described as a behavioural strategy but, since actions are observable, such a finite-memory strategy can be turned into a behavioural one, see [BGG17, Lemma 4.6 and 4.7].

Now that τ' is defined, we prove it is positively winning from L. Let E be the event 'player 1 plays only actions that are safe with respect to her belief', *i.e.*

$$E = \{\forall n \in \mathbb{N}, A_n \in \texttt{ISafe}_{\mathscr{L}}(\mathscr{B}^n_{\mathrm{Eve}})\} \ .$$

Then for every behavioral strategy σ:

- Either $\mathbb{P}_{\delta_L}^{\sigma,\tau'}(E) = 1$. In this case

$$\mathbb{P}_{\delta_L}^{\sigma,\tau'}(\texttt{Safety}) > 0 \ ,$$

because for every finite play $\pi = v_0 a_0 b_0 s_1 t_1 v_1 \cdots v_n$,

$$\left(\mathbb{P}_{\delta_L}^{\sigma,\tau'}(\pi) > 0\right) \implies \left(\mathbb{P}_{\delta_L}^{\sigma_{\mathsf{max}},\tau'}(\pi) > 0\right) \implies \texttt{Safety} \ ,$$

where the first implication holds because, by definition of σ_{max} and E, for every $s_1 \cdots s_n \in CS^*, \mathrm{supp}(\sigma(s_1 \cdots s_n)) \subseteq \mathrm{supp}(\sigma_{\mathsf{max}}(s_1 \cdots s_n))$ while the second implication is from Equation (8.11). Thus $\mathbb{P}^{\sigma,\tau}_{\delta_L}(\mathtt{Safety}) = 1$ and we get $\mathbb{P}^{\sigma,\tau'}_{\delta_L}(\mathtt{Safety}) > 0$ by definition of τ'.

- Or $\mathbb{P}^{\sigma,\tau'}_{\delta_L}(E) < 1$. Then by definition of E there exists $n \in \mathbb{N}$ such that

$$\mathbb{P}^{\sigma,\tau'}_{\delta_L}(A_n \notin \mathtt{ISafe}_{\mathscr{L}}(\mathscr{B}^n_{\mathrm{Eve}})) > 0 \ .$$

By definition of $\mathtt{ISafe}_{\mathscr{L}}$ it implies $\mathbb{P}^{\sigma,\tau'}_{\delta_L}\left(\mathscr{B}^{n+1}_{\mathrm{Eve}} \in \mathscr{L}\right) > 0$, thus there exists $B \in \mathscr{L}$ such that $\mathbb{P}^{\sigma,\tau'}_{\delta_L}\left(\mathscr{B}^{n+1}_{\mathrm{Eve}} = B\right) > 0$. By definition of τ' we get $\mathbb{P}^{\sigma,\tau_{n+1,B}}_{\delta_L}\left(\mathscr{B}^{n+1}_{\mathrm{Eve}} = B\right) > 0$, because whatever finite play $v_0, \ldots, v_{n+1}$ leads with positive probability to the event $\{\mathscr{B}^{n+1}_{\mathrm{Eve}} = B\}$, the same finite play can occur with $\tau_{n+1,B}$ since $\tau_{n+1,B}$ plays every possible action for the $n+1$ first steps. Since $\tau_{n+1,B}$ coincides with $\tau_{\mathtt{rand}}$ for the first $n+1$ steps then by definition of beliefs, $\mathbb{P}^{\sigma,\tau_{n+1,B}}_{\delta_L}\left(\mathscr{B}^{n+1}_{\mathrm{Eve}} = B\right) > 0$ and $B \subseteq \{k \in K \mid \mathbb{P}^{\sigma,\tau_{n+1,B}}_{\delta_L}\left(K_{n+1} = k \mid \mathscr{B}^{n+1}_{\mathrm{Eve}} = B\right) > 0\}$. Using the definition of τ_B we get $\mathbb{P}^{\sigma,\tau_{n+1,B}}_{\delta_L}(\mathtt{CoBuchi}) > 0$.

As a consequence by definition of τ' we get $\mathbb{P}^{\sigma,\tau'}_{\delta_L}(\mathtt{CoBuchi}) > 0$.

In both cases, for every σ, $\mathbb{P}^{\sigma,\tau'}_{\delta_L}(\mathtt{CoBuchi}) > 0$ thus τ' is positively winning from L. This contradicts the hypothesis $L \in \mathscr{L}_{\mathrm{Eve},=1}$. As a consequence we get Equation (8.10) by contradiction.

Using Equation (8.10), we apply Lemma 124 to the collection $\overline{\mathscr{L}_{\mathrm{Adam},>0}}$ and the strategy σ_{max}. The collection $\overline{\mathscr{L}_{\mathrm{Adam},>0}}$ is downward-closed because $\mathscr{L}_{\mathrm{Adam},>0}$ is upward-closed: if a support is positively winning for Adam then any greater support is positively winning as well, using the same positively winning strategy.

Thus σ_{max} is almost-surely winning for the Büchi game from every support in $\overline{\mathscr{L}_{\mathrm{Adam},>0}}$ i.e. every support which is not positively winning for Adam, hence the game is qualitatively determined.

8.3.5 Decidability: proof of Theorem 97 and Theorem 98

A naïve algorithm

As a corollary of the proof of qualitative determinacy (Theorem 96), we get a maximal strategy σ_{max} for player 1 (see Definition 27) to win almost-surely Büchi games.

Corollary 19. *If player* 1 *has an almost-surely winning strategy in a Büchi game with observable actions then the maximal strategy σ_{max} is almost-surely winning.*

A simple algorithm to decide for which player a game is winning can be derived from Corollary 19: this simple algorithm enumerates all possible belief strategies and test each one of them to see if it is almost-surely winning. The test reduces to checking positive winning in one-player co-Büchi games and can be done in exponential time.

As there is a doubly exponential number of belief strategies, this can be done in time doubly exponential. This algorithm also appears in [GS09b]. This settles the upper bound for Theorem 97.

Matching complexity lower bounds are established in [BGG17], proving that this enumeration algorithm is optimal for worst case complexity. While optimal in the worst case, this algorithm is likely to be inefficient in practice. For instance, if player 1 has no almost-surely winning strategy, then this algorithm will enumerate every single of the doubly exponential many possible belief strategies. Instead, we provide fixed point algorithms which do not enumerate every possible strategy in Theorem 99 for reachability games and Theorem 100 for Büchi games. Although they should perform better on games with particular structures, these fixed point algorithms still have a worst-case 2-EXPcomplexity.

A fixed point algorithm for reachability games

We turn now to the (fixed points) algorithms which compute the set of supports that are almost-surely or positively winning for various objectives.

Theorem 99 (Deciding positive winning in reachability games)**.** *In a reachability game each initial distribution δ is either positively winning for player* 1 *or surely winning for player* 2*, and this depends only on supp$(\delta) \subseteq V$. The corresponding partition of $\mathscr{P}(V)$ is computable in time $\mathscr{O}\left(|G| \cdot 2^{|V|}\right)$, where $|G|$ denotes the size of the description of the game, as the largest fixed point of a monotonic operator $\Phi : \mathscr{P}(\mathscr{P}(V)) \to \mathscr{P}(\mathscr{P}(V))$ computable in time linear in $|G|$.*

We denote TT the set of vertices whose colour is 1.

Proof. Let $\mathscr{L}_\infty \subseteq \mathscr{P}(V \setminus TT)$ be the greatest fixed point of the monotonic operator $\Phi : \mathscr{P}(\mathscr{P}(V \setminus TT)) \to \mathscr{P}(\mathscr{P}(V \setminus TT))$ defined by:

$$\Phi(\mathscr{L}) = \{L \in \mathscr{L} \mid \exists j_L \in J, \forall d \in T, (a_2(d) = j_L) \implies (\mathscr{B}_{\text{Adam}}(L,d) \in \mathscr{L} \cup \{\emptyset)\}\} \ , \tag{8.12}$$

in other words $\Phi(\mathscr{L})$ is the set of supports such that player 2 has an action which ensure his next belief will be in $\mathscr{L}$, whatever signal d he might receive. Let $\sigma_{\texttt{rand}}$ be the strategy for player 1 that plays randomly any action.

We are going to prove that:

1. every support in $\mathscr{L}_\infty$ is surely winning for player 2,
2. and $\sigma_{\texttt{rand}}$ is positively winning from any support $L \subseteq V$ which is not in $\mathscr{L}_\infty$.

We start with proving the first item. To win surely from any support $L \in \mathscr{L}_\infty$, player 2 uses the following belief strategy τ_B: when the current belief of player 2 is $L \in \mathscr{L}_\infty$ then player 2 plays an action j_L defined as in Equation (8.12). By definition of Φ and since $\mathscr{L}_\infty$ is a fixed point of Φ, there always exists such an action. When playing with the belief strategy τ_B, starting from a support in $\mathscr{L}_\infty$, the beliefs of player 2 stay in $\mathscr{L}_\infty$ and never intersect TT because $\mathscr{L}_\infty \subseteq \mathscr{P}(V \setminus TT)$. According to Equation (8.3) of beliefs (Lemma 124), this guarantees the play never visits TT, whatever strategy is used by player 1.

We now prove the second item. Let $\mathscr{L}_0 = \mathscr{P}(V \setminus TT) \supseteq \mathscr{L}_1 = \Phi(\mathscr{L}_0) \supseteq \mathscr{L}_2 = \Phi(\mathscr{L}_1) \dots$ and $\mathscr{L}_\infty$ be the limit of this sequence, the greatest fixed point of Φ. We prove that for any support $L \in \mathscr{P}(V)$, if $L \notin \mathscr{L}_\infty$ then:

$$\sigma_{\mathtt{rand}} \text{ is positively winning for player 1 from } L \ . \tag{8.13}$$

If $L \cap TT \neq \emptyset$, Equation (8.13) is obvious. To deal with the case where $L \cap TT = \emptyset$, we define for every $n \in \mathbb{N}$, $\mathscr{K}_n = \mathscr{P}(V \setminus TT) \setminus \mathscr{L}_n$, and we prove by induction on $n \in \mathbb{N}$ that for every $L \in \mathscr{K}_n$, for every initial distribution δ_L with support L, for every behavioral strategy τ,

$$\mathbb{P}_{\delta_L}^{\sigma_{\mathtt{rand}},\tau}(\exists m, 2 \leq m \leq n+1, V_m \in TT) > 0 \ . \tag{8.14}$$

For $n = 0$, Equation (8.14) is obvious because $\mathscr{K}_0 = \emptyset$. Suppose that for some $n \in \mathbb{N}$, Equation (8.14) holds for every $L' \in \mathscr{K}_n$, and let $L \in \mathscr{K}_{n+1} \setminus \mathscr{K}_n$. Then by definition of $\mathscr{K}_{n+1}$,

$$L \in \mathscr{L}_n \setminus \Phi(\mathscr{L}_n) \ . \tag{8.15}$$

Let δ_L be an initial distribution with support L and τ any behavioral strategy for player 2. Let $J_0 \subseteq J$ be the support of $\tau(\delta_L)$ and $j_L \in J_0$. According to Equation (8.15), by definition of Φ, there exists a signal $d \in D$ such that $a_2(d) = j_L$ and $\mathscr{B}_{\mathrm{Adam}}(L,d) \notin \mathscr{L}_n$ and $\mathscr{B}_{\mathrm{Adam}}(L,d) \neq \emptyset$. According to Equation (8.3) of beliefs (Lemma 124), $\forall k \in \mathscr{B}_{\mathrm{Adam}}(L,d), \mathbb{P}_{\delta_L}^{\sigma_{\mathtt{rand}},\tau}(V_2 = k \wedge D_1 = d) > 0$. If $\mathscr{B}_{\mathrm{Adam}}(L,d) \cap TT \neq \emptyset$ then according to the definition of beliefs, $\mathbb{P}_{\delta_L}^{\sigma_{\mathtt{rand}},\tau}(V_2 \in TT) > 0$. Otherwise $\mathscr{B}_{\mathrm{Adam}}(L,d) \in \mathscr{P}(V \setminus TT) \setminus \mathscr{L}_n = \mathscr{K}_n$ hence distribution $\delta_d : k \to \mathbb{P}_{\delta_L}^{\sigma_{\mathtt{rand}},\tau}(V_2 = k \mid D_1 = d)$ has its support in $\mathscr{K}_n$. By inductive hypothesis, for every behavioral strategy τ',

$$\mathbb{P}_{\delta_d}^{\sigma_{\mathtt{rand}},\tau'}(\exists m \in \mathbb{N}, 2 \leq m \leq n+1, V_m \in TT) > 0$$

hence using the shifting lemma and the definition of δ_d,

$$\mathbb{P}_{\delta}^{\sigma_{\mathtt{rand}},\tau}(\exists m \in \mathbb{N}, 3 \leq m \leq n+2, V_m \in TT) > 0 \ ,$$

which completes the proof of the inductive step.

Hence Equation (8.14) holds for every behavioural strategy τ. Thus Equation (8.14) holds as well for every general strategy τ.

To compute the partition of supports between those positively winning for player 1 and those surely winning for player 2, it is enough to compute the largest fixed point of Φ. Since Φ is monotonic, and each application of the operator can be computed in time linear in the size of the game ($|G|$) and the number of supports ($2^{|V|}$) the overall computation can be achieved in time $|G| \cdot 2^{|V|}$. To compute the strategy τ_B, it is enough to compute for each $L \in \mathscr{L}_\infty$ one action j_L such that $(a_2(d) = j_L) \implies (\mathscr{B}_{\mathrm{Adam}}(L,d) \in \mathscr{L}_\infty)$. □

As a byproduct of the proof one obtains the following bounds on time and probabilities before reaching a target state, when player 1 uses the uniform memoryless strategy $\sigma_{\mathtt{rand}}$. From an initial distribution positively winning for the reachability objective, for every strategy τ,

$$\mathbb{P}_{\delta}^{\sigma_{\mathtt{rand}},\tau}\left(\exists n \leq 2^{|V|}, C_n = 1\right) \geq \left(\frac{1}{p_{\min} \mid A \mid}\right)^{2^{|V|}} \ , \tag{8.16}$$

where $p_{\min}$ is the smallest non-zero transition probability.

A fixed point algorithm for Büchi games

To decide whether player 1 wins almost-surely a Büchi game, we provide an algorithm which runs in doubly-exponential time. It uses the algorithm for reachability games as a sub-procedure.

Theorem 100 (Deciding almost-sure winning in Büchi games)**.** *In a Büchi game each initial distribution δ is either almost-surely winning for player* 1 *or positively winning for player* 2*, and this depends only on supp$(\delta) \subseteq V$. The corresponding partition of $\mathscr{P}(V)$ is computable in time $\mathscr{O}\left(2^{2^{|G|}}\right)$, where $|G|$ denotes the size of the description of the game, as a projection of the greatest fixed point $\mathscr{L}_\infty$ of a monotonic operator*

$$\Psi : \mathscr{P}(\mathscr{P}(V) \times V) \to \mathscr{P}(\mathscr{P}(V) \times V) \ .$$

The operator Ψ is computable using as a nested fixed point the operator Φ of Theorem 99. The almost-surely winning belief strategy of player 1 *and the positively winning* 2*-belief strategy of player* 2 *can be extracted from $\mathscr{L}_\infty$.*

We sketch the main ideas of the proof of Theorem 100.

First, suppose that from *every* initial support, player 1 can win positively the reachability game. Then she can do so using a belief strategy and according to Lemma 124, this strategy guarantees almost-surely the Büchi condition.

In general though player 1 is not in such an easy situation and there exists a support L which is *not* positively winning for her for the reachability objective. Then by qualitative determinacy, player 2 has a strategy to achieve surely her safety objective from L, which is *a fortiori* surely winning for her co-Büchi objective as well.

We prove that in case player 2 can *force with positive probability the belief of player* 1 to be L eventually from another support L', then player 2 has a general strategy to win positively from L'. This is not completely obvious because in general player 2 cannot know exactly *when* the belief of player 1 is L (he can only compute the 2-Belief, letting him know all the possible beliefs player 1 can have). However player 2 can make blind guesses, and be right with > 0 probability. For winning positively from L', player 2 plays totally randomly until he guesses randomly that the belief of player 1 is L, at that moment he switches to a strategy surely winning from L. Such a strategy is far from being optimal, because player 2 plays randomly and in most cases he makes a wrong guess about the belief of player 1. However there is a non zero probability for his guess to be right.

Hence, player 1 should surely avoid her belief to be L or L' if she wants to win almost-surely. However, doing so player 1 may prevent the play from reaching target states, which may create another positively winning support for player 2, and so on. This is the basis of our fixed point algorithm.

Using these ideas, we prove that the set $\mathscr{L}_\infty \subseteq \mathscr{P}(V)$ of supports almost-surely winning for player 1 for the Büchi objective is the largest set of initial supports from

which:

player 1 has a strategy which win positively the reachability game
and also ensures at the same time her belief to stay in $\mathscr{L}_\infty$. (†)

Property Equation (†) can be reformulated as a reachability condition in a new game whose states are states of the original game augmented with beliefs of player 1, kept hidden to player 2.

The fixed point characterisation suggests the following algorithm for computing the set of supports positively winning for player 2: $\mathscr{P}(V) \setminus \mathscr{L}_\infty$ is the limit of the sequence $\emptyset = \mathscr{L}'_0 \subsetneq \mathscr{L}'_0 \cup \mathscr{L}''_1 \subsetneq \mathscr{L}'_0 \cup \mathscr{L}'_1 \subsetneq \mathscr{L}'_0 \cup \mathscr{L}'_1 \cup \mathscr{L}''_2 \subsetneq \ldots \subsetneq \mathscr{L}'_0 \cup \cdots \cup \mathscr{L}'_m = \mathscr{P}(V) \setminus \mathscr{L}_\infty$, where

- from supports in $\mathscr{L}''_{i+1}$ player 2 can surely guarantee the safety objective, under the hypothesis that player 1 guarantees for sure her beliefs to stay outside $\mathscr{L}'_i$,
- from supports in $\mathscr{L}'_{i+1}$ player 2 can ensure with positive probability the belief of player 1 to be in $\mathscr{L}''_{i+1}$ eventually, under the same hypothesis.

The overall strategy of player 2 positively winning for the co-Büchi objective consists in playing randomly for some time until he decides to pick up randomly a belief L of player 1 in some $\mathscr{L}''_i$, bets that the current belief of player 1 is L and that player 1 guarantees for sure her future beliefs will stay outside $\mathscr{L}'_i$. He forgets the signals he has received up to that moment and switches definitively to a strategy which guarantees the first item. With positive probability, player 2 guesses correctly the belief of player 1 at the right moment, and future beliefs of player 1 will stay in $\mathscr{L}'_i$, in which case the co-Büchi condition holds and player 2 wins.

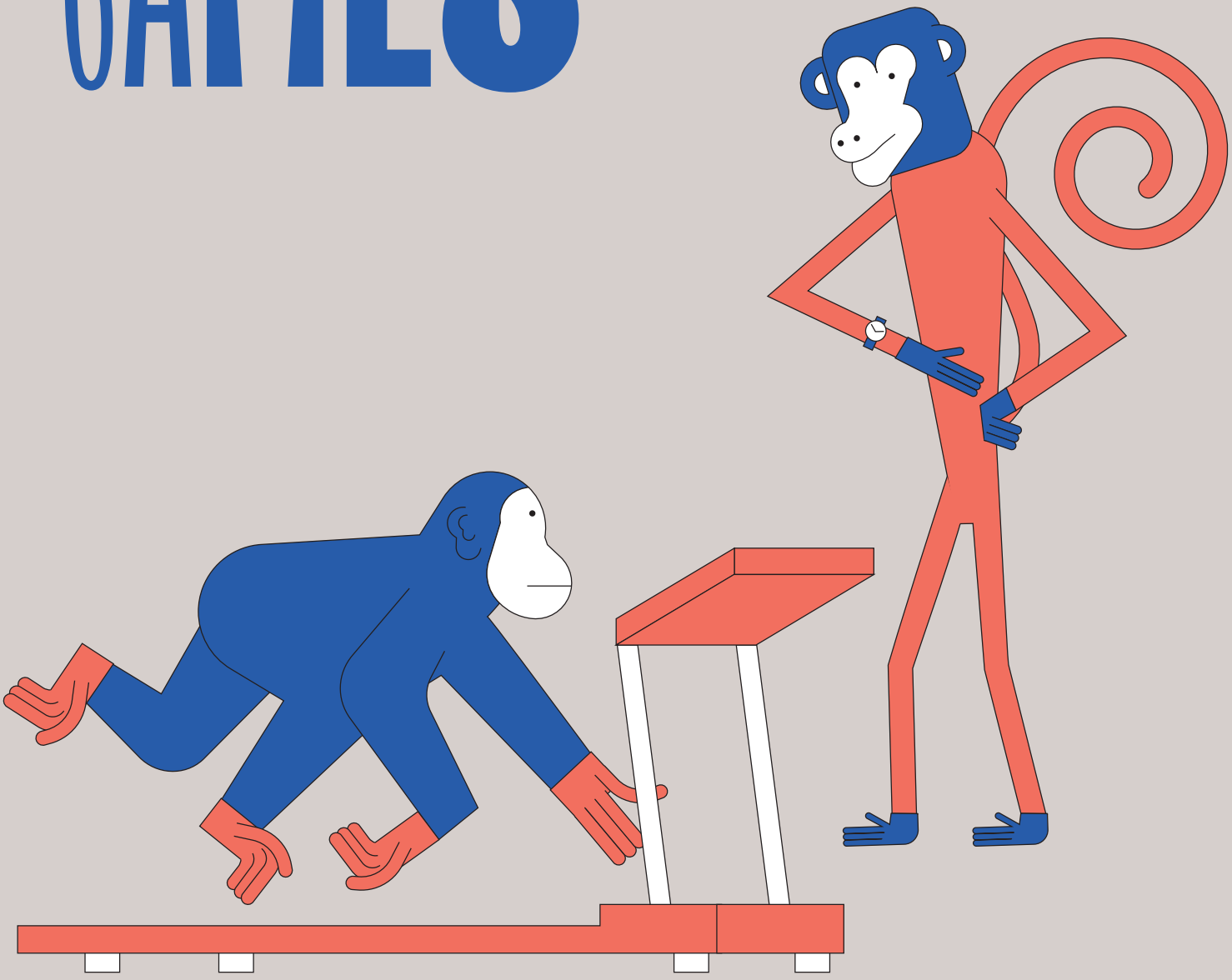
In order to ensure the first item, player 2 makes use of the hypothesis about player 1 beliefs staying outside $\mathscr{L}'_i$. For that player 2 needs to keep track of all the possible beliefs of player 1, hence the doubly-exponential memory. The reason is player 2 can infer from this data structure some information about the possible actions played by player 1: in case for every possible belief of player 1 an action $i \in I$ creates a risk to reach $\mathscr{L}'_i$ then player 2 knows for sure this action is not played by player 1. This in turn helps player 2 to know which are the possible states of the game. Finally, when player 2 estimates the state of the game using his 2-beliefs, this gives a potentially more accurate estimation of the possible states than simply computing his 1-beliefs.

The positively winning 2-belief strategy of player 2 has a particular structure. All memory updates are deterministic except for one: from the initial memory state $\emptyset$, whatever signal is received there is non-zero chance that the memory state stays $\emptyset$ but it may as well be updated to several other memory states.

Part IV

Infinite

TIMED GAMES



Chapter 9

Timed Games

NICOLAS MARKEY, OCAN SANKUR

The ability to model real-time constraints is crucial when automata and games are used for verification and synthesis. Timed automata [AD94] are a model of choice for reasoning about real-time systems: they extend finite-state automata with a finite number of *clocks*, which are real-valued variables all growing at the same rate, used to measure and constrain the elapse of time between various transitions in the automaton. Because these clocks can take arbitrary non-negative values, the set of configurations of a timed automaton is infinite. Still, reachability (and many other problems) remain decidable in timed automata. The interested reader can find more background in [AD94], but we will try to keep our presentation self-contained.

In this chapter, we consider game models based on timed automata; we call them *timed games* throughout this chapter. In timed games, besides choosing which transitions should be played, the players also decide how much time will elapse before each transition. The elapsed time is determined using clocks, and the edges have guards which determine clock values for which the edge can be taken.

Example 1. *Figure 9.1 contains a timed game with clock x, where Eve's objective is to reach the vertex G. We will define these arenas as concurrent arenas: dashed edges belong to Adam, and plain edges to Eve. Both players can take any edge at any time as long as the guard is satisfied. For instance, Eve's edge from* ℓ_1 *to* ℓ_2 *is only available if clock x has value at most* 1*, while Adam's edge from* ℓ_1 *to* ℓ_3 *is available if x is less than* 1.

In the next section, we define timed games and their semantics formally. Then we introduce some classical tools needed to reason about the space of clock valuations, and finally present an efficient algorithm for deciding the winner in timed games with reachability objectives.

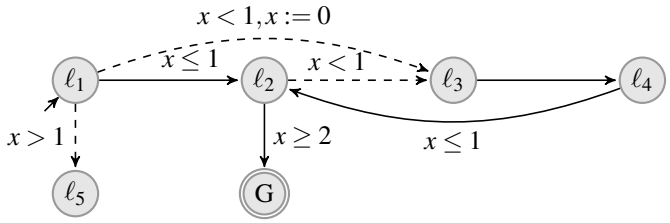


Figure 9.1: Timed game $\mathscr{A}_1$.

9.1 Notations

We fix a finite set $\mathscr{C}$ of *clock* variables to be used in our timed games. Elements of $\mathbb{R}_{\geq 0}^{\mathscr{C}}$, which assign a value to each clock, are called *valuations*.

Clocks will be used to define clock constraints, which in turn are used in timed automata to restrict the set of allowed behaviours: edges are decorated with clock constraints defining conditions for their availability. An *atomic clock constraint* is a formula of the form $k \preceq x \preceq' l$ or $k \preceq x - y \preceq' l$ where $x, y \in \mathscr{C}$, $k, l \in \mathbb{Z} \cup \{-\infty, \infty\}$ and $\preceq, \preceq' \in \{<, \leq\}$. A *clock constraint* is a conjunction of atomic clock constraints. A valuation $\nu \colon \mathscr{C} \to \mathbb{R}_{\geq 0}$ satisfies a clock constraint g, denoted $\nu \models g$, if all atomic clock constraints are satisfied when each $x \in \mathscr{C}$ is replaced with its value $\nu(x)$. We write $\Phi_{\mathscr{C}}$ for the set of clock constraints built on the clock set $\mathscr{C}$.

For a subset $R \subseteq \mathscr{C}$ and a valuation ν, we denote with $\nu[R \leftarrow 0]$ the valuation defined by $\nu[R \leftarrow 0](x) = 0$ if $x \in R$ and $\nu[R \leftarrow 0](x) = \nu(x)$ otherwise. Given $d \in \mathbb{R}_{\geq 0}$ and a valuation ν, the valuation $\nu + d$ is defined by $(\nu + d)(x) = \nu(x) + d$ for all $x \in \mathscr{C}$. We extend these operations to sets of valuations in the obvious way.

We now formally define *timed games*, which are finite representations that define infinite-state, non-stochastic concurrent games.

Definition 28. *A timed arena $\mathscr{T}$ is a tuple $(\mathscr{L}, \mathscr{C}, E_{Eve}, E_{Adam}, c)$, where $\mathscr{L}$ is a finite set of locations, $\mathscr{C}$ is a finite set of clocks, $E_{Eve}, E_{Adam} \subseteq \mathscr{L} \times \Phi_{\mathscr{C}} \times 2^{\mathscr{C}} \times \mathscr{L}$ are the sets of edges respectively controlled by Eve and Adam, and $c \colon E_{Eve} \cup E_{Adam} \to C$ is the coloring function. A timed game is a pair $(\mathscr{T}, \Omega)$ where $\Omega \subseteq C^{\omega}$ a qualitative objective.*

A configuration of such a timed automaton is a pair (ℓ, ν) where $\ell \in \mathscr{L}$ and $\nu \colon \mathscr{C} \to \mathbb{R}_{\geq 0}$. The set of configurations is the set of vertices of the infinite-state game defined by $\mathscr{T}$.

The actions of both players are pairs (d, e) where $d \in \mathbb{R}_{\geq 0}$ is a delay they want to wait before playing their controlled action e. Action (d, e) is available for Eve (resp. Adam) in configuration (ℓ, ν) if $e \in E_{\text{Eve}}(\ell)$ (resp. $e \in E_{\text{Adam}}(\ell)$) and, writing $e = (\ell, g, R, \ell')$, if additionally $\nu + d \models g$; in other terms, the clock constraint (called guard hereafter) on e must hold true under the clock valuation reached after elapsing d time units. We add an extra dummy action $\perp$ for the case where some player has no available action (this action is only available if no other actions are).

Once both players have selected an action available from configuration (ℓ, ν), the

action (d,e) with smallest delay is performed (by breaking ties in favor of Adam), leading to configuration $(\ell', (v+d)[R \leftarrow 0])$: this corresponds to letting d time units elapse, thereby reaching configuration $(\ell, v+d)$, and to following edge e (which by construction is available from $(\ell, v+d)$). We define $\mathsf{step}((\ell,v),(d,e))$ for the configuration $(\ell', (v+d)[R \leftarrow 0])$ reached from (ℓ,v) by applying action (d,e).

This definition captures the concurrent nature of the interaction between a controller (Eve) and its environment (Adam) in a real-time system, since none of the players knows in advance how long the opponent will want to wait before performing her transition. The semantics of a timed arena $(\mathscr{L}, \mathscr{C}, E_{\text{Eve}}, E_{\text{Adam}})$ can then formally be defined in terms of a concurrent arena (following the definition of Section 8.1). The underlying graph (V,E) is such that $V = \mathscr{L} \times \mathbb{R}^{\mathscr{C}}_{\geq 0}$, and $E = V \times C \times V$; the set of actions of Eve is $\mathbb{R}_{\geq 0} \times E_{\text{Eve}}$, and that of Adam is $\mathbb{R}_{\geq 0} \times E_{\text{Adam}}$; finally, the transition function, which is not stochastic in our case, maps any configuration (ℓ, v) and pair of actions $(d_{\text{Eve}}, e_{\text{Eve}})$ and $(d_{\text{Adam}}, e_{\text{Adam}})$ to the edge $((\ell,v), \gamma, \mathsf{step}((\ell,v),(d,e)))$, where $(d,e) = (d_{\text{Eve}}, e_{\text{Eve}})$ and $\gamma = c(e_{\text{Eve}})$ if $d_{\text{Eve}} < d_{\text{Adam}}$, and $(d,a) = (d_{\text{Adam}}, e_{\text{Adam}})$ and $\gamma = c(e_{\text{Adam}})$ otherwise.

A path in a timed arena $\mathscr{T}$ then is a path in its associated infinite-state concurrent arena. The qualitative objective Ω can then be evaluated along runs of a timed arena in the natural way.

Contrary to Chapter 7, in this chapter we only consider deterministic strategies[1]. As a result, timed games are not determined, as illustrated in the following example.

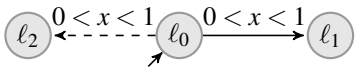


Figure 9.2: Timed arena $\mathscr{A}_2$. Solid arrow is Eve's, dashed one is Adam's.

Example 2 (Timed Games are not determined)**.** *In the timed arena $\mathscr{A}_2$ defined in Figure 9.2, from configuration $(\ell_0, \vec{0})$, Eve does not have a winning strategy to reach location ℓ_1, but Adam does not have a winning strategy either to avoid ℓ_1. In fact, available moves for both players consist in $(d, (\ell_0, \ell_1))$ with $0 < d < 1$ for Eve, and $(d', (\ell_0, \ell_2))$ with $0 < d' < 1$ for Adam. Thus, for any particular delay $0 < d < 1$ chosen by Eve, Adam has a possible delay $d < d' < 1$ which leads to ℓ_2, which is losing for Eve. This shows that Eve does not have a winning strategy. The argument is however symmetric, and Adam also does not have a winning strategy to avoid ℓ_1. Timed games are thus non determined.*

Example 3 (Winning strategy on running example)**.** *Let us consider again the example of Figure 9.1 and see whether Eve has a winning strategy from the initial configuration. At the initial configuration $\ell_1, x = 0$, Eve needs to make a move towards ℓ_2 while $x \leq 1$ since whenever $x > 1$, Adam can move to ℓ_5 which guarantees Eve's lost. Assume the game proceeds to ℓ_2 with any value $x \leq 1$. Here, Eve can try to wait until $x \geq 2$ and go to G. However, if $x < 1$, then Adam can move to ℓ_3. From this location, Eve can guarantee to come back to ℓ_2 with $x = 1$, and then move to G and win the game. Assume*

[1] Adding randomization over the infinite sets of actions is beyond the scope of this chapter.

now that from ℓ_1, the game proceeds to ℓ_3 with $x = 0$ due to Adam's move. Again, Eve can wait for 1 time unit, and go back to ℓ_2 with $x = 1$, and win the game. Hence, Eve has a winning strategy from ℓ_1 for all $x \leq 1$, and from ℓ_2 for all values of x.

In this chapter, the main problem we are interested in is determining whether Eve has a strategy for her reachability objective. Let $\vec{0}$ denote the clock valuation assigning 0 to all clocks.

Problem 9 (Solving a timed reachability game)**.**

INPUT: *A timed arena $\mathscr{T}$, initial location ℓ_0, and a reachability objective* $\texttt{Reach}(Win)$

OUTPUT: *Does Eve has a winning strategy in* $(\mathscr{T}, \texttt{Reach}(Win))$ *from configuration* $(\ell_0, \vec{0})$.

The difficulty of this problem is that the concurrent game $((V,E),\Delta,\Omega)$ has an infinite state-space, and players have infinitely many actions. We thus start by studying a data structure to represent sets of states and operations to compute successors and predecessors on these sets. We then give two algorithms to solve the above problem. We also show how such a strategy for Eve can be computed and finitely represented.

9.2 State-Space Representation

We introduce a data structure to represent sets of clock valuations and manipulate them efficiently in order to compute successors and predecessors in a given timed game. This will allow us to use a fixed point characterization of the winning states analogous to that in finite games as in Chapter 2.

A zone is any subset of $\mathbb{R}^{\mathscr{C}}_{\geq 0}$ that can be defined using a clock constraint (hence a zone is convex). We will see that sets of states that appear when exploring the state space of a timed game can be represented using zones. We use the *difference-bound matrices* to represent zones: this is one of the main data structures used in timed-automata verification [Dil90, BM83]. The idea is to store, in a matrix, upper bounds on clocks and on differences of pairs of clocks. Formally, given a clock set $\mathscr{C} = \{x_1, \ldots, x_m\}$, we define $\mathscr{C}_0 = \mathscr{C} \cup \{x_0\}$ where x_0 is seen as a special clock which is always 0. A *difference-bound matrix* (*DBM*) is a $|\mathscr{C}_0| \times |\mathscr{C}_0|$ matrix with coefficients in $\{\leq, <\} \times \mathbb{Z}$. For any DBM M, the (i,j)-component of the matrix M will be written $(\prec^M_{i,j}, M_{i,j})$ where $\prec^M_{i,j}$ is the inequality in $\{\leq, <\}$, and $M_{i,j}$ the integer coefficient. A DBM M defines the zone

$$[M] = \left\{ v \in \mathbb{R}^{\mathscr{C}}_{\geq 0} \;\middle|\; \bigwedge_{0 \leq i,j \leq |\mathscr{C}_0|} v(x_i) - v(x_j) \prec^M_{i,j} M_{i,j} \right\},$$

where $v(x_0) = 0$.

Example 4 (An example of a DBM)**.** *] Consider the clock set $\mathscr{C} = \{x_1, x_2\}$ and the zone Z defined by $x_1 \leq 1 \wedge x_1 - x_2 \leq 0 \wedge x_2 \leq 3 \wedge x_2 - x_1 \leq 2$, which can be written as the following DBM: For instance, $M[2,0] = (\leq, 3)$ represents the constraint $x_2 - x_0 \leq 3$, i.e., $x_2 \leq 3$. The diagram to the right of the figure represents the set $[M]$.*

$$M = \begin{pmatrix} (\leq,0) & (\leq,0) & (\leq,0) \\ (\leq,1) & (\leq,0) & (\leq,0) \\ (\leq,3) & (\leq,2) & (\leq,0) \end{pmatrix}$$

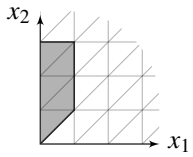


Figure 9.3: Example of a DBM

We now define elementary operations on DBMs which are used to explore the state space of timed games. We start by giving set-theoretic definitions and then comment on their computation with DBMs.

Let $\mathsf{Post}_{\geq 0}(Z)$ denote the zone describing the *time-successors* of Z, and $\mathsf{Pre}_{\geq 0}(Z)$ the *time-predecessors* of Z. Formally,

$$\mathsf{Post}_{\geq 0}(Z) = \{v \in \mathbb{R}^{\mathscr{C}}_{\geq 0} \mid \exists t \geq 0.\ v - t \in Z\}$$
$$\mathsf{Pre}_{\geq 0}(Z) = \{v \in \mathbb{R}^{\mathscr{C}}_{\geq 0} \mid \exists t \geq 0.\ v + t \in Z\}.$$

Given $R \subseteq \mathscr{C}$, we also define

$$\mathsf{Reset}_R(Z) = \{v \in \mathbb{R}^{\mathscr{C}}_{\geq 0} \mid \exists v' \in Z.\ v = v'[R \leftarrow 0]\}$$
$$\mathsf{Unreset}_R(Z) = \{v \in \mathbb{R}^{\mathscr{C}}_{\geq 0} \mid \exists v' \in Z.\ v' = v[R \leftarrow 0]\}.$$

These operations, together with intersection, suffice to describe one-step successors and predecessors by an edge of a timed automaton. For instance, given edge $e = (\ell, g, R, \ell')$ and set $S \subseteq \mathbb{R}^{\mathscr{C}}_{\geq 0}$, the set of states that are reached after letting time elapse and taking edge e can be obtained as

$$\mathsf{Post}_e(S) = \mathsf{Reset}_R(\mathsf{Post}_{\geq 0}(S) \cap G),$$

where G denotes the zone corresponding to the guard g. Similarly, we can compute the predecessors of S by edge e as

$$\mathsf{Pre}_e(S) = \mathsf{Pre}_{\geq 0}(G \cap \mathsf{Unreset}_R(S)).$$

We illustrate these constructions on Figure 9.4.

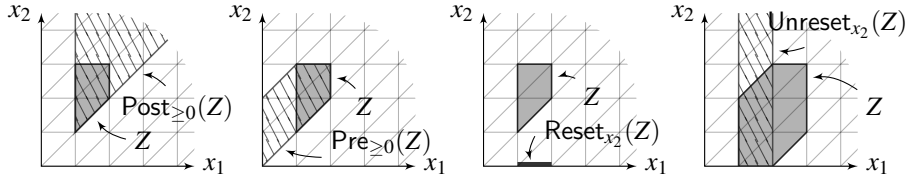


Figure 9.4: Operations on zones

It is not hard to prove that the above operations preserve zones: if S is a zone, then so is the result of any of these operations. Moreover, each single operation can be

computed in time $O(|\mathscr{C}|^3)$ using the DBM representation. The underlying algorithms often modify some elements of the matrix and run an all-pairs shortest path algorithm, namely the Floyd-Roy-Warshall algorithm, on a graph whose adjacency matrix is the given DBM. Computing the shortest paths renders the DBM *canonical*; in fact, this allows one to compute the tightest constraints on all differences of clock pairs, and this can be shown to yield a unique representation of a given zone.

Let us call the above operations *basic operations* on DBMs [BY04].

Theorem 101 (Complexity of basic operations on DBMs)**.** *Given a DBM of size $n \times n$, any basic operation yields a DBM and can be computed in time $O(n^3)$.*

Observe that a DBM always describes a convex subset of $\mathbb{R}^{\mathscr{C}}_{\geq 0}$ since it is a conjunction of convex clock constraints. However, the set of winning states is in general non-convex in timed games. The simple arena of Figure 9.5 provides an example: if Eve's objective is to reach ℓ_1, then it should just avoid the configurations satisfying $1 \leq x_1, x_2 \leq 2$. But this set of predecessors is then non-convex as shown in Figure 9.5. We thus have to work with unions of zones, also called federations of zones, or *federations* for short.

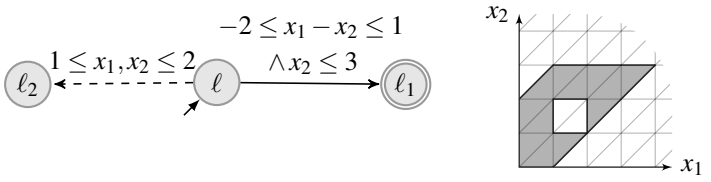


Figure 9.5: Winning configurations (in ℓ) for Eve to ensure reaching ℓ_1.

One particular operation that we need is complementation. The complement of a convex set is of course not convex in general, but we can still compute, in polynomial time, the complement of a DBM M, written M^c, as a federation of DBMs.

Theorem 102 (Complement of DBMs)**.** *The complement of a DBM of size $n \times n$ can be computed as a federation of at most $n(n-1)$ DBMs.*

The above theorem is seen easily as follows. Since a DBM represents a conjunction of constraints, the complement is computed easily as the disjunction of the complement of each elementary constraint appearing in it. For instance, the complement of $x_1 \leq 1 \wedge x_2 \geq 2$ is $x_1 > 1 \vee x_2 < 2$, which can be represented as the federation of two zones.

In the rest of this chapter, we describe two algorithms to solve timed games using the DBM data structure and the operations introduced above. Our algorithms are extensions of those used for finite games, but we explore the set of zones instead of the set of vertices, and predecessor and successor operations are replaced by their zone-based counterparts.

As for finite games, we are interested in computing a fixed point to determine whether a given configuration is winning for Eve. We start by introducing the zone-based counterparts of the controllable predecessors operator which is the main tool in the algorithms.

9.3 Controllable-Predecessor Operator

We present the controllable-predecessor operator which, given sets of states X and Y, returns the set of states from which Eve can reach X in one step, while avoiding states of Y during Eve's delay. Intuitively, the states in Y are the states from which Adam may force an action leading outside of X, which Eve would better avoid.

Recall that $V = \mathscr{L} \times \mathbb{R}^{\mathscr{C}}_{\geq 0}$. The set of safe time-predecessors to reach $X \subseteq V$ while avoiding $Y \subseteq V$ is defined as follows:

$$\mathsf{Pred}_{\geq 0}(X,Y) = \{(\ell,\nu) \in V \mid \exists d \geq 0.\ (\ell,\nu+d) \in X \wedge \forall d' \in [0,d).\ (\ell,\nu+d') \notin Y\}.$$

In words, from any configuration of $\mathsf{Pred}_{\geq 0}(X,Y)$, the set X can be reached by delaying while never crossing any configuration of Y on the way. For any $X \subseteq V$, let us denote

$$\mathsf{Pred}_c(X) = \{(\ell,\nu) \in V \mid \exists e \in E_{\text{Eve}}(\ell).\ \exists (\ell',\nu') \in X.\ (\ell,\nu) \xrightarrow{e} (\ell',\nu')\}.$$

This is the set of immediate predecessors of X by edges in E_{Eve}. Symmetrically, we define $\mathsf{Pred}_u(X)$ using E_{Adam} instead of E_{Eve}. Our controllable-predecessor operator is then defined as

$$\pi(X) = \mathsf{Pred}_{\geq 0}(X \cup \mathsf{Pred}_c(X), \mathsf{Pred}_u(V \setminus X)).$$

Intuitively, the states in $\pi(X)$ are those from which Eve can wait until she can take a controllable transition to reach X, and so that no transitions that Adam could take while Eve is waiting may lead outside of X.

Lemma 126. *The operator π is order-preserving: if $X \subseteq Y$, then $\pi(X) \subseteq \pi(Y)$.*

Example 5 (Example for the controllable predecessor operator)**.** *Consider the timed game to the left of Figure 9.6. In that game, Eve can reach her target when clock x_2 reaches value 3 while x_1 is in $[1;4]$. However, Adam can take the game to a bad state when simultaneously $x_1 \in [1;2]$ and $x_2 \leq 2$. The diagram to the right of Figure 9.6 shows the set of winning valuations for Eve: it is computed as $\mathsf{Pred}_{\geq 0}(X,Y)$ where $X = \mathsf{Pred}_c(\{W\} \times \mathbb{R}_{\geq 0})$ and $Y = \mathsf{Pred}_u(\{\ell_0,\ell_1\} \times \mathbb{R}_{\geq 0})$.*

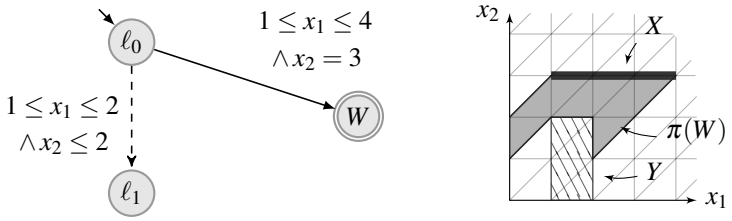


Figure 9.6: Controllable predecessors

Remark 12 (Reduction for zero-sum objectives)**.** *When considering zero-sum objectives (which we do in this chapter), a turn-based arena can be built to decide whether*

Eve has a winning strategy (and possibly construct one). Intuitively, the turn-based arena is obtained by letting Eve first pick a pair (d,e) with $e \in E_{Eve}$, and then letting Adam either respect this choice (i.e., play a larger delay) or preempt Eve's action by choosing (d',e') with $d' \leq d$ and $e' \in E_{Adam}$. Eve has a winning strategy in this turn-based arena if, and only if she has one in the original game. However, this does not apply to Adam; if he has a winning strategy, this only means that Eve does not have a winning strategy in the original game.

Given $\mathscr{T} = (\mathscr{L}, \mathscr{C}, E_{Eve}, E_{Adam}, c)$, we define the arena $\mathscr{A} = (V, V_{\text{Eve}}, V_{\text{Adam}}, E, c'')$ with objective Ω, where $V_{\text{Eve}} = \mathscr{L} \times \mathbb{R}_{\geq 0}^{\mathscr{C}}$ and $V_{\text{Adam}} = \mathscr{L} \times \mathbb{R}_{\geq 0}^{\mathscr{C}} \times (\mathbb{R}_{\geq 0} \times E_{Eve} \cup \{\bot\})$. The set of successors of configuration $(\ell, \nu) \in V_{\text{Eve}}$ is the set $\{(\ell, \nu, a) \mid a \text{ available at } (\ell, \nu)\}$. From a configuration $((\ell, \nu), (d,e)) \in V_{\text{Adam}}$, several cases may occur:

- *if Adam has no available action from (ℓ, ν), or if he can only play actions (d',e') with $d' > d$, then the only transition from $((\ell, \nu), (d,e))$ goes to $\mathsf{step}((\ell, \nu), (d,e))$;*
- *Otherwise, for all actions (d',e') for Adam satisfying $d' \leq d$, there is a transition from $((\ell, \nu), (d,e))$ to $\mathsf{step}((\ell, \nu), (d',e'))$. Moreover, if Adam has an available action (d',e') with $d' \geq d$, then there is also a transition from $((\ell, \nu), (d,e))$ to $\mathsf{step}((\ell, \nu), (d,e))$.*

From a configuration $((\ell, \nu), \bot)$, there is a transition to $\mathsf{step}((\ell, \nu), (d',e'))$ for each available action (d',e') of Adam. The coloring function is defined as $c''((\ell, \nu)) = c(\ell)$ and $c''((\ell, \nu), (d,e)) = c(\ell)$.

9.4 Backward Algorithm

Given the DBM data structure we presented, a backward fixed point algorithm follows for computing the winning configurations in a timed game:

Theorem 103 (Correctness of the backward algorithm)**.** *For any timed game $\mathscr{A}$ and target set $G \subseteq V_{\text{Eve}}$, the limit $\lim_{k \to \infty} \pi^k(G)$ exists, and is reached in a finite number of steps. This limit is precisely the set of states from which the controller has a winning strategy for reaching G.*

Sketch. We briefly explain why the fixed point computation terminates and refer to [AMPS98, CDF+05] for more details. The proof relies on the notion of *regions*. Intuitively, regions are minimal zones, not containing any other zone. More precisely, writing K for the maximal (absolute) integer constant appearing in the timed game, the set of regions is the set of zones associated with DBMs $(\prec_{i,j}, M_{i,j})$ such that

- $M_{i,j} \in [-K; K] \cup \{-\infty; +\infty\}$ for all i and j;
- for all $i \neq j$,
  - either $M_{i,j} = -M_{j,i}$ and $\prec_{i,j} = \prec_{j,i} = \leq$,
  - or $|M_{i,j} + M_{j,i}| = 1$ and $\prec_{i,j} = \prec_{j,i} = <$,
  - or $(\prec_{i,j}, M_{i,j}) = (<, +\infty)$ and $(\prec_{j,i}, M_{j,i}) = (<, -K)$.

The set of regions form a finite partition of $\mathbb{R}^{\mathscr{C}}_{\geq 0}$. The main argument for the proof is that the image by π of any finite union of regions is a finite union of regions. Since π is non-decreasing, the sequence $\pi^k(G)$ converges after finitely many steps.

The fact that $\pi^k(G)$ may only contain winning configurations follows from our construction, and can be proved easily by induction on k. One can also show that for all configurations outside of $\pi^k(G)$, there is no strategy that is winning within k steps. This follows since the definition of $\pi(\cdot)$ gives the actions to be played by Adam to prevent Eve from winning. Note that this does not necessarily yield a winning strategy for Adam, since the game is not determined. □

9.5 Forward Algorithm

The backward algorithm we just presented is conceptually simple, but it is often not very efficient in practice, as federations tend to grow too much in size in each iteration of the computation. The forward algorithm we will now present is more efficient in practice. It performs a forward exploration and only applies the controllable predecessor along branches that actually reach the target state from the initial state. If the witness trace is not excessively long, which is often the case in practice, this limits the size of the federations.

We present below the algorithm proposed in [CDF+05], and as a first step, we explain the untimed version of that algorithm, based on an algorithm of Liu and Smolka for computing fixed points [LS98].

A Forward Algorithm for Finite-State Games

The original algorithm of Liu and Smolka is expressed in terms of *pre-fixed points* in *dependency graphs*: a dependency graph is a pair $G = (V,E)$ in which $E \subseteq V \times 2^V$ relates states with sets of states. For any order-preserving function $f\colon 2^V \to 2^V$ (*order-preserving* meaning non-decreasing for the $\subseteq$-relation), a *pre-fixed point* is a set $X \subseteq V$ for which $f(X) \subseteq X$; it is a *fixed point* if $f(X) = X$. By Knaster-Tarski theorem, such functions always admit a least pre-fixed point which is also the least fixed point.

Fix a dependency graph $G = (V,E)$. For $W \subseteq V$, we define a mapping $f_W\colon 2^V \to 2^V$: for each $X \subseteq V$, we let $f_W(X) = W \cup \{v \in V \mid \exists (v,Y) \in E.\, Y \subseteq X\}$. Clearly, $X \subseteq X'$ implies $f_W(X) \subseteq f_W(X')$, so that f_W admits a least (pre-)fixed point. The Liu-Smolka algorithm aims at deciding whether a given vertex $v_0 \in V$ belongs to the least fixed point of f_W. Classical algorithms for computing least fixed points consist in iteratively computing $(f^i_W(\emptyset))_i$ until convergence (assuming V is finite). Observe that this corresponds to the algorithm of Theorem 103 for timed games. The Liu-Smolka algorithm proceeds from v_0, and explores the dependency graph until it can claim that v_0 is, or that it is not, in the least fixed point of f_W.

Before tackling the algorithm, let us link least fixed points in dependency graphs and winning sets in concurrent games (with reachability objectives): with a concurrent arena $\mathscr{C} = (V, \mathsf{Act}, \delta, c')$ and a target set Win, we associate the dependency graph $G = (V,E)$, where $(v,T) \in E$ whenever $v \in V$ and $T \subseteq V$ is such that there exists an action a for which $T = \{v' \mid \exists a' \in \mathsf{Act}.\, v' = \delta(v,a,a')\}$. Then for any set $X \subseteq V$, the set $f_{\mathrm{Win}}(X)$

contains Win and all the states from which Eve can force a visit to X in one step. We then have:

Proposition 3. *The least fixed point of f_{Win} in G corresponds to the set W of winning states for Eve in $\mathscr{C}$.*

Proof. The winning states of Eve form a pre-fixed point of f_{Win} containing Win: indeed, for any $v \in f_{\mathrm{Win}}(W)$, either $v \in \mathrm{Win}$, or Eve has an action to move from v to some state in W. Hence v is winning, i.e., $v \in W$.

Conversely, from any state v that is not in the least pre-fixed point X, for any edge (v,T), there is a state $v' \in T$ that again is not in X. This defines a strategy for Adam to avoid reaching Win, so that Eve does not have a winning strategy from v. □

Algorithm 9.1: Liu-Smolka algorithm for least fixed point of f_W

```
Data: A dependency graph G = (V,E), a set W ⊆ V, a node v0 ∈ V
Result: Is v0 in the least fixed point of f_W?
for v ∈ V do
    if v ∈ W then F(v) := 1;
    else if v == v0 then F(v) := 0;
    else F(v) := ⊥;
Dep(v0) := ∅;
Wait := {(v,T) ∈ E | v = v0};
while (Wait ≠ ∅ and F(v0) == 0) do
    (v,T) := pop(Wait);
    if F(v) == 0 and ∀v' ∈ T. F(v') == 1 then                    // case 1
        F(v) := 1;
        Wait := Wait ∪ Dep(v);
    else if ∃v' ∈ T. F(v') == 0 then                             // case 2
        Dep(v') := Dep(v') ∪ {(v,T)};
    else if ∃v' ∈ T. F(v') == ⊥ then                             // case 3
        F(v') := 0;
        Dep(v') := {(v,T)};
        Wait := Wait ∪ {(w,U) ∈ E | w = v'};
return F(v0)
```

[2]

Algorithm 9.1 can be seen as an alternation of forward exploration and backward propagation. Intuitively, the algorithm first explores the graph in a forward manner, remembering for each node v the set $\mathsf{Dep}(v)$ of nodes that *depend* on v, and have to be reexplored if the status of v is updated. For each v, the algorithm maintains a value $F(v)$, which is $\bot$ if v has not been explored yet, 0 if v has been explored but not

[2] Our version of the algorithm slightly differs from the original one [LS98]: we let $\mathsf{Dep}(v') := \{(v,T)\}$ at the penultimate line of the **while** loop (which otherwise results in a wrong result, as already noticed in [JLSØ13]), reinforce the condition for case 1 (which otherwise would not guarantee termination), and reinforce the condition of the **while** loop to get earlier termination.

yet been shown to be winning, and 1 if v is known to be winning. Whenever a vertex v whose all successors are winning is found, the value of v is set to 1, and its parents (in $\mathsf{Dep}(v)$) are scheduled to be visited again to check whether their statuses have to be changed. This is how the backward propagation is triggered; in fact, the search will climb in the tree as long as the values of vertices can be updated to 1.

The correctness of this algorithm relies on the following lemma [LS98]:

Lemma 127 (Invariant)**.** *The following properties hold at the end of each run in the **while** loop of Algorithm 9.1:*

- *for any $v \in V$, if $F(v) = 1$, then v belongs to the least fixed point containing W;*
- *for any $v \in V$ with $F(v) = 0$, and any $(v,T) \in E$, either $(v,T) \in \mathsf{Wait}$ or $(v,T) \in \mathsf{Dep}(v')$ for some $v' \in T$ with $F(v') = 0$.*

Proof. We fix an execution of Algorithm 9.1, and prove that both claims are true at the beginning and end of each iteration through the **while** loop. To clarify the presentation, we use superscript i to indicate the value of variables at the end of the i-th run through the loop, so that Wait^3 is the value of variable Wait after three iterations (and Wait^0 is its value after initialization). In particular, (v^i, T^i) is the symbolic transition popped from the Wait list during the i-th iteration (and is not defined for $i = 0$).

Let us first prove the first property: the initialization phase clearly enforces that if $F^0(x) = 1$, then $x \in W$, which is included in any fixed point of f_W. Now, assume that the property holds true at the beginning of the i-th run through the loop (i.e., if $F^{i-1}(x) = 1$, then x is in the least fixed point), and pick some $x \in V$ such that $F^i(x) = 1$. If $F^{i-1}(x) = 1$, our result follows; if not, then the i-th iteration of the loop must have run via case 1, hence $F^{i-1}(x') = 1$ for all $x' \in T^i$. By our induction hypothesis, this indicates that T^i is part of the least fixed point, and by definition of f_W, x must also belong to the least fixed point.

The second statement also clearly holds after initialization: initially, $F^0(x) = 0$ only for $x = v_0$, and all transitions from v_0 have been stored in Wait^0. We now assume that the property holds when entering the **while** loop for the i-th time, and consider x such that $F^i(x) = 0$ at the end of that loop. We pick $(x,Y) \in E$.

- if the i-th run through the loop visits case 1, then already $F^{i-1}(x) = 0$ (hence $x \neq v^i$). Then either $(x,Y) \in \mathsf{Wait}^{i-1}$, or $(x,Y) \in \mathsf{Dep}^{i-1}(x')$ for some $x' \in Y$ with $F(x') = 0$. In the former case: since $x \neq v^i$, if $(x,Y) \in \mathsf{Wait}^{i-1}$ then also $(x,Y) \in \mathsf{Wait}^i$; in the latter case: if $(x,Y) \in \mathsf{Dep}^{i-1}(x')$ for some $x' \in Y$ with $F(x') = 0$, then *(i)* either $x' = v^i$, and $(x,Y) \in \mathsf{Wait}^i$ because $\mathsf{Dep}^{i-1}(v^i) \subseteq \mathsf{Wait}^i$ (last line of case 1), *(ii)* or $x' \neq v^i$, and $(x,Y) \in \mathsf{Dep}^{i-1}(x') = \mathsf{Dep}^i(x')$ and $F^i(x') = F^{i-1}(x') = 0$.
- if the i-th run goes to case 2, then again $F^{i-1}(x) = 0$, and the induction hypothesis applies: either (x,Y) is in Wait^{i-1}, or it is in $\mathsf{Dep}^{i-1}(x')$ for some $x' \in Y$ with $F^{i-1}(x') = 0$. For the latter case, observe that $\mathsf{Dep}^{i-1}(x') \subseteq \mathsf{Dep}^i(x')$, and $F^i(x') = F^{i-1}(x')$ when running case 2; for the former case, we have $\mathsf{Wait}^i = \mathsf{Wait}^{i-1} \setminus \{(v^i, T^i)\}$, so that (x,Y) remain in Wait^i if $(x,Y) \neq (v^i, T^i)$; finally, if $(x,Y) = (v^i, T^i)$, then case 2 precisely adds (v^i, T^i) to $\mathsf{Dep}^i(v')$ for some $v' \in T^i$ with $F^i(v') = 0$, which concludes the proof for this case.

- for case 3, we first consider the case where x is the state v' selected at the beginning of case 3: in that case, all transitions from x are added to Wait, so that $(x,Y) \in \mathsf{Wait}^i$. Now, if x is not the selected state v', then $F^{i-1}(x) = 0$, and again either $x \in \mathsf{Wait}^{i-1}$ or $x \in \mathsf{Dep}^{i-1}(x')$ for some $x' \in Y$ with $F^{i-1}(x') = 0$. The latter case is preserved when running case 3; if $x \in \mathsf{Wait}^{i-1}$: if $(x,Y) \neq (v^i, T^i)$, then $x \in \mathsf{Wait}^i$, while if $(x,Y) = (v^i, T^i)$, then $(x,Y) \in \mathsf{Dep}^i(v')$ for the state $v' \in T^i$ selected at the beginning of case 3 (and for which $F^i(v') = 0$).

□

As a corollary, if the algorithm terminates after n rounds of the **while** loop, then either $F^n(v_0) = 1$, or $F^n(v_0) = 0$ and $\mathsf{Wait}^n = \emptyset$.

- From the first claim of the lemma above, the former case entails that v_0 belongs to the least fixed point.

- Now consider the second case, and let $B = \{v \in V \mid F^n(v) \in \{\bot, 1\}\}$. We prove that $f_W(B) \subseteq B$. For this, we pick $v \in f_W(B)$:

  - if $v \in W$, then $F(v)$ is set to 1 initially, and may never be changed, so that $v \in B$;
  - otherwise, there is a transition (v,T) such that $T \subseteq B$. If $v \notin B$, then $(v,T) \in \mathsf{Dep}^n(v')$ for some $v' \in T$ with $F^n(v') = 0$, which contradicts the fact that $T \subseteq B$.

  This proves that $f_W(B) \subseteq B$, so that B is a pre-fixed point of f_W. It thus contains the least (pre-)fixed point of f_W, so that any state v not in B (i.e., with $F^n(v) = 0$) for sure does not belong to that least fixed point. In particular, v_0 is not in the fixed point.

It remains to prove termination. For this, we first notice that, for any hyperedge (v,S), if $(v,S) \in \mathsf{Dep}^i(v')$ then $(v,S) \in \mathsf{Wait}^j$ for some $j < i$, and if $(v,S) \in \mathsf{Wait}^j$ or $(v,S) \in \mathsf{Dep}^j(v')$ for some v', then $F^j(v) \neq \bot$.

We then set

$$M = 2|\mathsf{Wait}| + 2 \sum_{v \text{ s.t. } F(v)=\bot} |\{(v,S) \in E\}| + \sum_{v \text{ s.t. } F(v)=0} |\mathsf{Dep}(v)| - \sum_{v \text{ s.t. } F(v)=1} |\mathsf{Dep}(v)|$$

again writing M^i for the value of M after the i-th run through the **while** loop. This value is at most $2|E|$ when entering the **while** loop for the first time; clearly, it can never go below $-|E|$. We now prove that $M^i < M^{i-1}$, which implies termination of the algorithm.

Consider the i-th run through the loop, popping (v^i, T^i) from Wait^{i-1} (which decreases M by 2). We consider all three cases:

- if case 1 applies, $F^i(v)$ is set to 1. The set $\mathsf{Dep}^{i-1}(v)$ is added to Wait^i. This globally leaves M unchanged, so that globally $M^i = M^{i-1} - 2$;

- if case 2 applies, $\mathsf{Dep}^{i-1}(v')$ is augmented by (at most) one edge, and since $F^i(v') = 0$, this increases M by at most 1. Hence globally $M^i \leq M^{i-1} - 1$;
- finally, case 3 increases M by 1: the edges added to Wait are compensated by the fact that $F(v')$ no longer is $\bot$, and only one extra edge has to be considered in the second sum. Hence again $M^i \leq M^{i-1} - 1$.

This concludes the correctness proof of Algorithm 9.1.

Theorem 104. *Algorithm 9.1 terminates, and returns* 1 *if, and only if,* v_0 *belongs to the least fixed point of* f_W.

Using Proposition 3, we get a *forward* algorithm for deciding if a given state of a concurrent game is winning for Eve. This corresponds to the OTFUR algorithm of [CDF+05].

Extension to Timed Games

We now explain how to adapt the algorithm above to (infinite-state) timed games. For efficiency, the algorithm relies on zones (and DBMs); instead of computing whether a given zone (ℓ, Z) is winning, the algorithm maintains, for each zone $S = (\ell, Z)$ it explores, a subzone (ℓ, Z') of configurations that it knows are winning; this subzone is stored as $F(S)$, and is updated during the execution. As in Algorithm 9.1, a waiting list keeps track of the zones to be explored, and a dependency list stores the list of nodes to be revisited upon update of the winning subzone of a zone. The algorithm is given in Algorithm 9.2.

The correctness of the algorithm can be proven using the following lemma. We omit the proof of this lemma, as it is tedious and and does not contain any difficult argument.

Lemma 128. *The following properties hold at the end of each run through the* ***while*** *loop of Algorithm 9.2:*

- *for any* $S \in \mathsf{Passed}$ *and any transition* α*, if* $T = \mathsf{Post}_{\geq 0}(\mathsf{Post}_\alpha(S)) \neq \emptyset$*, then either* $(S, \alpha, T) \in \mathsf{Wait}$*, or* $T \in \mathsf{Passed}$ *and* $(S, \alpha, T) \in \mathsf{Dep}(T)$*;*
- *for any* $S \in \mathsf{Passed}$ *and* $q \in F(S)$*,* q *is winning for Eve;*
- *for any* $S \in \mathsf{Passed}$ *and* $q \in S \setminus F(S)$*, either* Wait *contains a symbolic transition* (S, α, S') *from* S *with* $S' \in \mathsf{Passed}$*, or*

$$q \notin \mathsf{Pred}_{\geq 0}\Big(F(S) \cup \bigcup_{\substack{S \xrightarrow{c} V \\ V \in \mathsf{Passed}}} \mathsf{Pred}_c(F(V)), \bigcup_{\substack{S \xrightarrow{u} V \\ V \in \mathsf{Passed}}} \mathsf{Pred}_u(V \setminus F(V))\Big) \cap S.$$

The proof is omitted but can be found in [CDF+05].

Using these invariants, we get the following result:

Lemma 129. *If Algorithm 9.2 terminates, all configurations in* $F(S)$ *for any* $S \in \mathsf{Passed}$ *are winning for Eve; if additionally* $\mathsf{Wait} = \emptyset$*, then all configurations in* $S \setminus F(S)$*, for any* $S \in \mathsf{Passed}$*, are losing for Eve.*

Algorithm 9.2: Symbolic on-the-fly algorithm for timed reachability

Data: A reachability timed game $\mathscr{G} = (\mathscr{A}, \texttt{Reach}(\text{Win}))$, a location $\ell_0 \in \mathscr{L}$
Result: Is $(\ell_0, \mathbf{0})$ winning for Eve?
$S_0 := \mathsf{Post}_{\geq 0}(\ell_0, \mathbf{0})$;
if $c(S_0) == \mathit{Win}$ **then** $F(S_0) := S_0$ **else** $F(S_0) := \emptyset$;
$\mathsf{Passed} := \{S_0\}$; // Passed stores all configurations for which F is defined
$\mathsf{Dep}(S_0) := \emptyset$;
$\mathsf{Wait} := \{(S_0, \alpha, T) \mid T = \mathsf{Post}_{\geq 0}(\mathsf{Post}_\alpha(S_0)) \neq \emptyset, \alpha \text{ transition of } \mathscr{G}\}$;
while *(*$\mathit{Wait} \neq \emptyset$ *and* $(v_0, \mathbf{0}) \notin F(S_0)$*)* **do**
  $(S, \alpha, T)) := \texttt{pop}(\mathit{Wait})$;
  if $T \in \mathit{Passed}$ **then** // case A
    $\mathsf{Dep}(T) := \mathsf{Dep}(T) \cup \{(S, \alpha, T)\}$;
    $W :=$
    $\mathsf{Pred}_{\geq 0}\left(F(S) \cup \bigcup_{\substack{S \xrightarrow{c} V \\ V \in \mathsf{Passed}}} \mathsf{Pred}_c(F(V)), \bigcup_{\substack{S \xrightarrow{u} V \\ V \in \mathsf{Passed}}} \mathsf{Pred}_u(V \setminus F(V))\right) \cap$
    S;
    if $F(S) \subsetneq W$ **then**
      $F(S) := W$;
      $\mathsf{Wait} := \mathsf{Wait} \cup \mathsf{Dep}(S)$;
  else // case B
    $\mathsf{Passed} := \mathsf{Passed} \cup \{T\}$;
    if $c(T) == \mathit{Win}$ **then**
      $F(T) := T$
      $\mathsf{Wait} := \mathsf{Wait} \cup \{(S, \alpha, T)\}$
    else
      $F(T) := \emptyset$
    $\mathsf{Dep}(T) := \{(S, \alpha, T)\}$;
    $\mathsf{Wait} := \mathsf{Wait} \cup \{(T, \alpha, U) \mid U = \mathsf{Post}_{\geq 0}(\mathsf{Post}_\alpha(T)) \neq$
    $\emptyset, \alpha \text{ transition of } \mathscr{G}\}$;
if $(v_0, \mathbf{0}) \in F(S_0)$ **then return** 1 **else return** 0;

Proof. The first statement corresponds to the second statement of Lemma 128. Now, assume $\mathsf{Wait}^n = \emptyset$ after termination at the n-th step, and let $L = \{q \in \mathscr{L} \times \mathbb{R}^{\mathscr{C}}_{\geq 0} \mid \exists S \in \mathsf{Passed}^n.\ q \in S \setminus F^n(S)\}$, and M be the complement of L. For any $S \in \mathsf{Passed}^n$, we then have $M \cap S \subseteq F^n(S)$.

Pick $q \in \pi(M) \cup \mathrm{Win}$, where we abusively write Win to denote all configurations with location colored Win. We prove that $q \in M$:

- if $q \in \mathrm{Win}$: assume $q \in L$, and pick $S \in \mathsf{Passed}^n$ such that $q \in S \setminus F^n(S)$. Since $q \in S$, it holds $c(S) = \mathrm{Win}$; but then $F^n(S)$ is defined (since $S \in \mathsf{Passed}^n$), and it equals S by initialization of F. This contradicts the fact that $q \in S \setminus F^n(S)$, hence $q \in M$.

- if $q \in \pi(M)$: we again assume $q \in L$. Then for some $S \in \mathsf{Passed}^n$, $q \in S \setminus F^n(S)$. By the third property of Lemma 128, $q \notin W^n$ (because Wait^n is empty). Since $M \cap U \subseteq F^n(U)$ for all $U \in \mathsf{Passed}^n$ and $F^n(U) = \emptyset$ for all $U \notin \mathsf{Passed}^n$, and by monotonicity, we get

$$q \notin \mathsf{Pred}_{\geq 0}\left((M \cap S) \cup \bigcup_{S \xrightarrow{c} V} \mathsf{Pred}_c(M \cap V),\ \bigcup_{S \xrightarrow{u} V} \mathsf{Pred}_u(V \setminus (M \cap V))\right) \cap S.$$

  Now, notice that any $S \in \mathsf{Passed}^n$ is closed under $\mathsf{Post}_{\geq 0}$. Then

$$\begin{aligned}\pi(M) \cap S &= \mathsf{Pred}_{\geq 0}(M \cup \mathsf{Pred}_c(M), \mathsf{Pred}_u(\overline{M})) \\ &= \mathsf{Pred}_{\geq 0}((M \cap S) \cup (\mathsf{Pred}_c(M) \cap S), \mathsf{Pred}_u(\overline{M})).\end{aligned}$$

  Now, it can be checked that $\mathsf{Pred}_c(M) \cap S \subseteq \bigcup_{S \xrightarrow{c} V} \mathsf{Pred}_c(M \cap V)$ and $\mathsf{Pred}_u(\overline{M}) \supseteq \bigcup_{S \xrightarrow{u} V} \mathsf{Pred}_u(V \setminus (M \cap V))$. So the fact that $q \in \pi(M)$ and $q \in S$ leads to a contradiction. This entails that $q \notin L$, hence $q \in M$.

In the end, we have proven that $\pi(M) \cup \mathrm{Win} \subseteq M$, so that M is a pre-fixed point of $X \mapsto \pi(X) \cup \mathrm{Win}$, hence it contains all winning configurations of Eve and L only contains losing configurations. □

The procedure given in Algorithm 9.2 will in general not terminate: as in the case of timed automata, the number of zones generated by the algorithm may be infinite. This is classically avoided using *extrapolation*: this consists in abstracting the zones being considered by larger zones, defined by only using integer constants less than the maximal constant appearing in the timed arena (as we did for regions in the proof of Theorem 103). This can be proven to preserve correctness, and makes the number of zones finite. Termination follows by noticing that any triple (S, α, T) may be added to Wait only a finite number of times (bounded by the number of regions in T). We can then conclude:

Theorem 105. *Algorithm 9.2 terminates when extrapolation is used, and returns* 1 *if, and only if,* $(\ell_0, \mathbf{0})$ *is winning for Eve.*

Bibliographic references

Timed games were first introduced by Asarin, Maler, Pnueli and Sifakis [MPS95, AMPS98], in a slightly different form. Our presentation is based on the algorithms proposed by Liu and Smolka [LS98], and extended to timed games by Cassez, David, Fleury, Larsen and Lime [CDF+05]; its main advantage is that it runs *on-the-fly*, building (part of) the arena while exploring it, and terminating as soon as a winning strategy is found. A first on-the-fly algorithm was proposed by Tripakis and Altisen in [TA99], but it was not fully on-the-fly as it would run on a quotient graph of the timed arena, which involves an expensive preprocessing step.

Timed games—and already timed automata—may exhibit unrealistic behaviours, such as finite-duration executions containing infinitely many transitions (often referred to as *Zeno* behaviours). In our semantics, Adam may always prevent Eve from playing her move by choosing a shorter delay than hers, even if it means selecting a convergent sequence of delays. Figure 9.7 displays a simple example of such a situation: in that game, Adam may prevent Eve from reaching her winning state q_1 by always selecting a delay shorter than the delay proposed by Eve.

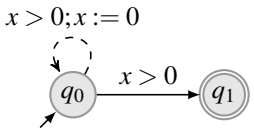


Figure 9.7: A timed game where Adam may prevent Eve from reaching q_1

An alternative semantics of timed games was proposed by de Alfaro, Faella, Henzinger, Majumdar, and Stoelinga [dAFH+03] to circumvent this problem: it consists in *blaming* at each round the player playing the shortest delay, and declaring any player losing (even if she reaches her goal) in case the infinite sequence of delays converges and that player received infinitely many blames. For such a semantics, Eve has a winning strategy in the game of Figure 9.7, which consists in proposing a converging sequence of delays, until her action is applied; such a strategy requires infinite memory, but strategies with randomized delays can circumvent this [CHP08]. Other approaches to avoid arbitrary-precision strategies have been explored in [BMS15, BFM15, LLTW14, ORS14].

Timed games have been extended with weights, in order to model other quantities besides time (e.g. energy consumption). This is somewhat similar to the finite-state games extended with payoffs of Chapter 4; in the timed setting however, besides evolving along transitions in the game, the payoff is also modified when timed elapses, and the change is proportional to the time spent. As proven in [BBR05, BBM06], optimal reachability (a.k.a. the shortest-path problem) is undecidable in that setting, but arbitrary-precise approximation of the optimal cost can be computed [BJM15].

Timed games with partial observability have been investigated in [BDMP03]: in that setting, Eve only has partial observation of the state and clock valuation of the arena; she also owns a finite set of clocks she can use to measure other delays. Whether Eve has a winning strategy in such a setting has been proved decidable if the set of clocks

and their precision are fixed; it is undecidable if they are not fixed. A different setting was developed in [CDL+07], with *stuttering-invariant strategies*, which are triggered by observation changes. The on-the-fly algorithm presented in this chapter can be adapted to that setting [CDL+07].

The algorithm we presented in this chapter is implemented in the tool Uppaal TiGa [BCD+07]. Uppaal TiGa has been applied on various cases [CJL+09], and combined with reinforcement-learning techniques to efficiently synthesise, optimise and evaluate strategies for stochastic timed games [DJL+15]. Another approach, relying on abstraction-refinement techniques, was proposed in [EMP10]; it merges locations so as to obtain simpler games, while weakening one player and strengthening her opponent. Solving the smaller games provides approximations on the winning set, which are used to refine the abstractions and accelerate their analyses. This approach is implemented in the tool Synthia [PEM11].

PUSHDOWN GAMES

Chapter 10

Pushdown Games

ARNAUD CARAYOL, OLIVIER SERRE

This chapter studies two-player games whose arena is defined by a pushdown system[1]. The vertices of the arena are the configurations of the pushdown system (i.e., pairs composed of a control state and a word representing the content of the stack) and the edges of the arena are defined by the pushdown system's transitions. For simplicity, both the ownership of a configuration and the objective will only depend on the control state of the configuration. Hence the partition of all the configurations between the two players will simply be given by a partition the control states. We will mainly consider the parity objective. Via a standard reduction (using a deterministic parity automaton, see Section 1.4), parity pushdown game can be used to solve any pushdown game with an ω-regular objectives (which in our setting is simply an ω-regular set of infinite words over the alphabet of control states).

The main conceptual novelty of this chapter is that the arena is no longer finite. However as these games are described by a finite amount of information: the pushdown system, the ownership partition and the ω-regular objective, they are amenable to algorithmic treatment. The first natural problem in this line is to decide the winner of the game from a given configuration. We will also consider the computation of *finite representations* for the winning regions and the winning strategies.

10.1 Notations

A *pushdown system* is a tuple $\mathscr{P} = (Q, Q_{\text{Eve}}, Q_{\text{Adam}}, \Gamma, \Delta, C)$ where:

- Q is a finite set of control states with $Q = Q_{\text{Eve}} \uplus Q_{\text{Adam}}$ which is partitioned between the two players Eve and Adam. Moreover with each state q is associated a colour $\mathfrak{c}(q) \in C$;

[1] We use here the term pushdown system rather than pushdown automaton to stress the fact that we are not considering these devices as language acceptors but rather focus on the transitions systems they define.

- Γ is the stack alphabet. There is a special bottom-of-stack symbol, denoted $\bot$, which does not belong to Γ; we let $\Gamma_\bot$ denote the alphabet $\Gamma \cup \{\bot\}$;
- $\Delta : Q \times \Gamma_\bot \to 2^{Q \times \{\text{pop}, \text{push}(\gamma) \mid \gamma \in \Gamma\}}$ is the transition relation. We additionally require that for all states $p, q \in Q$, $(q, \text{pop}) \notin \Delta(p, \bot)$, i.e., the bottom of stack symbol is never popped.

We call a pair $(p,s) \in Q \times \bot\Gamma^*$ a *configuration* of $\mathscr{P}$: p is the control state of the configuration while s is its stack content. We let $\text{sh}((p,s)) = |s| - 1$ denote the *stack height* of the configuration (p,s). Intuitively, if $(q, \text{push}(\gamma'))$ belongs to $\Delta(p,\gamma)$, the pushdown system in any configuration of the form $(p, s\gamma)$ can go to state q after pushing the symbol γ on top of the stack, leading to the configuration $(q, s\gamma\gamma')$. Similarly, if (q, pop) belongs to $\Delta(p,\gamma)$, the pushdown system in any configuration of the form $(p, s\gamma)$ can go to the configuration (q,s) after *poping* the top symbol from the stack.

A pushdown system induces an arena $\mathscr{A} = (G, V_{\text{Eve}}, V_{\text{Adam}})$ called a *pushdown arena* where

- the set of vertices is the set $V = Q \times \bot\Gamma^*$ of configurations of $\mathscr{P}$ with $V_{\text{Eve}} = Q_{\text{Eve}} \times \bot\Gamma^*$ and $V_{\text{Adam}} = Q_{\text{Adam}} \times \bot\Gamma^*$;
- the set E of edges induced by Δ is

$$\begin{aligned} E = &\{((p,s\gamma), \mathfrak{c}(p), (q,s)) \mid (q, \text{pop}) \in \Delta(p,\gamma)\} \quad \cup \\ &\{((p,s\gamma), \mathfrak{c}(p), (q,s\gamma\gamma')) \mid (q, \text{push}(\gamma')) \in \Delta(p,\gamma)\}. \end{aligned}$$

Remark 13. *In this chapter, we deviate slightly from the general setting used in the book as we colour vertices and not edges. Because we only consider qualitative objectives, it is more convenient to consider the equivalent setting where we label vertices by colours rather than edges which is the usual convention in pushdown games. In the definition of a pushdown arena, the colour of an edge is uniquely determined by the colour of the control state of the source vertex. Therefore, we also view in this chapter plays as sequences of vertices rather than sequences of edges.*

Finally, a *pushdown game* is a game played on a pushdown arena. In this chapter we only consider qualitative objectives of the form $\Omega \subseteq C^\omega$ where C is the set of colours. As in our definition, colours are associated to control states, the objective we consider only depend on the sequence of control states of the configurations visited along a play. By a slight abuse of notation, for a play $\pi \in V^*$, we let $\pi = (p_1,s_1)(p_2,s_2)\cdots \in \Omega$ denotes the fact that the sequences of colours $(\mathfrak{c}(p_i))_{i \geq 1}$ belongs to Ω.

Example 6. *Consider the pushdown system $\mathscr{P} = (Q, Q_{\text{Eve}}, Q_{\text{Adam}}, \Gamma, \Delta)$ where:*

- *$Q_{\text{Eve}} = \{q,r\}$ and $Q_{\text{Adam}} = \{p\}$; $\mathfrak{c}(p) = 1$, $\mathfrak{c}(q) = 2$ and $\mathfrak{c}(r) = 0$.*
- *$\Gamma = \{\gamma\}$ is a singleton.*
- *$\Delta(p,\bot) = \{(p, \text{push}(\gamma))\}$, $\Delta(p,\gamma) = \{(p, \text{push}(\gamma)), (q, \text{pop})\}$, $\Delta(q,\bot) = \{(r, \text{push}(\gamma))\}$, $\Delta(q,\gamma) = \{(q, \text{pop})\}$ and $\Delta(r,\gamma) = \{(q, \text{pop})\}$.*

Figure 10.1 depicts the part of the pushdown arena $\mathscr{A}$ induced by $\mathscr{P}$ when restricted to the vertices reachable from $(p, \bot)$.

Consider the reachability game $(\mathscr{A}, \texttt{Reach}(\{0\}))$. Then, every vertex of the form $(q, \bot v)$ is winning for Eve and every vertex of the form $(p, \bot v)$ is winning for Adam as Adam can always choose to push a γ-symbol while remaining in state p hence always avoiding the state r (which is the only state with colour 0). If instead we consider, the Büchi game $\texttt{Buchi}(\{0,2\})$, then Eve is winning form all vertices. The strategy for Adam consisting in always pushing a γ-symbol while remaining in state p results in a play that infinitely often sees the colour 2.

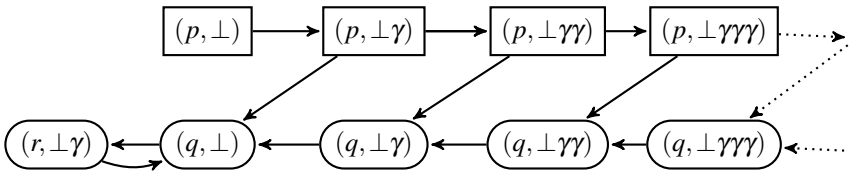


Figure 10.1: Pushdown arena from Example 6

As a pushdown arena is in general infinite, the *winning region* for Eve, i.e., the set of winning vertices for Eve may not admit a finite presentation. Similarly, for objectives for which finite-memory strategies exists, the question of whether such a strategy can be finitely presented (and computed) is raised. Hence, we will in general distinguish the following three algorithmic problems.

Problem 10 (Solving a pushdown game)**.**

INPUT: *A pushdown game $\mathscr{G}$ and an initial vertex v_0*

OUTPUT: *Does Eve win $\mathscr{G}$ from v_0?*

Problem 11 (Computing the winning region)**.**

INPUT: *A pushdown game $\mathscr{G}$*

OUTPUT: *Output a finite presentation of the set v of vertices from which Eve wins $\mathscr{G}$*

In Theorem 106, we will show that the winning region can be described by a finite-state automaton for a large class of qualitative winning conditions.

Problem 12 (Computing a winning strategy)**.**

INPUT: *A pushdown game $\mathscr{G}$*

OUTPUT: *Output a finite presentation of a strategy for Eve that is winning from any vertex in the winning region for Eve in $\mathscr{G}$*

We will show, for parity pushdown games, that the winning strategy can be described using either a finite-state automaton or a pushdown automaton (see Section 10.3.3).

10.2 Profiles and regularity of the winning regions

In this section, we consider a large class of objectives called *prefix independent*. For these objectives, a pushdown game can be meaningfully decomposed by considering the part of the game between the moment a symbol is pushed onto the stack and stopping as soon as it is popped. As a consequence, we will see that for prefix independent objectives, the winning region can be described using finite state automata.

To this extent, we introduce *reduced games* which start with a stack containing only one symbol γ and stop as soon as this symbol is popped from the stack. If the symbol is never popped, the objective is unchanged and otherwise the winner of the game is determined by the state reached when popping the symbol.

Let $\mathscr{G} = (\mathscr{A}, \Omega)$ be a pushdown game played on an arena $\mathscr{A} = (G, V_{\text{Eve}}, V_{\text{Adam}})$ generated by a pushdown system $\mathscr{P} = (Q, Q_{\text{Eve}}, Q_{\text{Adam}}, \Gamma, \Delta)$. For any subset $R \subseteq Q$ of control states of $\mathscr{P}$, we define a new objective $\Omega(R)$ such that a play π belongs to $\Omega(R)$ if one of the following happens:

- the play π belongs to Ω and does not contain any configuration with an empty stack (i.e., of the form $(q, \bot)$ for some state $q \in Q$),
- the play π contains a configuration with the empty stack and the first such configuration has a state in R.

More formally, letting $V = V_{\text{Eve}} \cup V_{\text{Adam}}$, $V_R = R \times \bot\Gamma^*$ and $V_\bot = Q \times \{\bot\}$

$$\Omega(R) = (\Omega \setminus V^* V_\bot V^\omega) \cup (V \setminus V_\bot)^* V_R V^\omega$$

Finally, we let $\mathscr{G}(R)$ denote the game $(\mathscr{A}, \Omega(R))$.

Remark that contrarily to the rest of the objectives considered in this chapter, $\Omega(R)$ does depend on the sequences of vertices visited by the play and not only on their colour. It would have been possible by a slight modification of the arena to only express the objective on their colours. However, our choice simplifies the presentation of the reduced games.

For any state $q \in Q$ and any stack letter $\gamma \in \Gamma$, we denote by $\mathscr{R}(q, \gamma)$ the set of subsets $R \subseteq Q$ for which Eve wins in $\mathscr{G}(R)$ from $(q, \bot\gamma)$:

$$\mathscr{R}(q, \gamma) = \{R \subseteq Q \mid (q, \bot\gamma) \text{ is winning for Eve in } \mathscr{G}(R)\}$$

and we refer to $\mathscr{R}(q, \gamma)$ as the (q, γ)-*profile* of $\mathscr{G}$.

An objective $\Omega \subseteq C^\omega$ is *prefix independent* if the following holds: for every $u \in C^\omega$ and for every $v \in C^*$, $u \in \Omega$ if and only if $vu \in \Omega$. The Büchi, co-Büchi and parity objectives are examples of prefix independent objectives and the reachability objective is not.

Remark 14. *For a prefix independent objective, a play respecting a winning strategy for Eve that starts in Eve's winning region always stays in this region. Obviously this is no longer true if the objective is not prefix independent. For instance in a reachability game, a play following a winning strategy for Eve can leave the winning region of Eve once the target has been reached.*

This simple property allows to use profiles to give an inductive characterization of the winning region for Eve when the objective is prefix independent.

Proposition 4. *Assume that $\Omega \subseteq C^\omega$ is prefix independent. Let $s \in \Gamma^*$, $q \in Q$ and $\gamma \in \Gamma$. Then Eve has a winning strategy in $\mathscr{G}$ from $(q, \bot s\gamma)$ if and only if there exists some $R \in \mathscr{R}(q,\gamma)$ such that $(r, \bot s)$ is winning for Eve in $\mathscr{G}$ for every $r \in R$.*

Proof. Assume Eve has a winning strategy σ from $(q, \bot s\gamma)$ in $\mathscr{G}$. Consider the set Π_σ of all plays in $\mathscr{G}$ that starts from $(q, \bot s\gamma)$ and where Eve respects σ. Define R to be the (possibly empty) set that consists of all $r \in Q$ such that there is a play in Π_σ of the form $v_0 \cdots v_k(r, \bot s)v_{k+1} \cdots$ where each v_i for $0 \leq i \leq k$ is of the form $(p_i, \bot st_i)$ for some non-empty t_i. In other words, R consists of all states that can be reached on popping γ for the first time in a play where Eve respects σ. As seen in Remark 14, Eve is winning from $(r, \bot s)$ for all $r \in R$. It remains to show that $R \in \mathscr{R}(q,\gamma)$.

To this end, define a (partial) function ξ as $\xi((p, \bot st)) = (p, \bot t)$ for every $p \in Q$ and set $\xi^{-1}((p, \bot t)) = (p, \bot st)$. Then ξ^{-1} is extended as a morphism over V^*. Now a winning strategy for Eve in $\mathscr{G}(R)$ is defined as follows:

- if some empty stack configuration has already been visited play any valid move,
- otherwise go to $\xi(\sigma(\xi^{-1}(\pi))$, where π is the current play.

By definition of Π_σ and R, it easily follows that the previous strategy is winning for Eve in $\mathscr{G}(R)$, and therefore $R \in \mathscr{R}(p,\gamma)$.

Conversely, let us assume that there is some $R \in \mathscr{R}(q,\gamma)$ such that $(r, \bot s)$ is winning for Eve in $\mathscr{G}$ for every $r \in R$. For every $r \in R$, let us denote by σ_r a winning strategy for Eve from $(r, \bot s)$ in $\mathscr{G}$. Let σ_R be a winning strategy for Eve in $\mathscr{G}(R)$ from $(q, \bot\gamma)$. Let us define ξ and ξ^{-1} as in the direct implication and extend them as (partial) morphism over V^*. Define the following strategy σ for Eve in $\mathscr{G}$ for plays starting from $(q, \bot s\gamma)$. For any such play π,

- if π does not contain a configuration of the form $(p, \bot s)$ then we take $\sigma(\pi) = \xi^{-1}(\sigma_R(\xi(\pi)))$;
- otherwise let $\pi = \pi' \cdot (r, \bot s) \cdot \pi''$ where π' does not contain any configuration of the form $(p, \bot s)$. If r does not belong to R, σ is undefined. Note that this situation will never be encountered in a play respecting σ as σ_R ensures that $r \in R$. If $r \in R$, one finally sets $\sigma(\pi) = \sigma_r((r, \bot s)\pi'')$.

The strategy σ is a winning strategy for Eve in $\mathscr{G}$ from $(q, \bot s\gamma)$. To see this, consider a play π starting from $(p, \bot s\gamma)$ and respecting σ.

If the play π does not contain configurations of the form $(r, \bot s)$ for some $r \in Q$, then the play $\xi(\pi)$ starting in $(p, \bot\gamma)$ respects σ_R and is won by Eve. As $\xi(\pi)$ does not contain configurations with an empty stack, it must be the case that $\xi(\pi) \in \Omega$. As Ω only depends on the colours of the states, it is also the case that $\pi \in \Omega$ and hence, π is winning for Eve.

If the play π can be decomposed as $\pi'(r, \bot s)\pi''$ where π' does not contain any configuration with the stack $\bot s$, the play $\xi(\pi'(r, \bot s))$ respects σ_R in $\mathscr{G}(R)$. As σ_R is winning, it follows that $r \in R$. By definition of σ, $(r, \bot s)\pi''$ respects σ_r which being

winning for Eve implies that $(r, \bot s)\pi'' \in \Omega$. As Ω is prefix independent, it follows that $\pi \in \Omega$. □

Proposition 4 implies that the winning region of any pushdown game equipped with a prefix independent objectives can be described by regular languages.

Theorem 106. *Let $\mathcal{G} = (\mathcal{A}, \Omega)$ be a pushdown game played on an arena $\mathcal{A} = (G, V_{\text{Eve}}, V_{\text{Adam}})$ generated by a pushdown system $\mathcal{P} = (Q, Q_{\text{Eve}}, Q_{\text{Adam}}, \Gamma, \Delta)$, and such that $\Omega \subseteq C^\omega$ is prefix independent. Then for any state $q \in Q$, the set*

$$L_q = \{u \in \Gamma^* \mid (q, \bot u) \in W_{\text{Eve}}\}$$

is a regular language over the alphabet Γ.

Proof. Fix a control state $q \in Q$, we consider a deterministic finite state automaton defined as follows. Its set of control states consists of the subsets of Q and the initial state is $S_{in} = \{p \mid (p, \bot) \in W_{\text{Eve}}\}$. From the state S upon reading the letter γ the automaton goes to the state $\{p \mid S \in \mathcal{R}(p, \gamma)\}$. Finally a state S is final if and only if $q \in S$. It is then an immediate consequence of Proposition 4 that this automaton accepts the language L_q. □

Remark 15. *By a slight abuse, we can think of the $|Q|$ automata in Theorem 106 as a single automaton that is design to first read the stack content and finally reads the control state q (in this latter step, from state S it either go to a final state if $q \in S$ or to a rejecting one otherwise).*

Remark 16. *Note that the characterisation in Theorem 106 is* a priori *not effective. Indeed, to construct automata for the languages L_q one needs to be able to compute all the (q, γ)-profiles $\mathcal{R}(q, \gamma)$ of $\mathcal{G}$ and compute the winner from configurations of the form $(q, \bot)$.*

Consider, for instance, the objective over the set of the colours $C = \{0, 1, \#, \$\}$

$$\Omega_{pal} = \{w \in C^\omega \mid w \text{ contains infinitely many factors of the form } \#u\$\tilde{u}\# \text{ with } u \in \{0,1\}^*\}$$

where $\tilde{u}$ denotes the mirror of the word u.

As a language of ω-words, this objective is accepted by a deterministic ω-pushdown automaton with a Büchi acceptance condition. Between two consecutive occurrences of the #-symbol, the automaton checks that the word w appearing in between these two occurrences is of the form $u\$\tilde{u}$ for some word $u \in \{0,1\}^*$. This can be done in a deterministic manner as follows. First the automaton pushes onto the stack a symbol $\bot'$ (which will play the same role as the bottom of stack symbol) then it pushes onto the stack all symbols in $\{0,1\}$ that are read. Then when the first \$-symbol is read, it only allows to read the symbol that is on the top of the stack before popping it. Finally when a #-symbol is read, if the top-most symbol of the stack is $\bot'$ the unique final state is visited. Hence ensuring that a final state is visited if the word w is of the required form. If w contains several \$ symbols or if the symbol read does not correspond to the top of the stack, the automaton enters a non-final state is in which it waits for the next #-symbol.

For games with finite arenas and the Ω_{pal} objective, deciding the winner reduces to deciding the winner in a pushdown game with the Büchi objective which is decidable as we will prove later in this chapter. The pushdown game is essentially a synchronized product between the finite arena and the ω-pushdown automaton described previously.

However the problem of deciding the winner in a pushdown game with the Ω_{pal} objective is undecidable even if all vertices belong to Eve. The undecidability is proved by a reduction from Post correspondance problem (PCP) which is a well-known to be undecidable. Recall that an instance of PCP is a finite sequence $(r_1,\ell_1),\dots,(r_n,\ell_n)$ of pairs of words over $\{0,1\}$. Such an instance is said to admit a solution if there exists a sequence of indices $i_1\cdots i_k \in [1,n]^*$ such that:

$$r_{i_1} r_{i_2} \cdots r_{i_k} = \ell_{i_1}\ell_{i_2}\cdots\ell_{i_k}.$$

The PCP problem is, given an instance, to decide if it admits a solution.

For an instance $I = (\ell_1,r_1),\dots,(\ell_n,r_n)$ of PCP, we construct a pushdown game G_I with the objective Ω_{pal} such that Eve wins G_I from $(p_\star,\bot)$ if and only if I admits a solution. In this game, Eve plays alone and the play is decomposed in two phases that will repeat:

- *in phase 1, Eve can push any index i in $[1,n]$ while producing the sequence of colors r_i. As soon as at least one index has been pushed, she can also choose to move to the sequence phase while producing the colour* \$.
- *in phase 2, Eve must (until the bottom of stack symbol is reached) pop the top most element of the stack i while producing the sequence of colours $\tilde{\ell}_i$. When the bottom of stack symbol is encountered, Eve goes back to the first phase while producing the colour* #.

If I has a solution $i_1\cdots i_k$ then the strategy in which Eve always pushes this sequence in phase 1 is winning for her. As $i_1\cdots i_k$ is a solution of I, we have $u = r_{i_1}\cdot r_{i_k} = \ell_{i_1}\cdots\ell_{i_k}$ and by construction of the game, the sequence of colors associated with the play is $(u\$\tilde{u})^\omega \in \Omega_{pal}$. Conversely if Eve has a winning strategy from $(q_\star,\bot)$ then the sequence of colours associated with the winning play belongs to Ω_{pal}. In particular, it must contain a factor of the form $\#u\$\tilde{u}\#$. By construction of the game the sequence of indices $i_1\cdots i_k$ pushed while producing u is a solution of I.

10.2.1 Reachability Pushdown Game

We will see in the following section that for the parity condition and more generally for any ω-regular winning conditions the (q,γ)-profiles can be computed for any $q\in Q$ and $\gamma\in\Gamma$. We start by the simpler case of the reachability objective.

At first sight, the reachability objective is not captured by Theorem 106 as it is not prefix independent. However with a slight adaptation of the reduced game $\mathscr{G}(R)$ Theorem 106 for the reachability condition. Intuitively we ask that the play stops as soon as a target vertex is reached.

More formally, for a reachability objective $\texttt{Reach}(F)$ and letting $V = V_{\text{Eve}}\cup V_{\text{Adam}}$ and $V_F = \{v\in V \mid \mathfrak{c}(v)\in F\}$:

$$\overline{\Omega}(R) = [(V\setminus Q\times\{\bot\})^*\cdot V_F\cdot V^\omega]\cup[(V\setminus(Q\times\{\bot\}\cup V_F))^*\cdot(R\times\{\bot\})\cdot V^\omega]$$

It is easily shown that with this modification to the definition of profiles both Proposition 4 and Theorem 106 are true for the reachability objective. In the special case of the reachability objective, the set Profs of triples (p,γ,R) such that $R \in \mathscr{R}(p,\gamma)$ can be expressed as a smallest fixed point.

More precisely, Profs is the smallest subset of $Q \times \Gamma \times \mathscr{P}(Q)$ such that for $p \in Q$, $\gamma \in \Gamma$ and $R \subseteq Q$, (p,γ,R) belongs Profs if either:

1. $p \in Q_F = \{q \in Q \mid \mathfrak{c}(q) \in F\}$,
2. or $p \in Q_{\text{Eve}}$ and for some $q \in Q$,
   - either $(q,\text{pop}) \in \Delta(p,\gamma)$ and $q \in R$,
   - or $(q,\text{push}(\gamma')) \in \Delta(p,\gamma)$ and $(q,\gamma',R') \in$ Profs for some $R' \subseteq Q$ such that for all $p' \in R'$, $(p',\gamma,R) \in$ Profs.
3. or $p \in Q_A$ and for all $q \in Q$ the following hold:
   - $(q,\text{pop}) \in \Delta(p,\gamma)$ implies $q \in R$,
   - $(q,\text{push}(\gamma')) \in \Delta(p,\gamma)$ implies that there exists $R' \subseteq Q$ such that $(q,\gamma',R') \in$ Profs and for all $p' \in R'$, $(p',\gamma,R) \in$ Profs.

Using this characterization, the set Profs can be computed using the standard method for computing small-fixed point of a monotonic function by computing the sequence of approximants $\text{Profs}_0 = \emptyset \subseteq \text{Profs}_1 \subseteq \text{Profs}_2 \cdots$ until it stabilizes. More precisely, for all $i \geq 0$, Profs_{i+1} is obtained by adding to Profs_i all the tuples that can be inferred using the properties (1), (2) and (3) above applied to Profs_i. As at most $|Q| \cdot |\Gamma| \cdot 2^{|Q|}$ tuples can be added, the sequence must stabilize in at most $|Q| \cdot |\Gamma| \cdot 2^{|Q|}$ steps. As the computation of Profs_{i+1} from Profs_i can be performed in polynomial time, the profils in a reachability pushdown game can be computed in time $p(|Q| \cdot |\Gamma| \cdot 2^{|Q|})$ for some polynomial p.

Remark 17. *In the case where Eve plays alone (i.e., $Q = Q_{\text{Eve}}$), there is only on play respecting a fixed strategy for Eve and as a result, one only need to compute profils of the form (p,γ,R) with $|R| \leq 1$. In this setting, the fixed point characterization yields a polynomial time algorithm to compute the set of profiles.*

10.3 Parity pushdown games

We now focus on the central case of parity objectives. In Section 10.3.1, we show how to compute the set of profiles using a reduction to finite parity game.

For the rest of this section, we fix a parity pushdown game $\mathscr{G}$ played on an arena $\mathscr{A} = (G, V_{\text{Eve}}, V_{\text{Adam}})$ generated by a pushdown system $\mathscr{P} = (Q, Q_{\text{Eve}}, Q_{\text{Adam}}, \Gamma, \Delta)$. We also let $V = V_{\text{Eve}} \cup V_{\text{Adam}}$ and we let the colours used in the game be $\{0,\ldots,d\}$.

10.3.1 Computing the Profiles

In this section we show how to build a parity game played on a *finite* arena that permits to compute the profiles in $\mathscr{G}$.

We first start with some terminology and a basic result. For an infinite play $\pi = v_0 v_1 \cdots$, let $Steps_\pi$ be the set of indices of positions where no configuration of strictly smaller stack height is visited later in the play. More formally,

$$Steps_\pi = \{i \in \mathbb{N} \mid \forall j \geq i, \mathrm{sh}(v_j) \geq \mathrm{sh}(v_i)\}.$$

Note that $Steps_\pi$ is always infinite and hence induces a factorisation of the play π into finite pieces.

In the factorisation induced by $Steps_\pi$, a factor $v_i \cdots v_j$ is called a *bump* if $\mathrm{sh}(v_j) = \mathrm{sh}(v_i)$, called a *Stair* otherwise (that is, if $\mathrm{sh}(v_j) = \mathrm{sh}(v_i) + 1$).

For any play π with $Steps_\pi = \{n_0 < n_1 < \cdots\}$, we can define the sequence $(\mathfrak{c}(\pi)_i)_{i\geq 0} \in \{0, \ldots, d\}^{\mathbb{N}}$ by setting $\mathfrak{c}(\pi)_i = \max\{\mathfrak{c}(v_k) \mid n_i \leq k \leq n_{i+1}\}$. This sequence fully characterises the parity objective.

Proposition 5. *Let π be a play. Then π satisfies the parity condition if and only if* $\limsup((\mathfrak{c}(\pi)_i)_{i\geq 0})$ *is even.*

Simulation Game

In the sequel, we build a new parity game $\widetilde{\mathscr{G}}$ over a *finite* arena $\widetilde{\mathscr{A}}$. This new game *simulates* the original pushdown game, in the sense that the sequence of visited colours during a correct simulation of some play π in $\mathscr{G}$ is exactly the sequence $(\mathfrak{c}(\pi)_i)_{i\geq 0}$. Moreover, a play in which a player does not correctly simulate the pushdown game is losing for that player. We shall see that the winning region in $\widetilde{\mathscr{G}}$ allows us to compute the set of profiles $\{\mathscr{R}(q,\gamma) \mid q \in Q \text{ and } \gamma \in \Gamma\}$. Hence, by Theorem 106, it will imply that one can solve a pushdown game as well as compute its winning region.

Before providing a precise description of the arena $\widetilde{\mathscr{A}}$, let us consider the following informal description of this simulation game. We aim at simulating a play in the pushdown game from some initial vertex $(p_{in}, \bot)$. In $\widetilde{\mathscr{A}}$ we keep track of only the control state and the top stack symbol of the currently simulated configuration.

The interesting case is when it is in a control state p with top stack symbol α, and the player owning p wants to push a symbol β onto the stack and change the control state to q. For every strategy of Eve, there is a certain set of possible (finite) continuations of the play that will end with popping β from the stack. We require Eve to declare a vector $\overrightarrow{S} = (S_0, \ldots, S_d)$ of $(d+1)$ subsets of Q, where S_i is the set of all states the game can be in after popping β along those plays where in addition the largest visited colour while β was on the stack is i.

Adam has then two choices. He can continue the game by pushing β onto the stack and updating the state (we call this a *pursue* move). Otherwise, he can pick a set S_i and a state $r \in S_i$, and continue the simulation from that state r (we call this a *jump* move). If he does a pursue move, then he remembers the vector $\overrightarrow{S}$ claimed by Eve; if later on, a pop transition is simulated, the play goes in a sink vertex and Evewins if and only if the resulting state is in S_c where c is the largest colour seen in the current stack level

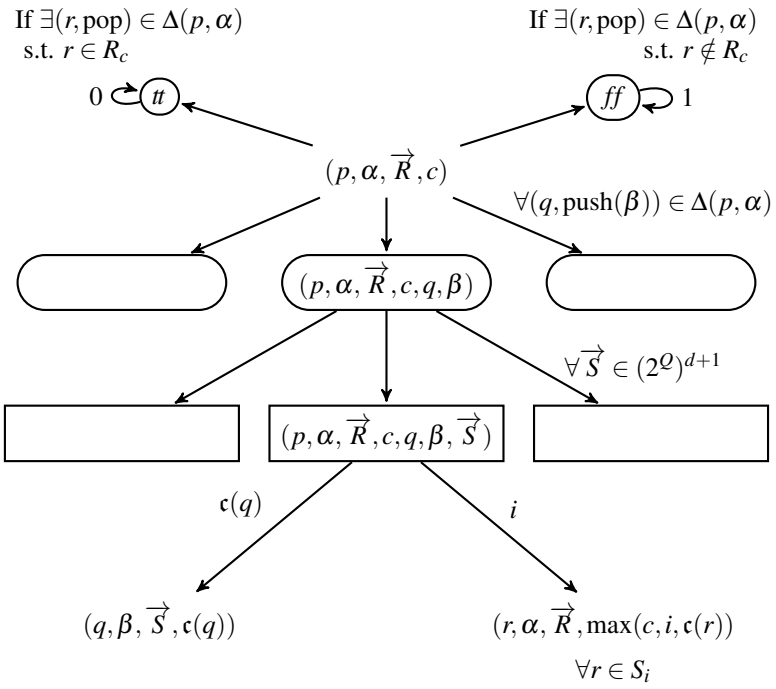


Figure 10.2: Local structure of the arena $\widetilde{\mathscr{A}}$

(this information is encoded in the vertex, reset after each pursue move and updated after each jump move). If Adamdoes a jump move to a state r in S_i, the currently stored value for c is updated to $\max(c,i,\mathfrak{c}(r))$, which is the largest colour seen since the current stack level was reached.

Therefore the main vertices of this new arena are of the form $(p,\alpha,\overrightarrow{R},c)$, which are controlled by the player who controls p. Intermediate vertices are used to handle the previously described intermediate steps. The local structure is given in Figure 10.2. Two special sink vertices *tt* and *ff* are used to simulate pop moves. This arena is equipped with a colouring function on the edges: an edge from a vertex $(p,\alpha,\overrightarrow{R},c,q,\beta,\overrightarrow{S})$ to a vertex $(r,\alpha,\overrightarrow{R},\max(c,i,\mathfrak{c}(r))$ has colour i where i is the colour of the simulated bump, an edge from a vertex $(p,\alpha,\overrightarrow{R},c,q,\beta,\overrightarrow{S})$ to a vertex $(q,\beta,\overrightarrow{S}\,\mathfrak{c}(q))$ simulating a jump move has colour $\mathfrak{c}(q)$, the loop on *tt* has colour 0 while the loop on *ff* has colour 1; all other edges get the irrelevant colour 0.

We now formally describe arena $\widetilde{\mathscr{A}}$ (we refer to Figure 10.2) and provide some extra insight.

- The main vertices of $\widetilde{\mathscr{A}}$ are those of the form $(p,\alpha,\overrightarrow{R},c)$, where $p \in Q$, $\alpha \in \Gamma$, $\overrightarrow{R} = (R_0,\ldots,R_d) \in (2^Q)^{d+1}$ and $c \in \{0,\ldots,d\}$. A vertex $(p,\alpha,\overrightarrow{R},c)$ is reached when simulating a finite play π in $\mathscr{G}$ such that:

  - The last vertex in π is $(p, \bot s\alpha)$ for some $s \in \Gamma^*$.
  - Eveclaims that she has a strategy to continue π in such a way that if α is eventually popped, the control state reached after popping belongs to R_m, where m is the largest colour visited since the stack height was at least $|s\alpha|$.
  - The colour c is the largest one since the current stack level was reached from a lower stack level.

  A vertex $(p, \alpha, \overrightarrow{R}, c)$ is controlled by Eve if and only if $p \in Q_{\text{Eve}}$.

- The vertices tt and ff are here to ensure that the vectors $\overrightarrow{R}$ encoded in the main vertices are correct. They are both controlled by Eve and are sink vertices with a self loop with colour 0 for tt and 1 for ff.

  There is a transition from some vertex $(p, \alpha, \overrightarrow{R}, c)$ to tt, if and only if there exists a transition rule $(r, pop) \in \Delta(p, \alpha)$, such that $r \in R_c$ (this means that $\overrightarrow{R}$ is correct with respect to this transition rule). Dually, there is a transition from a vertex $(p, \alpha, \overrightarrow{R}, c)$ to ff if and only if there exists a transition rule $(r, pop) \in \Delta(p, \alpha)$ such that $r \notin R_c$ (this means that $\overrightarrow{R}$ is not correct with respect to this transition rule).

- To simulate a transition rule $(q, push(\beta)) \in \Delta(p, \alpha)$, the player that controls $(p, \alpha, \overrightarrow{R}, c)$ moves to $(p, \alpha, \overrightarrow{R}, c, q, \beta)$. This vertex is controlled by Eve who has to give a vector $\overrightarrow{S} = (S_0, \ldots, S_d) \in (2^Q)^{d+1}$ that describes the control states that can be reached if β is eventually popped. To describe this vector, she goes to the corresponding vertex $(p, \alpha, \overrightarrow{R}, c, q, \beta, \overrightarrow{S})$.

  Any vertex $(p, \alpha, \overrightarrow{R}, c, q, \beta, \overrightarrow{S})$ is controlled by Adam who chooses either to simulate a bump or a stair. In the first case, he additionally has to pick the maximal colour of the bump. To simulate a bump with maximal colour i, he goes, through an edge coloured by i, to a vertex $(r, \alpha, \overrightarrow{R}, \max(c, i, \mathfrak{c}(r)))$, for some $r \in S_i$.

  To simulate a stair, Adam goes, through an edge coloured by $\mathfrak{c}(q)$, to the vertex $(q, \beta, \overrightarrow{S}, \mathfrak{c}(q))$.

  The last component of the vertex (that stores the largest colour seen since the currently simulated stack level was reached) has to be updated in all those cases. After simulating a bump of maximal colour i, the maximal colour is $\max(c, i, \mathfrak{c}(r))$. After simulating a stair, this colour has to be initialized (since a new stack level is simulated). Its value, is therefore $\mathfrak{c}(q)$, which is the unique colour since the (new) stack level was reached.

The edges for which we did not precise the colour are assigned colour 0.

The following theorem relates this new game $\widetilde{\mathscr{G}}$ and the profiles in the pushdown game $\mathscr{G}$.

Theorem 107. *The following holds.*

(i) *A configuration* $(p_{in}, \bot)$ *is winning for Eve in* $\mathscr{G}$ *if and only if* $(p_{in}, \bot, (\emptyset, \dots, \emptyset), \mathfrak{c}(p_{in}))$ *is winning for Eve in* $\widetilde{\mathscr{G}}$.

(ii) *For every* $q \in Q$, $\gamma \in \Gamma$ *and* $R \subseteq Q$, $R \in \mathscr{R}(q,\gamma)$ *if and only if* $(q, \gamma, (R, \dots, R), \mathfrak{c}(q))$ *is winning for Eve in* $\widetilde{\mathscr{G}}$.

The rest of the section is devoted to the proof of Theorem 107. We only prove point (i) as the proof of point (ii) is a subpart of the proof of (i).

Factorisation of a play in $\mathscr{G}$.

Recall that, for an infinite play $\pi = v_0 v_1 \cdots$ in $\mathscr{G}$, $Steps_\pi$ denotes the set of indices of positions where no configuration of strictly smaller stack height is visited later in the play. Note that $Steps_\pi$ is always infinite and hence induces a factorisation of the play π into finite pieces.

Indeed, for any play π with $Steps_\pi = \{n_0 < n_1 < \cdots\}$, one can define the sequence $(\pi_i)_{i \geq 0}$ by setting $\pi_i = v_{n_i} \cdots v_{n_{i+1}}$. Note that each of the Λ_i is either a bump or a stair. We designate $(\pi_i)_{i \geq 0}$ as the *rounds factorisation* of π and we let $\mathfrak{c}(\pi_i)$ denotes the largest colour in π_i.

Factorisation of a play in $\widetilde{\mathscr{G}}$.

Recall that in $\widetilde{\mathscr{A}}$ only some edges have a relevant colour while all others get colour 0. Hence, to represent a play, we only keep the relevant colours of edges. More precisely, we only need to encode the colours in $\{0, \dots, d\}$ that appears when simulating a bump: a play will be represented as a sequence of vertices together with colours in $\{0, \dots, d\}$ that correspond to (relevant) colours appearing on edges.

For any play in $\widetilde{\mathscr{G}}$, a *round* is a factor between two visits through vertices of the form $(p, \alpha, \overrightarrow{R}, c)$. We have the following possible forms for a round:

- The round is of the form

$$(p, \alpha, \overrightarrow{R}, c)(p, \alpha, \overrightarrow{R}, c, q, \beta)(p, \alpha, \overrightarrow{R}, c, q, \beta, \overrightarrow{S})i(r, \alpha, \overrightarrow{R}, \max(c, i, \mathfrak{c}(s)))$$

  and corresponds therefore to the simulation of a rule pushing β followed by a sequence of moves that ends by popping β. Moreover i is the largest colour encountered while β was on the stack.

- The round is of the form

$$(p, \alpha, \overrightarrow{R}, c)(p, \alpha, \overrightarrow{R}, c, q, \beta)(p, \alpha, \overrightarrow{R}, c, q, \beta, \overrightarrow{S})\mathfrak{c}(q)(q, \beta, \overrightarrow{S}, \mathfrak{c}(q))$$

  and corresponds therefore to the simulation of a rule pushing a symbol β leading to a new stack level below which the play will never go. We designate it as a *stair*.

For any play $\widetilde{\pi} = v_0 v_1 v_2 \cdots$ in $\widetilde{\mathscr{G}}$, we consider the subset of indices corresponding to vertices of the form $(p, \alpha, \overrightarrow{R}, c)$. More precisely:

$$Rounds_{\widetilde{\pi}} = \{n \mid v_n = (p, \alpha, \overrightarrow{R}, c),\ p \in Q,\ \alpha \in \Gamma, \overrightarrow{R} \in (2^Q)^{d+1},\ 0 \leq c \leq d\}$$

Therefore, the set $Rounds_{\widetilde{\pi}}$ induces a natural factorisation of $\widetilde{\pi}$ into rounds.

Definition 29 (Rounds factorisation)**.** *For a (possibly finite) play $\widetilde{\pi} = v_0 v_1 v_2 \cdots$, we call* rounds factorisation *of $\widetilde{\pi}$, the (possibly finite) sequence $(\widetilde{\pi}_i)_{i \geq 0}$ of rounds defined as follows. Let $Rounds_{\widetilde{\pi}} = \{n_0 < n_1 < n_2 < \cdots\}$, then for all $0 \leq i < |Rounds_{\widetilde{\pi}}|$, define $\widetilde{\pi}_i = v_{n_i} \cdots v_{n_{i+1}}$.*

Therefore, for every $i \geq 0$, the first vertex in $\widetilde{\pi}_{i+1}$ equals the last one in $\widetilde{\pi}_i$. Moreover, $\widetilde{\pi} = \widetilde{\pi}_1 \odot \widetilde{\pi}_2 \odot \widetilde{\pi}_3 \odot \cdots$, where $\widetilde{\pi}_i \odot \widetilde{\pi}_{i+1}$ denotes the concatenation of $\widetilde{\pi}_i$ with $\widetilde{\pi}_{i+1}$ without its first vertex. Finally, the colour *of a round is the unique colour in $\{0, \ldots, d\}$ appearing in the round.*

In order to prove both implications of Theorem 107 , we build from a winning strategy for Eve in one game a winning strategy for her in the other game. The main argument to prove that the new strategy is winning is to prove a correspondence between the factorisations of plays in both games.

Proof of the Direct Implication of Theorem 107

Assume that the configuration $(p_{in}, \bot)$ is winning for Eve in $\mathscr{G}$, and let σ be a corresponding winning strategy for her.

Using σ, we define a strategy $\widetilde{\sigma}$ for Eve in $\widetilde{\mathscr{G}}$ from $(p_{in}, \bot, (\emptyset, \ldots, \emptyset), \mathfrak{c}(p_{in}))$. This strategy stores a finite play in $\mathscr{G}$, that is an element in V^*. This memory will be denoted π. At the beginning π is initialized to the vertex $(p_{in}, \bot)$. We first describe $\widetilde{\sigma}$, and then we explain how π is updated. Both the strategy $\widetilde{\sigma}$ and the update of π, are described for a round.

Choice of the move. Assume that the play is in some vertex $(p, \alpha, \overrightarrow{R}, c)$ for $p \in Q_{\text{Eve}}$. The move given by $\widetilde{\sigma}$ depends on $\sigma(\pi)$:

- If $\sigma(\pi) = (r, pop)$, then Eve goes to tt (Proposition Proposition 6 will prove that this move is always possible).
- If $\sigma(\pi) = (q, push(\beta))$, then Eve goes to $(p, \alpha, \overrightarrow{R}, c, q, \beta)$.

In this last case, or in the case where $p \in Q_{\text{Adam}}$ and Adam goes to $(p, \alpha, \overrightarrow{R}, c, q, \beta)$, we also have to explain how Eve behaves from $(p, \alpha, \overrightarrow{R}, c, q, \beta)$. She has to provide a vector $\overrightarrow{S} \in (2^Q)^{d+1}$ that describes which states can be reached if β is eventually popped, depending on the largest colour visited in the meantime. In order to define $\overrightarrow{S}$, Eve considers the set of all possible continuations of $\pi \cdot (q, s\alpha\beta)$ (where $(p, s\alpha)$ denotes the last vertex of π) where she respects her strategy σ. For each such play, she checks whether some configuration of the form $(r, s\alpha)$ is visited after $\pi \cdot (q, s\alpha\beta)$, that is if the stack level of β is eventually left. If it is the case, she considers the first configuration $(r, s\alpha)$ appearing after $\pi \cdot (q, \sigma\alpha\beta)$ and the largest colour i since β was

on the stack. For every $i \in \{0,\dots d\}$, S_i, is exactly the set of states $r \in Q$ such that the preceding case happens. More formally,

$$S_i = \{r \mid \exists\, \pi \cdot (q, s\alpha\beta) v_0 \cdots v_k (r, s\alpha) \cdots \text{ play in } \mathscr{G} \text{ where Eve respects } \sigma \text{ and s.t.} \\ \mathrm{sh}(v_j) > |\sigma\alpha|,\ \forall j = 0,\dots,k, \text{ and } \max(\{\mathfrak{c}(v_j) \mid j = 0,\dots,k\} \cup \{\mathfrak{c}(q)\}) = i\}$$

Finally, we set $\overrightarrow{S} = (S_0,\dots,S_d)$ and Eve moves to $(p, \alpha, \overrightarrow{R}, c, q, \beta, \overrightarrow{S})$.

Update of π. The memory π is updated after each visit to a vertex of the form $(p, \alpha, \overrightarrow{R}, c)$. We have two cases depending on the kind of the last round:

- The round is a bump, and therefore a bump of colour i (where i is the colour of the round) starting with some transition $(q, \mathit{push}(\beta))$ and ending in a state $r \in S_i$ was simulated. Let $(p, s\alpha)$ be the last vertex in π. Then the memory becomes π extended by $(q, s\alpha\beta)$ followed by a sequence of moves, where Eve respects σ, that ends by popping β and reach $(r, s\alpha)$ while having i as largest colour. By definition of S_i such a sequence of moves always exists.

- The round is a stair and therefore we have simulated a transition $(q, \mathit{push}(\beta))$. If $(p, s\alpha)$ denotes the last vertex in π, then the updated memory is $\pi \cdot (q, s\alpha\beta)$.

Therefore, with any finite play $\widetilde{\pi}$ in $\widetilde{\mathscr{G}}$ in which Eve respects her strategy $\widetilde{\sigma}$, is associated a finite play π in $\mathscr{G}$. An immediate induction shows that Eve respects σ in π. The same arguments works for an infinite play $\widetilde{\pi}$, and the corresponding play π is therefore infinite, starts from $(p_{in}, \bot)$ and Eve respects σ in that play. Therefore it is a winning play.

The following proposition is a direct consequence of how $\widetilde{\sigma}$ was defined.

Proposition 6. *Let $\widetilde{\pi}$ be a finite play in $\widetilde{\mathscr{G}}$ that starts from $(p_{in}, \bot, (\emptyset,\dots,\emptyset), \mathfrak{c}(p_{in}))$, ends in a vertex of the form $(p, \alpha, \overrightarrow{R}, c)$, and where Eve respects $\widetilde{\sigma}$. Let π be the play associated with $\widetilde{\pi}$ built by the strategy $\widetilde{\sigma}$. Then the following holds:*

1. *π ends in a vertex of the form $(p, s\alpha)$ for some $s \in \Gamma^*$.*
2. *c is the largest colour visited in π since α was pushed.*
3. *Assume that π is extended, that Eve keeps respecting σ and that the next move after $(p, \sigma\alpha)$ is to some vertex (r, σ). Then $r \in R_c$.*

Proposition Proposition 6 implies that the strategy $\widetilde{\sigma}$ is well defined when it provides a move to tt. Moreover, one can deduce that, if Eve respects $\widetilde{\sigma}$, ff is never reached.

For plays that do not visits tt nor ff, using the definitions of $\widetilde{\mathscr{A}}$ and $\widetilde{\sigma}$, we easily deduce the following proposition.

Proposition 7. *Let $\widetilde{\pi}$ be an infinite play in $\widetilde{\mathscr{G}}$ that starts from $(p_{in}, \bot, (\emptyset,\dots,\emptyset), \mathfrak{c}(p_{in}))$, and where Eve respects $\widetilde{\sigma}$. Let π be the associated play built by the strategy $\widetilde{\sigma}$, and let $(\pi_i)_{i\geq 0}$ be its rounds factorisation. Let $(\widetilde{\pi}_i)_{i\geq 0}$ be the rounds factorisation of $\widetilde{\pi}$. Then, for every $i \geq 1$ the following hold:*

1. *$\widetilde{\pi}_i$ is a bump if and only if π_i is a bump*

2. *$\widetilde{\pi}_i$ has colour $\mathfrak{c}(\pi_i)$.*

Proposition 7 implies that for any infinite play $\widetilde{\pi}$ in $\widetilde{\mathscr{G}}$ starting from $(p_{in}, \bot, (\emptyset, \ldots, \emptyset), \mathfrak{c}(p_{in}))$ where Eve respects $\widetilde{\sigma}$, the sequence of visited colours in $\widetilde{\pi}$ is $(\mathfrak{c}(\pi)_i)_{i\geq 0}$ for the corresponding play π in $\mathscr{G}$. Hence, using Proposition 5 we conclude that $\widetilde{\pi}$ is winning if and only if π is winning. As π is winning for Eve, it follows that $\widetilde{\pi}$ is also winning for her.

Proof of the Converse Implication of Theorem 107

Note that in order to prove the converse implication of Theorem 107 one could follow the direct implication and consider the point of view of Adam. Nevertheless the proof we give here starts from a winning strategy for Eve in $\widetilde{\mathscr{G}}$ and deduces a strategy for her in $\mathscr{G}$: this induces a more involved proof but has the advantage to lead to an effective construction of a winning strategy for Eve in $\mathscr{G}$ if one has an effective strategy for her in $\widetilde{\mathscr{G}}$

Assume now that Eve has a winning strategy $\widetilde{\sigma}$ in $\widetilde{\mathscr{G}}$ from $(p_{in}, \bot, (\emptyset, \ldots, \emptyset), \mathfrak{c}(p_{in}))$. Using $\widetilde{\sigma}$, we build a strategy σ for Eve in $\mathscr{G}$ for plays starting from $(p_{in}, \bot)$.

The strategy σ uses, as memory, a stack Π, to store the complete description of a play in $\widetilde{\mathscr{G}}$. Recall here that a play in $\widetilde{\mathscr{G}}$ is represented as a sequence of vertices together with colours in $\{0, \ldots d\}$. Up to coding we can assume that we distinguish for free between stairs and bumps for transitions from vertices of the form $(p, \alpha, \overrightarrow{R}, c, q, \beta, \overrightarrow{S})$.

The stack alphabet of Π is the set of vertices of $\widetilde{\mathscr{A}}$ together with the colours $\{0, \ldots, d\}$. In the following, $top(\Pi)$ will denote the top stack symbol of Π while $StCont(\Pi)$ will be the word obtained by reading Π from bottom to top (without considering the bottom-of-stack symbol of Π). In any play where Eve respects σ, $StCont(\Pi)$ will be a play in $\widetilde{\mathscr{G}}$ that starts from $(p_{in}, \bot, (\emptyset, \ldots, \emptyset), \mathfrak{c}(p_{in}))$ and where Eve respects her winning strategy $\widetilde{\sigma}$. Moreover, for any play π where Eve respects σ, we will always have that $top(\Pi) = (p, \alpha, \overrightarrow{R}, c)$ if and only if the current configuration in π is of the form $(p, s\alpha)$. Finally, if Eve keeps respecting σ, and if α is eventually popped the configuration reached will be of the form (r, s) for some $r \in R_i$, where i is the largest visited colour since α was on the stack. Initially, Π only contains $(p_{in}, \bot, (\emptyset, \ldots, \emptyset), \mathfrak{c}(p_{in}))$.

In order to describe σ, we assume that we are in some configuration $(p, s\alpha)$ and that $top(\Pi) = (p, \alpha, \overrightarrow{R}, c)$. We first describe how Eve plays if $p \in Q_{\text{Eve}}$, and then we explain how the stack is updated.

- **Choice of the move.** Assume that $p \in Q_{\text{Eve}}$ and that Eve has to play from some vertex $(p, s\alpha)$. For this, she considers the value of $\widetilde{\sigma}$ on $StCont(\Pi)$.

  If it is a move to tt, Eve plays a transition (r, pop) for some state $r \in R_c$. Lemma 130 will prove that such an r always exists.

  If the move given by $\widetilde{\sigma}$ is to go to some vertex $(p, \alpha, \overrightarrow{R}, c, q, \beta)$, then Eve applies the transition $(q, push(\beta))$.

- **Update of Π.** Assume that the last move, played by Eve or Adam, was to go from $(p, s\alpha)$ to some configuration (r, s). The update of Π is illustrated by Figure 10.3 and explained in what follows. Eve pops in Π until she finds some configuration of the form $(p', \alpha', \overrightarrow{R'}, c', p'', \alpha, \overrightarrow{R})$ that is part of a stair. This configuration is therefore in the stair that simulates the pushing of α onto the stack. Eve updates Π by pushing c in Π followed by $(r, \alpha', \overrightarrow{R'}, \max(c', c, \mathfrak{c}(r)))$.

  Assume that the last move, played by Eve or Adam, was to go from $(p, s\alpha)$ to some configuration $(q, s\alpha\beta)$, and let $(p, \alpha, \overrightarrow{R}, c, q, \beta, \overrightarrow{S}) = \widetilde{\sigma}(StCont(\Pi) \cdot (p, \alpha, \overrightarrow{R}, c, q, \beta))$. Intuitively, $\overrightarrow{S}$ describes which states Eve can force a play to reach if β is eventually popped. Eve updates Π by successively pushing $(p, \alpha, \overrightarrow{R}, c, q, \beta)$, $(p, \alpha, \overrightarrow{R}, c, q, \beta, \overrightarrow{S})$, and $(q, \beta, \overrightarrow{S}, \mathfrak{c}(q))$.

The following lemma gives the meaning of the information stored in Π.

Lemma 130. *Let π be a finite play in $\mathscr{G}$, where Eve respects σ, that starts from $(p_{in}, \bot)$ and that ends in a configuration $(p, s\alpha)$. We have the following facts:*

1. *$top(\Pi) = (p, \alpha, \overrightarrow{R}, c)$ with $\overrightarrow{R} \in (2^Q)^{d+1}$ and $0 \leq c \leq d$.*
2. *$StCont(\Pi)$ is a finite play in $\widetilde{\mathscr{G}}$ that starts from $(p_{in}, \bot, (\emptyset, \dots, \emptyset), \mathfrak{c}(p_{in}))$, that ends with $(p, \alpha, \overrightarrow{R}, c)$ and where Eve respects $\widetilde{\sigma}$.*
3. *c is the largest colour visited since α was pushed.*
4. *If π is extended by some move that pops α, the configuration (r, s) that is reached is such that $r \in R_c$.*

Proof. The proof goes by induction on π. We first show that the last point is a consequence of the second and third points. To aid readability, one can refer to Figure 10.3. Assume that the next move after $(p, s\alpha)$ is to apply a transition $(r, pop) \in \Delta(p, \alpha)$. The second point implies that $(p, \alpha, \overrightarrow{R}, c)$ is winning for Eve in $\widetilde{\mathscr{G}}$. If $p \in Q_{\text{Eve}}$, by definition of σ, there is some edge from that vertex to tt, which means that $r \in R_c$ and allows us to conclude. If $p \in Q_{\text{Adam}}$, note that there is no edge from $(p, \alpha, \overrightarrow{R}, c)$ (winning position for Eve) to the (losing) vertex ff. Hence we conclude in the same way.

Let us now prove the other points. For this, assume that the result is proved for some play π, and let π' be an extension of π. We have two cases, depending on how π' extends π:

- π' is obtained by applying a push transition. The result is trivial in that case.
- π' is obtained by applying a pop transition. Let $(p, s\alpha)$ be the last configuration in π, and let $\overrightarrow{R}$ be the last vector component in $top(\Pi)$ when in configuration $(p, s\alpha)$. By the induction hypothesis, it follows that $\pi' = \pi \cdot (r, s)$ with $r \in R_c$. Considering how Π is updated, and using the fourth point, we easily deduce that the new strategy stack Π is as desired (one can have a look at Figure 10.3 for more intuition).

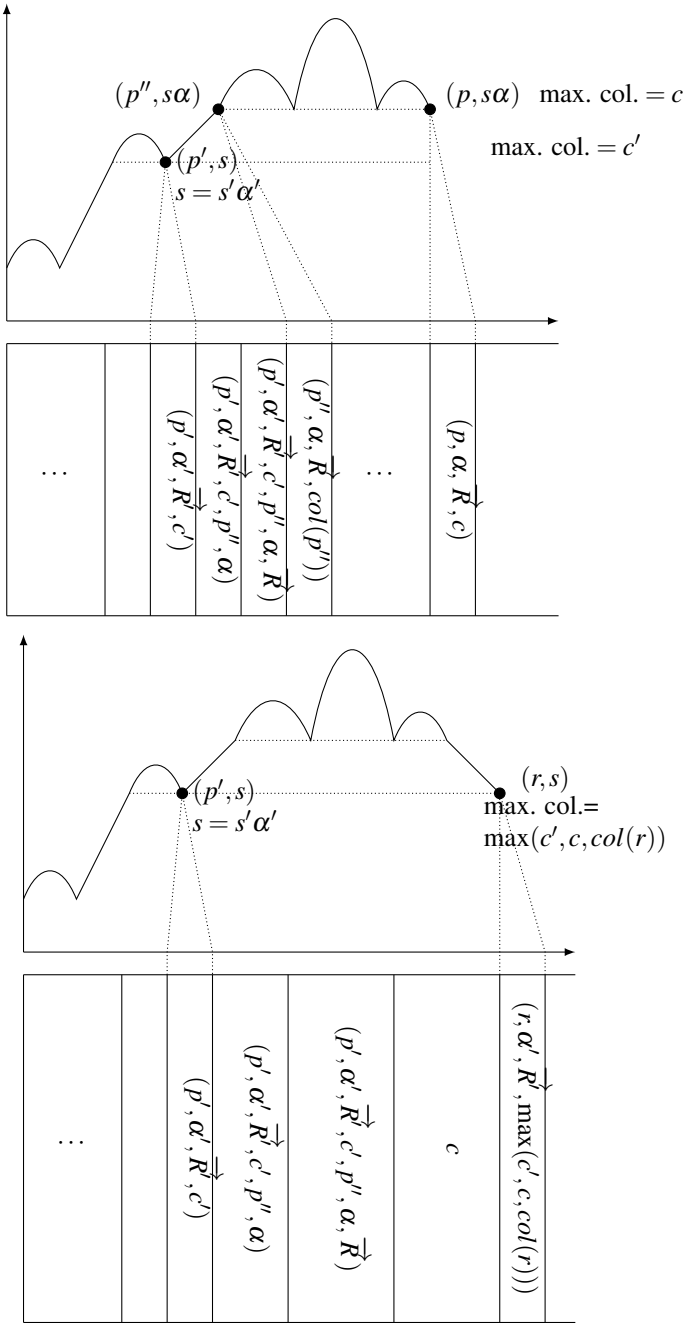


Figure 10.3: Updating the strategy's stack Π

□

Actually, we easily deduce a more precise result.

Lemma 131. *Let π be a finite play in $\mathscr{G}$ starting from $(p_{in}, \bot)$ and where Eve respects σ. Let $(\pi_i)_{i\geq 0}$ be its rounds factorisation. Let $\pi = StCont(\Pi)$, where Π denotes the strategy's stack in the last vertex of π. Let $(\pi_i)_{i=0,\dots,k}$ be the rounds factorisation of π. Then the following holds:*

- *π_i is a bump if and only if π_i is a bump.*
- *π_i has colour $\mathfrak{c}(\pi)_i$.*

Both Lemma 130 and Lemma 131 are for finite plays. A version for infinite plays would allow us to conclude. Let π be an infinite play in $\mathscr{G}$. We define an infinite version of π by considering the limit of the stack contents $(StCont(\Pi_i))_{i\geq 0}$ where Π_i is the strategy's stack after the first i moves in π. It is easily seen that such a limit always exists, is infinite and corresponds to a play won by Eve in $\widetilde{\mathscr{G}}$. Moreover the results of Lemma 131 apply.

Let π be a play in $\mathscr{G}$ with initial vertex $(p_{in}, \bot)$, and where Eve respects σ, and let π be the associated infinite play in $\widetilde{\mathscr{G}}$. Therefore π is won by Eve. Using Lemma 131 and Proposition 5, we conclude, as in the direct implication that π is winning.

10.3.2 Solving the Game and Computing the Winning Region and Strategy

Combining Theorem 107 and Theorem 106, we obtain the following upper bounds regarding the problem of deciding the winner in a pushdown game and on constructing a finite state automaton recognising the winning region (in the sense of Remark 15).

Theorem 108. *Let $\mathscr{G}$ be a parity pushdown game using colours $\{0,\dots,d-1\}$ and played on an arena generated by a pushdown system with n control states and with a stack alphabet of size m. Then the following holds*

(i) One can construct in time $\mathscr{O}(m^d 2^{nd^2})$ a deterministic finite state automaton with 2^n states recognising the winning region of Eve in $\mathscr{G}$.

(ii) One can decide in time $\mathscr{O}(m^d 2^{nd^2} + |s|)$, for any configuration $(p, \bot s)$, whether it is winning for Eve in $\mathscr{G}$.

Proof. Consider the parity game $\widetilde{\mathscr{G}}$ from Section 10.3.1. Let $n = |Q|$ and $m = |\Gamma|$. Then $\widetilde{\mathscr{G}}$ is played on an arena with $\mathscr{O}(nm2^{2nd})$ vertices and it uses d colours. Hence, computing the winning region of this later game can be achieved in time $\mathscr{O}(m^d 2^{nd^2})$, see Chapter 2.

Using, Theorem 107, it follows that the set of profiles can be computed in time $\mathscr{O}(m^d 2^{nd^2})$, and by Theorem 106 (and its proof) we know that we can construct (in the same time complexity) a deterministic finite state automaton with 2^n states recognising the winning region W_{Eve}.

The upper bound for the second item simply follows from the fact that running a deterministic automaton on a word is performed in linear time in the length of the word. □

Regarding lower bound, the following result shows that the previous upper bound is optimal. Note that it is enough to consider a reachability objective.

Theorem 109. *Let $\mathscr{G}$ be a reachability pushdown game. Then the following problem is hard for* EXP*: decide whether $(p,\bot)$ is winning for Eve in $\mathscr{G}$.*

Proof. The lower bound is established by reducing the halting problem for alternating linear space bounded Turing machine.

Consider an alternating linear space bounded Turing machine $\mathscr{M}$. We can safely assume that $\mathscr{M}$ has a unique tape and on an input of size n it uses at most n tape squares. Let $Q = Q_\exists \cup Q_\forall$ be the states of the Turing machine where $Q_\exists$ are the existential states and $Q_\forall$ are the universal ones; we let q_a be the (unique) accepting state of the machine. Call A the tape alphabet and let $T \subseteq Q \times A \times Q \times A \times \{\leftarrow, \rightarrow\}$ the transition table of the machine. A configuration of $\mathscr{M}$ is a word C of the form $uqv \in A^*QA^*$ of length $n+1$ (the meaning being that $\mathscr{M}$ is in state q and that the tape contains uv).

We now informally describe a two-player game simulating a computation of $\mathscr{M}$ and argue that it can be encoded as a reachability pushdown game. Think first of $\mathscr{M}$ as being non-deterministic, i.e. $Q_\forall = \emptyset$ and call C_0 the initial configuration. A run of $\mathscr{M}$ can be encoded as a word $r = C_0\sharp t_0\sharp C_1\sharp t_1\sharp C_2\sharp t_2\sharp C_3\sharp\cdots$ where for every $i \geq 0$, $t_i \in T$ is a transition of $\mathscr{M}$ that can be applied in configuration C_i and C_{i+1} is the configuration reached from C_i by applying t_i; it is accepting if some t_i is a transition to the accepting state of $\mathscr{M}$. A way to encode such r with a pushdown game is that Eve pushes symbols in the stack to describe r and to use the control states to impose some structural constraints on the sequence of pushed symbols:

- The first pushed configuration is C_0.
- Every configuration pushed has the right form, i.e. it is a word in A^*QA^* of length $n+1$.
- Every configuration C is followed by some pattern $\sharp t\sharp$ and the transition t in this pattern can be applied from the configuration. This is ensured by storing in the control state of the pushdown process the state of $\mathscr{M}$ and the content of the currently read cell in C.

To ensure these properties a linear number of control states suffices.

Of course, this is not enough because Eve could cheat and push a configuration $C_{i+1} = x_0x_1\cdots x_n$ which is not the successor of $C_i = y_0y_1\cdots y_n$ by t_i. To avoid this, after she described a configuration (say C_{i+1}) and pushed the $\sharp$ symbol, Adam can stop the simulation and claim a mistake by indicating the index k of a wrong update in C_{i+1}. If so, the game goes to a special mode where the following is performed:

- the $\sharp$ symbol is popped as well as the next $n-i$ symbols;
- the current top symbol is x_i and it is stored in the control state of the pushdown process

- the players keep popping until a $\sharp$ symbol is seen and the next symbol t_i is also stored in the control state
- then the players pop $n-i-1$ symbols and then considering the next three symbols they can check whether the update was correct or not (there are several cases depending whether the reading tape was at distance at most 1 of the position of index i).

Again, this can be implemented thanks to a linear number of control states in the pushdown process.

In case Eve cheats the play loops in a non-final sink configuration. Otherwise it loops in a final sink configuration.

Now, if the Turing Machine $\mathscr{M}$ is alternating, the only difference is that the choice of the transition t_i is made by Eve if the control state in C_i is existential and by Adam if it is universal. The rest of the game is unchanged (in particular Eve is still in charge of describing all configurations, regardless of whom picks the transition).

It is then immediate to check that Eve has a winning strategy in this game if and only if the Turing machine accepts from its initial configuration. □

10.3.3 Pushdown and Regular Winning Strategies

In Section 10.3.1, we have seen that the proof of Theorem 107 shows that a winning strategy for Eve (when it exists) can be implemented by pushdown automaton that reads the pushdown system transitions chosen by the players and indicates Eve moves by a function depending only of the current control state and the top-most stack symbol of the strategy automaton.

In this section, we present a different reduction whose aim is to be able to compute a positional winning strategy for Eve which furthermore can be implemented by a finite state automata. As an added benefit, we will see that this strategy is uniform in the sense that it is winning from every vertex of the winning region of Eve.

For the rest of this section, we fix a parity pushdown game $\mathscr{G}$ played on an arena $\mathscr{A} = (G, V_{\text{Eve}}, V_{\text{Adam}})$ generated by a pushdown system $\mathscr{P} = (Q, Q_{\text{Eve}}, Q_{\text{Adam}}, \Gamma, \Delta)$. We also let $V = V_{\text{Eve}} \cup V_{\text{Adam}}$ and we let the colours used in the game be $\{0, \dots, d\}$.

A *summary* is a triple $(p,c,q) \in Q \times \{0,\dots,d\} \times Q$. A set S of summaries is *complete* if $(p_1,c_1,q),(q,c_2,p_2) \in S$ implies that $(p_1, \max(c_1,c_2), p_2) \in S$; it is winning if $(p,c,p) \in S$ implies that c is even. For $R \subseteq Q$, a set of R-summaries is a set of summaries $S \subseteq R \times \{0,\dots,d\} \times R$. Associated with some stack content s, a summary (p,c,q) aims to encode the existence of a sequence of moves from (p,s) to (q,s) where the top symbol of s is never removed and where c is the largest colour visited in the sequence.

Let $P \subseteq Q_{\text{Eve}}$ and $\gamma \in \Gamma_\perp$. A (P,γ)-*local strategy* for Eve is a partial function $\sigma_\gamma : P \to Q \times \{\text{pop}, \text{push}(\gamma) \mid \gamma \in \Gamma\}$ such that $\sigma_\gamma(p) \in \Delta(p,\gamma)$ for all $p \in P$. Equivalently it is a selection for every state in P of a consistent transition of $\mathscr{P}$ when the top symbol is γ. For a subset $R \subseteq Q$, we say that σ_γ *pops in* R if $\sigma_\gamma(q) = (r, pop)$ implies $r \in R$. We say that a (P,γ)-local strategy is *safe* if $\sigma_\gamma(p) = (q, push(\alpha))$ implies that $P \in \mathscr{R}(q,\alpha)$. From now on, we only allowed safe local strategies.

Let $R \subseteq Q$ and $\gamma \in \Gamma$. We associate with (R,γ) the subset

$$W(R,\gamma) = \{q \mid R \in \mathscr{R}(q,\gamma)\}$$

By a small abuse of notation we let $W(\emptyset,\bot) = \{q \mid (q,\bot) \in W_{\text{Eve}}\}$. Remark that for every $q \in W(R,\gamma) \cap Q_{\text{Adam}}$ and $(r,pop) \in \Delta(q,\gamma)$ one has $r \in R$.

We now define a new game $\widehat{\mathscr{G}}$ played on a finite arena and equipped with an ω-regular objective. We start by an informal description of plays in $\widehat{\mathscr{G}}$ and later formally describe the arena and the objective.

A play in $\widehat{\mathscr{G}}$ begins by an initialisation phase:

- The play starts in $(\bot,R)$ where $R = W(\emptyset,\bot) = \{q \mid (q,\bot) \in W_{\text{Eve}}\}$.
- From there, Eve chooses $\sigma_\bot$ an $(R,\bot)$-local strategy and a set of R-summaries that is both complete and winning. Then, the play goes to $(\bot,R,\sigma_\bot,S)$.

Then, the plays goes for rounds of the following form:

- From a vertex $(\gamma,R,\sigma_\gamma,S)$, where $\gamma \in \Gamma$, $R \subseteq Q$, σ_γ is an (R,γ)-local strategy and S is a set of R-summaries, Eve chooses for every $\alpha \in \Gamma$ a $(W(R,\alpha),\alpha))$-local strategy σ_α that pops in R and a set of $W(R,\alpha)$-summaries S_α that is both complete and winning. The play then goes in $(\gamma,R,\sigma_\gamma,S,(\sigma_\alpha,S_\alpha)_{\alpha\in\Gamma})$.
- Then, Adam chooses some α in Γ and the play goes in $(\alpha,W(R,\alpha),\sigma_\alpha,S_\alpha)$.

Consider a tuple $(\gamma,R,\sigma_\gamma,S,(\sigma_\alpha,S_\alpha)_{\alpha\in\Gamma})$ where $\gamma \in \Gamma$, $R \subseteq Q$, σ_γ is an (R,γ)-local strategy, S is a set of R-summaries, and, for every $\alpha \in \Gamma$, σ_α is a $(W(R,\alpha),\alpha))$-local strategy σ_α that pops in R and S_α is a set of $W(R,\alpha)$-summaries that is both complete and winning. The tuple $(\gamma,R,\sigma_\gamma,(\sigma_\alpha,S_\alpha)_{\alpha\in\Gamma})$ is *consistent* if, for every $(p,r) \in R^2$, one has $(p,\max(\mathfrak{c}(p),c,\mathfrak{c}(r)),r) \in S$ as soon as we are in one of the following two situations (the second one being the degenerated version of the first one).

- There exists $\alpha \in \Gamma$, $(q,c,q') \in S_\alpha$ such that
  - (i) either $p \in Q_{\text{Eve}}$ and $(q,\text{push}(\alpha)) = \sigma_\gamma(p)$, or $p \in Q_{\text{Adam}}$ and $(q,\text{push}(\alpha)) \in \Delta(p,\gamma)$, and
  - (ii) either $q' \in Q_{\text{Eve}}$ and $(r,\text{pop}) = \sigma_\alpha(q')$, or $q' \in Q_{\text{Adam}}$ and $(r,\text{pop}) \in \Delta(q',\alpha)$.

  Intuitively, if with state p and top symbol γ one can push α and go to state q from which we know that we can later go back to the same stack content with state q' and maximal colour c, and finally pop γ and end in state r, then we conclude that we can go from p to r while seeing $\max(\mathfrak{c}(p),c,\mathfrak{c}(r))$ as the maximal colour.
- There exists $\alpha \in \Gamma$ and $q \in W(R,\alpha)$ such that
  - (i) either $p \in Q_{\text{Eve}}$ and $(q,\text{push}(\alpha)) = \sigma_\gamma(p)$, or $p \in Q_{\text{Adam}}$ and $(q,\text{push}(\alpha)) \in \Delta(p,\gamma)$, and
  - (ii) either $q \in Q_{\text{Eve}}$ and $(r,\text{pop}) = \sigma_\alpha(q)$, or $q \in Q_{\text{Adam}}$ and $(r,\text{pop}) \in \Delta(q,\alpha)$,
  - (iii) $c = \mathfrak{c}(q)$.

Intuitively, if with state p and top symbol γ one can push α and go to state q and directly pops γ and end in state r, then we conclude that we can go from p to r while seeing $\max(\mathfrak{c}(p),\mathfrak{c}(q),\mathfrak{c}(r))$ as the maximal colour.

In the previous informal description, the only allowed choices for $(\sigma_\alpha, S_\alpha)_{\alpha\in\Gamma}$ are those that leads to consistent tuples. Formally, we define the arena $\widehat{\mathscr{A}}$ as follows:

- There is a special initial vertex $(\bot, W(\emptyset,\bot))$ controlled by Eve.
- For every $\gamma\in\Gamma_\bot$, every $R\subseteq Q$, every (R,γ)-local strategy σ_γ and every set of R-summaries that is both complete and winning there is a vertex $(\gamma,R,\sigma_\gamma,S)$ controlled by Eve.
- There is a vertex $(\gamma,R,\sigma_\gamma,S,(\sigma_\alpha,S_\alpha)_{\alpha\in\Gamma})$ controlled by Adam for every consistent such tuple.
- From every vertex $(\gamma,R,\sigma_\gamma,S)$ there is an edge to every vertex of the form $(\gamma,R,\sigma_\gamma,S,(\sigma_\alpha,S_\alpha)_{\alpha\in\Gamma})$.
- From every vertex $(\gamma,R,\sigma_\gamma,S,(\sigma_\alpha,S_\alpha)_{\alpha\in\Gamma})$ there is an edge to $(\alpha,W(R,\alpha),\sigma_\alpha,S_\alpha)$ for every $\alpha\in\Gamma$.

Hence, a play in $\widehat{\mathscr{G}}$ from the initial vertex $(\gamma_0,R_0)=(\bot,W(\emptyset,\bot))$ is a sequence of vertices

$$\begin{aligned}\widehat{\pi}=&(\bot,R_0)(\gamma_0,R_0,\sigma_0,S_0)(\gamma_0,R_0,\sigma_0,S_0,(\sigma^1_\alpha,S^1_\alpha)_{\alpha\in\Gamma}))(\gamma_1,R_1,\sigma_1,S_1)\\&(\gamma_1,R_1,\sigma_1,S_1,(\sigma^2_\alpha,S^2_\alpha)_{\alpha\in\Gamma}))(\gamma_2,R_2,\sigma_2,S_2)\cdots\end{aligned}$$

with $\sigma_i=\sigma^i_{\gamma_i}$ and $S_i=S^i_{\gamma_i}$ for every $i\geq 1$.

It is losing for Eve if there exists $(q_i)_{i\geq 0},(p_i)_{i\geq 0}\in Q^{\mathbb{N}}$ and $(c_i)_{i\geq 0}\in\{0,\dots,d\}^{\mathbb{N}}$ such that $\limsup(c_i)_{i\geq 0}$ is odd and for every $i\geq 0$ one has

- $(q_i,c_i,p_i)\in S_i$; and
- either $p_i\in Q_{\text{Eve}}$ and $(q_{i+1},\text{push}(\gamma_{i+1}))=\sigma_i(p_i,\gamma_i)$ or $p_i\in Q_{\text{Adam}}$ and $(q_{i+1},\text{push}(\gamma_{i+1}))\in\Delta((p_i,\gamma_i)$.

Note that it is easily seen that the previous objective is an ω-regular one.

We denote by $\widehat{\mathscr{G}}$ the previous game. The following result relies on the connection between $\widehat{\mathscr{G}}$ and the original pushdown game $\mathscr{G}$.

Theorem 110. *Eve has a finite memory winning strategy in $\widehat{\mathscr{G}}$ from $(\bot,W(\emptyset,\bot))$.*

Proof. As the game $\widehat{\mathscr{G}}$ is played on a finite arena and equipped with an ω-regular objective, it suffices to prove that Eve has a winning strategy from $(\bot,W(\emptyset,\bot))$.

Consider a *positional* strategy σ for Eve in $\mathscr{G}$ that is winning on the whole winning region W_{Eve}. Note that existence of positional winning strategies is ensure because $\mathscr{G}$ is a parity game.

Let $s \in \bot\Gamma^*$ be some stack content. We define a set of summaries S_s^σ associated with s (and σ) by letting

$$\begin{aligned} S_\sigma^s = \{(p,c,q) \mid \exists \pi = v_0 \cdots v_k \text{ with } k>0,\ v_0=(p,s),\ v_k=(q,s),\ \mathrm{sh}(v_i) \geq \mathrm{sh}(v_0) \\ \text{for all } 0 \leq i \leq k, \text{ and such that Eve respects } \sigma \text{ in } \pi \text{ and } c \text{ is the largest} \\ \text{colour visited in } \pi\} \end{aligned}$$

and, for every $(p,c,q) \in S_\sigma^s$, we select a play $\pi^s_{(p,c,q)}$ that witnesses $(p,c,q) \in S_\sigma^s$.

Using σ we define a strategy $\widehat{\sigma}$ for Eve in $\widehat{\mathscr{G}}$ and we later argue that it is winning for her from $(\bot, W(\emptyset,\bot))$.

- At the beginning of the play in $(\bot,R)$ with $R = W(\emptyset,\bot)$, Eve moves to $(\bot,R,\sigma_\bot,S)$ where $\sigma_\bot(r) = \sigma((r,\bot))$ for every $r \in R$, and $S = S_\sigma^\bot$
- Assume the current play is

  $$\widehat{\pi} = (\bot,R_0)(\gamma_0,R_0,\sigma_0,S_0)(\gamma_0,R_0,\sigma_0,S_0,(\sigma_\alpha^1,S_\alpha^1)_{\alpha\in\Gamma}))(\gamma_1,R_1,\sigma_1,S_1)\cdots(\gamma_k,R_k,\sigma_k,S_k)$$

  and let $s_{\widehat{\pi}} = \gamma_0\cdots\gamma_k$. Then, Eve goes to $(\gamma_k,R_k,\sigma_k,S_k,(\sigma_\alpha^{k+1},S_\alpha^{k+1})_{\alpha\in\Gamma}))$ with $\sigma_\alpha^{k+1}(r) = \sigma((r,s_{\widehat{\pi}}\alpha))$ for every r such that $(r,s_{\widehat{\pi}}\alpha) \in W_{\mathrm{Eve}}$ and $S_\alpha^{k+1} = S_\sigma^{\widehat{\pi}\alpha}$.

Assume now by contradiction that $\widehat{\sigma}$ is not winning and consider a losing play

$$\begin{aligned} \widehat{\pi} = &(\bot,R_0)(\gamma_0,R_0,\sigma_0,S_0)(\gamma_0,R_0,\sigma_0,S_0,(\sigma_\alpha^1,S_\alpha^1)_{\alpha\in\Gamma}))(\gamma_1,R_1,\sigma_1,S_1) \\ &(\gamma_1,R_1,\sigma_1,S_1,(\sigma_\alpha^2,S_\alpha^2)_{\alpha\in\Gamma}))(\gamma_2,R_2,\sigma_2,S_2)\cdots \end{aligned}$$

Hence, there exists $(q_i)_{i\geq 0},(p_i)_{i\geq 0} \in Q^{\mathbb{N}}$ and $(c_i)_{i\geq 0} \in \{0,\ldots,d\}^{\mathbb{N}}$ such that $\limsup(c_i)_{i\geq 0}$ is odd and for every $i \geq 0$ one has

- $(q_i,c_i,p_i) \in S_i$; and
- either $p_i \in Q_{\mathrm{Eve}}$ and $(q_{i+1},\mathrm{push}(\gamma_{i+1})) = \sigma_i(p_i,\gamma_i)$ or $p_i \in Q_{\mathrm{Adam}}$ and $(q_{i+1},\mathrm{push}(\gamma_{i+1})) \in \Delta((p_i,\gamma_i)$.

Now, consider the play

$$\pi = \pi^{\gamma_0}_{(q_0,c_0,p_0)}\pi^{\gamma_0\gamma_1}_{(q_1,c_1,p_1)}\pi^{\gamma_0\gamma_1\gamma_2}_{(q_2,c_2,p_2)}\pi^{\gamma_0\gamma_1\gamma_2\gamma_3}_{(q_3,c_3,p_3)}\cdots$$

Then it is easily seen by definition that π is losing (because $\widehat{\pi}$ is) while Eve respects her winning strategy σ, which leads a contradiction and concludes the proof. □

Following Theorem 110, fix a finite memory winning strategy $\widehat{\sigma}$ for Eve in $\widehat{\mathscr{G}}$ from $(\bot, W(\emptyset,\bot))$. Using $\widehat{\sigma}$ we define a positional strategy σ for Eve in $\mathscr{G}$.

First, we inductively associate, with any word $s \in \bot\Gamma^*$, a finite play $\widehat{\pi}_s$ in $\widehat{\mathscr{G}}$ where Eve respects her strategy $\widehat{\sigma}$:

- If $s = \bot$, we let $\widehat{\pi}_s = (\bot, W(\emptyset,\bot))(\bot, W(\emptyset,\bot),\sigma_\bot,S)$ where $(\bot, W(\emptyset,\bot),\sigma_\bot,S) = \widehat{\sigma}((\bot, W(\emptyset,\bot)))$.

- If $s = s'\beta$ for some $\beta \in \Gamma$, let $\widehat{\sigma}(\widehat{\pi}_{s'}) = (\gamma, R, \sigma_\gamma, S, (\sigma_\alpha, S_\alpha)_{\alpha\in\Gamma}))$ and define

$$\widehat{\pi}_s = \widehat{\pi}_{s'}(\gamma, R, \sigma_\gamma, S, (\sigma_\alpha, S_\alpha)_{\alpha\in\Gamma}))(\beta, W(R,\beta), \sigma_\beta, S_\beta).$$

Now, for every configuration (q,s) with $q \in Q_{\text{Eve}}$, we let $(\gamma, R, \sigma_\gamma, S_\gamma)$ be the last vertex in $\widehat{\pi}_s$ and if $q \in R$ we let $\sigma((q,s)) = \sigma_\gamma(q)$ and otherwise we pick an arbitrary transition for $\sigma((q,s))$ as (q,s) will be a losing position for Eve (see Proposition 8 below).

The following is a direct rephrasing of the proof of Theorem 106.

Proposition 8. *Let $s \in \Gamma^*\perp$ and let $(\gamma, R, \sigma_\gamma, S)$ be the last vertex in $\widehat{\pi}_s$. Then $R = \{p \mid (p,s) \in W_{\text{Eve}}\}$.*

The following is a consequence of Proposition 8 and of the requirement that in a vertex $(\gamma, R, \sigma_\gamma, S)$ Eve should only propose $(W(R,\alpha),\alpha)$-local strategies that pops in R.

Proposition 9. *Let π be an infinite play in $\mathscr{G}$ starting from some winning position for Eve and where Eve respects strategy σ. Then any vertex visited in π is a winning position for Eve.*

Proof. It suffices to prove that the property is true for the second vertex in π (and then conclude by induction as the strategy π is positional). If the initial vertex belongs to Adam, then by definition all possible successors are winning for Eve (otherwise the initial one would be winning for Adam as well by prefix independence of the parity objective). If the initial vertex is controlled by Eve there are two cases depending whether her move is to push or pop a symbol. If the move is to pop a symbol then, by construction, the state reached belong to R where the last vertex in $\widehat{\pi}_s$ is $(\gamma, R, \sigma_\gamma, S)$, if s denote the stack content after popping: hence, by Proposition 8 we conclude. If the move is to push a symbol then the result follows directly from the fact that we only consider safe local strategies and by Proposition 8. □

The following is an easy consequence of the notion of consistent tuples.

Proposition 10. *Let $(p,s) \in V$ and let $(\gamma, R, \sigma_\gamma, S)$ be the last vertex in $\widehat{\pi}_s$. Assume that $p \in R$. Let $\pi = v_0 \cdots v_k$ be a finite play in $\mathscr{G}$ such that $k > 0$, $v_0 = (p,s)$, $\text{sh}(v_i) \geq \text{sh}(v_0)$ for every $0 < i < k$ and $v_k = (q,s)$ for some $q \in Q$. Then $(p,c,q) \in S$ where c is the largest colour visited in π.*

Proof. We do the proof only when we assume that the inequality $\text{sh}(v_i) \geq \text{sh}(v_0)$ is strict. The case where it is large is then a consequence of the fact that S is complete with successive application of the strict case. The proof is by induction on k. The base case is when $k = 2$, and it corresponds to the degenerated case in the definition of consistent tuple. Now for the general case, when $k > 2$, one simply considers the play $v_1 \cdots v_{k-1}$, applies the induction hypothesis and conclude with the definition of consistent tuple again. □

We are now ready to conclude and prove that σ is a winning strategy for Eve.

Theorem 111. *The positional strategy σ is winning for Eve on the whole winning region in $\mathscr{G}$.*

Proof. Consider an infinite play $\pi = v_0 v_1 \cdots$ starting from some winning position for Eve. Then by Proposition 9 we know that the play stays in the winning region.

By contradiction assume that π is losing. We distinguish between two cases depending whether there is some vertex that is infinitely visited or not in π.

- Assume that there is a vertex $v = (q,s)$ that appears infinitely often in π and choose one of minimal stack height. Let k_0 be such that $v_{k_0} = v$ and such that $\mathrm{sh}(v_j) \geq \mathrm{sh}(v)$ for every $j \geq k_0$. Let $(k_i)_{i\geq 0}$ be the increasing sequence of integers $k_i \geq k_0$ such that $v_{k_i} = v$. We claim that the largest colour visited in the segment $v_{k_i} \cdots v_{k_{i+1}}$ is even: indeed, it is a direct consequence of Proposition 10 and of the fact that the set of summaries we consider are winning. We then conclude that the largest colour infinitely visited in π is even hence, leading a contradiction.

- Assume that no vertex is infinitely often visited in π. As the parity objective is prefix independent we can assume without loss of generality that there is no visited vertex with stack-height strictly smaller than $h = \mathrm{sh}(v_0)$. Factorise π as $v_{i_0} \cdots v_{i_1-1} v_{i_1} \cdots v_{i_2-1} v_{i_2} \cdots v_{i_3-1} \cdots$ where $\mathrm{sh}(v_{i_j}) = \mathrm{sh}(v_{i_{j+1}-1}) = h + j$ and $\mathrm{sh}(v_k) > h + j$ for all $k \geq j+1$ (equivalently stack height $h+j$ is left forever in v_{j+1}). Call s_j the stack content in v_{i_j} and consider the infinite play $\widehat{\pi}$ defined as the limit of the increasing (for prefix ordering) sequence of finite plays $(\widehat{\pi}_{s_j})_{j\geq 0}$: it is a play in $\widehat{\mathscr{G}}$ where Eve respects $\widehat{\sigma}$ hence, it is winning for her. Now let $v_{i_j} = (q_j, s_j)$, $v_{i_{j+1}-1} = (p_j, s_j)$ and let c_j be the largest colour visited in $v_{i_j} \cdots v_{i_{j+1}-1}$. Then, as we assume that π is losing one has $\limsup(c_i)_{i\geq 0}$ is odd. Moreover, if one lets

$$\begin{aligned}\widehat{\pi} =& (\bot, R_0)(\gamma_0, R_0, \sigma_0, S_0)(\gamma_0, R_0, \sigma_0, S_0, (\sigma^1_\alpha, S^1_\alpha)_{\alpha\in\Gamma}))(\gamma_1, R_1, \sigma_1, S_1) \\ & (\gamma_1, R_1, \sigma_1, S_1, (\sigma^2_\alpha, S^2_\alpha)_{\alpha\in\Gamma}))(\gamma_2, R_2, \sigma_2, S_2) \cdots\end{aligned}$$

  one has that

  - $(q_i, c_i, p_i) \in S_i$ (by Proposition 10); and
  - either $p_i \in Q_{\mathrm{Eve}}$ and $(q_{i+1}, \mathrm{push}(\gamma_{i+1})) = \sigma_i(p_i, \gamma_i)$ (by definition of σ) or $p_i \in Q_{\mathrm{Adam}}$ and $(q_{i+1}, \mathrm{push}(\gamma_{i+1})) \in \Delta((p_i, \gamma_i)$ (by definition of σ).

  This means that $\widehat{\pi}$ is losing, leading a contradiction.

Hence, we conclude that π is winning which concludes the proof. □

Remark 18. *Note that the previous strategy σ can be computed by a finite state automaton.*

Bibliographic references

The decidability of pushdown parity games is a consequence of the decidability of monadic second-order logic (MSO) on the infinite complete binary tree [Rab69b]. The

key observation is that any pushdown arena can be defined using MSO in the infinite complete binary tree Δ_2. Indeed every node in Δ_2 can be identified with the binary word encoding the path from the root to this node. Now, if one chooses a fixed-length encoding for the stack alphabet Γ and for the set of states, a configuration (p,s) can be associated to the node of Δ_2 encoded by sp. Using this representation, the effect of a transition of the pushdown system can easily be captured by an MSO-formula. As a consequence, any property expressible in MSO on the arena can be effectively translated into equivalent MSO-property on the full binary tree.

This observation can be used to decide the winner of the game from any given vertex (i.e., Problem 1). It is indeed possible to write an MSO formula $\varphi_{\text{Eve}}(x)$ which expresses that Eve wins the games from x, see [Wal02] for parity games with possibly uncountable arenas. In the simpler case of pushdown arenas in which all vertices have a bounded outdegree writing such a formula is straightforward (see for instance [Cac03, Section 2.3.4]).

Rabin's lemma can be used to show that the winning regions are regular sets of configurations, as defined in Theorem 106. Recall that Rabin's lemma states that if the infinite tree Δ_2 satisfies an existential MSO-formula $\exists X, \varphi(X)$ then there exists a regular set of nodes R such that Δ_2 satisfies $\varphi[R]$. Furthermore, a finite automaton accepting R can be effectively constructed from φ. If we consider the formula $\varphi(X) = \forall x, x \in X \Leftrightarrow \varphi_{\text{Eve}}(x)$, we immediately obtain a finite automaton accepting the binary encodings of the configurations in the winning region for Eve. This automaton can easily be transformed to accept the configurations as in Theorem 106.

Using similar arguments, one can derive the existence of a regular positional winning strategy for both players in the sense of Section 10.3.3.

Although effective, these results do not provide tight upper-bounds as they rely on the translation of MSO properties into equivalent parity tree automata which is non-elementary in the alternation rank of the formula. In the sequel of this bibliographic note, we will focus on the history of decision procedures for pushdown games which provide explicit complexity bounds.

The first result on pushdown games can be traced back to the work of Büchi, which shows that the set of configurations reachable from the initial configuration of a pushdown system forms a regular language and hence can be represented by a finite state automaton [Büc64]. While Büchi's procedure is exponential, Caucal showed that this problem can be solved in polynomial time [Cau88]. The improved algorithm is a saturation process where transitions are incrementally added to a finite automaton. This technique was simplified and adapted to two-player reachability pushdown games where the target set is a regular set of configurations by Bouajjani et al. in [BEM97] and independently in [FWW97]. This saturation approach corresponds to the fixpoint characterization of strategy profiles in Section 10.2.1. This method was latter extended to Büchi pushdown games [Cac02] and finally to parity pushdown games [HO09]. We refer the reader to [CH14] for a survey of this method.

The decidability of parity pushdown games was established in [Wal01] where Walukiewicz gives algorithms to compute winning strategies for both players from a given configuration based on a deterministic pushdown automaton reading the moves of the play (as in Section 10.3.3). In this article, it is also shown that deciding the winner in a two-player pushdown reachability game is hard for EXP. In [Ser03, Cac02], the win-

ning region for both players was shown to be regular based on similar arguments as those presented in Section 10.2. The proof presented in this chapter follows [Ser04], except for the construction of a regular winning strategy given in Section 10.3.3 which is novel.

Kupferman and Vardi in [KV00] reduced deciding the winner in a parity pushdown game to the emptiness problem for two-way alternating parity tree automata. In [Var98], Vardi had previously shown that this emptiness problem for this model of automata is in EXP. In addition, this approach permits to compute a finite automaton accepting the winning region for both players, as well as regular winning strategies for both players. The reduction is very close to the reduction to the decidability of MSO on the full binary tree sketched in the beginning of this section, but by replacing MSO-formulas with two-way alternating parity tree automata, one can obtain optimal complexity (see [Cac02] for a survey of this approach). A similar approach was used by Serre in [Ser06b] to study parity one-counter games (*i.e.* parity pushdown games with a one-letter stack alphabet): by a reduction to the emptiness problem for two-way alternating parity *word* automaton, one obtains a PSPACE upper bound (which is also a matching lower bound).

They were many subsequent works on extensions of parity pushdown games. One line of research considered non-regular winning conditions, based on properties of the stack behaviour along the play [CDT02, BSW03, Gim04, Ser06a]. Another line of research focused on considering games played on infinite arenas generated by extensions of pushdown automata, mainly higher-order pushdown automata and collapsible pushdown automata, mostly motivated by connections with higher-order recursion schemes (for further references on the topic we refer to [BCH$^+$21, CS21]).

GAMES WITH COUNTERS

Chapter 11

Games with Counters

SYLVAIN SCHMITZ

Just like timed games arise from timed systems and pushdown games from pushdown systems, counter games arise from (multi-)counter systems. Those are finite-state systems further endowed with a finite number of counters whose values range over the natural numbers, and are widely used to model and reason about systems handling discrete resources. Such resources include for instance money on a bank account, items on a factory line, molecules in chemical reactions, organisms in biological ones, replicated processes in distributed computing, etc. As with timed or pushdown systems, counter systems give rise to infinite graphs that can be turned into infinite game arenas.

One could populate a zoo with the many variants of counter systems, depending on the available counter operations. One of the best known specimens in this zoo are *Minsky machines* [Min67], where the operations are incrementing a counter, decrementing it, or testing whether its value is zero. Minsky machines are a universal model of computation: their reachability problem is undecidable, already with only two counters. From the algorithmic perspective we promote in this book, this means that the counter games arising from Minsky machines are not going to be very interesting, unless perhaps if we restrict ourselves to a single counter. A more promising species in our zoo are *vector addition systems with states* [Gre78, HP79]—or, equivalently, *Petri nets* [Pet62]—, where the only available operations are increments and decrements. Vector addition systems with states enjoy a decidable reachability problem [May81, Kos82, Lam92, Ler11], which makes them a much better candidate for studying the associated games.

In this chapter, we focus on vector games, that is, on games defined on arenas defined by vector addition systems with states with a partition of states controlled by Eve and Adam. As we are going to see in Section 11.1, those games turn out to be undecidable already for quite restricted objectives and just two counters. We then investigate two restricted classes of vector games.

1. In Section 11.2, we consider *one-counter games*. These can be reduced to the

pushdown games of Chapter 10 and are therefore decidable. Most of the section is thus devoted to proving sharp complexity lower bounds, already in the case of so-called *countdown games*.

2. In Section 11.3, we turn our attention to the main results of this chapter. By suitably restricting both the systems, with an *asymmetry* condition that forbids Adam to manipulate the counters, and the objective, with a *monotonicity* condition that ensures that Eve's winning region is upwards closed—meaning that larger counter values make it easier for her to win—, one obtains a class of decidable vector games where finite memory strategies are sufficient.

   - This class is still rich enough to find many applications, and we zoom in on the connections with resource-conscious games like *energy* games and *bounding* games in Section 11.4—a subject that will be taken further in Chapter 12.
   - The computational complexity of asymmetric monotonic vector games is now well-understood, and we devote Section 11.5 to the topic; Table 11.1 at the end of the chapter summarises these results.

11.1 Vector games

A *vector system* is a finite directed graph with a partition of the vertices and weighted edges. Formally, it is a tuple $\mathscr{V} = (\mathscr{L}, A, \mathscr{L}_{\text{Eve}}, \mathscr{L}_{\text{Adam}}, k)$ where $k \in \mathbb{N}$ is a dimension, $\mathscr{L}$ is a finite set of locations partitioned into the locations controlled by Eve and Adam, i.e., $\mathscr{L} = \mathscr{L}_{\text{Eve}} \uplus \mathscr{L}_{\text{Adam}}$, and $A \subseteq \mathscr{L} \times \mathbb{Z}^k \times \mathscr{L}$ is a finite set of weighted actions. We write $\ell \xrightarrow{\vec{u}} \ell'$ rather than $(\ell, \vec{u}, \ell')$ for actions in A. A *vector addition system with states* is a vector system where $\mathscr{L}_{\text{Adam}} = \emptyset$, i.e., it corresponds to the one-player case.

Example 7 (vector system)**.** *Figure 11.1 presents a vector system of dimension two with locations $\{\ell, \ell'\}$ where ℓ is controlled by Eve and ℓ' by Adam.*

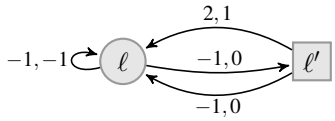


Figure 11.1: A vector system.

The intuition behind a vector system is that it maintains k counters $\mathsf{c}_1, \ldots, \mathsf{c}_k$ assigned to integer values. An action $\ell \xrightarrow{\vec{u}} \ell' \in A$ then updates each counter by adding the corresponding entry of $\vec{u}$, that is for all $1 \leq j \leq k$, the action performs the update $\mathsf{c}_j := \mathsf{c}_j + \vec{u}(j)$.

Before we proceed any further, let us fix some notations for vectors in $\mathbb{Z}^k$. We write '$\vec{0}$' for the *zero vector* with $\vec{0}(j) \stackrel{\text{def}}{=} 0$ for all $1 \leq j \leq k$. For all $1 \leq j \leq k$, we write '$\vec{e}_j$' for the *unit vector* with $\vec{e}_j(j) \stackrel{\text{def}}{=} 1$ and $\vec{e}_j(j') \stackrel{\text{def}}{=} 0$ for all $j' \neq j$. Addition

and comparison are defined componentwise, so that for instance $\vec{u} \leq \vec{u}'$ if and only if for all $1 \leq j \leq k$, $\vec{u}(j) \leq \vec{u}'(j)$. We write $w(\ell \xrightarrow{\vec{u}} \ell') \stackrel{\text{def}}{=} \vec{u}$ for the weight of an action and $w(\pi) \stackrel{\text{def}}{=} \sum_{1 \leq j \leq |\pi|} w(\pi_j)$ for the cumulative weight of a finite sequence of actions $\pi \in A^*$. For a vector $\vec{u} \in \mathbb{Z}^k$, we use its *infinity norm* $\|\vec{u}\| \stackrel{\text{def}}{=} \max_{1 \leq j \leq k} |\vec{u}(j)|$, hence $\|\vec{0}\| = 0$ and $\|\vec{e}_j\| = \|-\vec{e}_j\| = 1$, and we let $\|\ell \xrightarrow{\vec{u}} \ell'\| \stackrel{\text{def}}{=} \|w(\ell \xrightarrow{\vec{u}} \ell')\| = \|\vec{u}\|$ and $\|A\| \stackrel{\text{def}}{=} \max_{a \in A} \|w(a)\|$. Unless stated otherwise, we assume that all our vectors are represented in binary, hence $\|A\|$ may be exponential in the size of $\mathscr{V}$.

11.1.1 Arenas and Games

A vector system gives rise to an infinite graph $G_{\mathbb{N}} \stackrel{\text{def}}{=} (V, E)$ over the set of vertices $V \stackrel{\text{def}}{=} (\mathscr{L} \times \mathbb{N}^k) \uplus \{\bot\}$. The vertices of the graph are either *configurations* $\ell(\vec{v})$ consisting of a location $\ell \in \mathscr{L}$ and a vector of non-negative integers $\vec{v} \in \mathbb{N}^k$—such a vector represents a valuation of the counters $\mathsf{c}_1, \ldots, \mathsf{c}_k$—, or the *sink* $\bot$.

Consider an action in $a = (\ell \xrightarrow{\vec{u}} \ell')$ in A: we see it as a partial function $a : \mathscr{L} \times \mathbb{N}^k \nrightarrow \mathscr{L} \times \mathbb{N}^k$ with domain $\mathrm{dom}\, a \stackrel{\text{def}}{=} \{\ell(\vec{v}) \mid \vec{v} + \vec{u} \geq \vec{0}\}$ and image $a(\ell(\vec{v})) \stackrel{\text{def}}{=} \ell'(\vec{v} + \vec{u})$; let also $\mathrm{dom} A \stackrel{\text{def}}{=} \bigcup_{a \in A} \mathrm{dom}\, a$. This allows us to define the set E of edges as a set of pairs

$$E \stackrel{\text{def}}{=} \{(\ell(\vec{v}), a(\ell(\vec{v}))) \mid a \in A \text{ and } \ell(\vec{v}) \in \mathrm{dom}\, a\} \\ \cup \{(\ell(\vec{v}), \bot) \mid \ell(\vec{v}) \notin \mathrm{dom} A\} \cup \{(\bot, \bot)\}\,,$$

where $\mathrm{In}((v, v')) \stackrel{\text{def}}{=} v$ and $\mathrm{Out}((v, v')) \stackrel{\text{def}}{=} v'$ for all edges $(v, v') \in E$. There is therefore an edge (v, v') between two configurations $v = \ell(\vec{v})$ and $v' = \ell'(\vec{v}')$ if there exists an action $\ell \xrightarrow{\vec{u}} \ell' \in A$ such that $\vec{v}' = \vec{v} + \vec{u}$. Note that, quite importantly, $\vec{v} + \vec{u}$ must be non-negative on every coordinate for this edge to exist. If no action can be applied, there is an edge to the sink $\bot$, which ensures that E is *total*: for all $v \in V$, there exists an edge $(v, v') \in E$ for some v', and thus there are no 'deadlocks' in the graph.

The configurations are naturally partitioned into those in $V_{\mathrm{Eve}} \stackrel{\text{def}}{=} \mathscr{L}_{\mathrm{Eve}} \times \mathbb{N}^k$ controlled by Eve and those in $V_{\mathrm{Adam}} \stackrel{\text{def}}{=} \mathscr{L}_{\mathrm{Adam}} \times \mathbb{N}^k$ controlled by Adam. Regarding the sink, the only edge starting from $\bot$ loops back to it, and it does not matter who of Eve or Adam controls it. This gives rise to an infinite arena $\mathscr{A}_{\mathbb{N}} \stackrel{\text{def}}{=} (G_{\mathbb{N}}, V_{\mathrm{Eve}}, V_{\mathrm{Adam}})$ called the *natural semantics* of $\mathscr{V}$.

Although we work in a turn-based setting with perfect information, it is sometimes enlightening to consider the partial map $\Delta : V \times A \nrightarrow E$ defined by $\Delta(\ell(\vec{v}), a) \stackrel{\text{def}}{=} (\ell(\vec{v}), a(\ell(\vec{v})))$ if $\ell(\vec{v}) \in \mathrm{dom}\, a$ and $\Delta(\ell(\vec{v}), a) \stackrel{\text{def}}{=} (\ell(\vec{v}), \bot)$ if $\ell(\vec{v}) \notin \mathrm{dom} A$. Note that a sequence π over E that avoids the sink can also be described by an initial configuration $\ell_0(\vec{v}_0)$ paired with a sequence over A.

Example 8 (natural semantics)**.** *Figure 11.2 illustrates the natural semantics of the system of Figure 11.1; observe that all the configurations $\ell(0, n)$ for $n \in \mathbb{N}$ lead to the sink.*

A vector system $\mathscr{V} = (\mathscr{L}, A, \mathscr{L}_{\mathrm{Eve}}, \mathscr{L}_{\mathrm{Adam}}, k)$, a colouring $\mathfrak{c} : E \to C$, and an objective $\Omega \subseteq C^\omega$ together define a *vector game* $\mathscr{G} = (\mathscr{A}_{\mathbb{N}}(\mathscr{V}), \mathfrak{c}, \Omega)$. Because $\mathscr{A}_{\mathbb{N}}(\mathscr{V})$ is an infinite arena, we need to impose restrictions on our colourings $\mathfrak{c} : E \to C$ and the qualitative objectives $\Omega \subseteq C^\omega$; at the very least, they should be recursive.

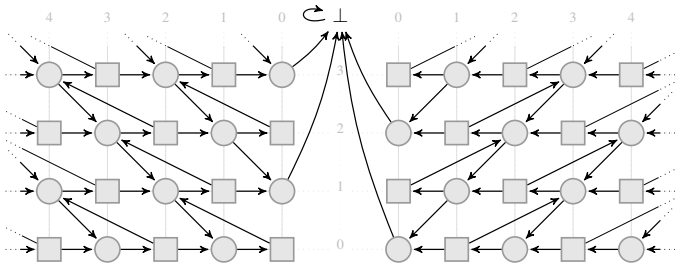


Figure 11.2: The natural semantics of the vector system of Figure 11.1: a circle (resp. a square) at position (i, j) of the grid denotes a configuration $\ell(i, j)$ (resp. $\ell'(i, j)$) controlled by Eve (resp. Adam).

There are then two variants of the associated decision problem:

- the *given initial credit* variant, where we are given $\mathscr{V}$, $\mathfrak{c}$, Ω, a location $\ell_0 \in \mathscr{L}$ and an initial credit $\vec{v}_0 \in \mathbb{N}^k$, and ask whether Eve wins $\mathscr{G}$ from the initial configuration $\ell_0(\vec{v}_0)$;
- the *existential initial credit* variant, where we are given $\mathscr{V}$, $\mathfrak{c}$, Ω, and a location $\ell_0 \in \mathscr{L}$, and ask whether there exists an initial credit $\vec{v}_0 \in \mathbb{N}^k$ such that Eve wins $\mathscr{G}$ from the initial configuration $\ell_0(\vec{v}_0)$.

Let us instantiate the previous abstract definition of vector games. We first consider two 'reachability-like' objectives, where $C \stackrel{\text{def}}{=} \{\varepsilon, \text{Win}\}$ and $\Omega \stackrel{\text{def}}{=} \texttt{Reach}$, namely configuration reachability and coverability. The difference between the two is that, in the configuration reachability problem, a specific configuration $\ell_f(\vec{v}_f)$ should be visited, whereas in the coverability problem, Eve attempts to visit $\ell_f(\vec{v}')$ for some vector $\vec{v}'$ componentwise larger or equal to $\vec{v}_f$.[1]

Problem 13 (configuration reachability vector game with given initial credit)**.**

INPUT: *A vector system $\mathscr{V} = (\mathscr{L}, A, \mathscr{L}_{\text{Eve}}, \mathscr{L}_{\text{Adam}}, k)$, an initial location $\ell_0 \in \mathscr{L}$, an initial credit $\vec{v}_0 \in \mathbb{N}^k$, and a target configuration $\ell_f(\vec{v}_f) \in \mathscr{L} \times \mathbb{N}^k$.*

OUTPUT: *Does Eve have a strategy to reach $\ell(\vec{v})$ from $\ell_0(\vec{v}_0)$? That is, does she win the* configuration reachability *game $(\mathscr{A}_{\mathbb{N}}(\mathscr{V}), \mathfrak{c}, \texttt{Reach})$ from $\ell_0(\vec{v}_0)$, where $\mathfrak{c}(e) =$ Win if and only if $In(e) = \ell_f(\vec{v}_f)$?*

Problem 14 (coverability vector game with given initial credit)**.**

INPUT: *A vector system $\mathscr{V} = (\mathscr{L}, A, \mathscr{L}_{\text{Eve}}, \mathscr{L}_{\text{Adam}}, k)$, an initial location $\ell_0 \in \mathscr{L}$, an initial credit $\vec{v}_0 \in \mathbb{N}^k$, and a target configuration $\ell_f(\vec{v}_f) \in \mathscr{L} \times \mathbb{N}^k$.*

OUTPUT: *Does Eve have a strategy to reach $\ell(\vec{v}')$ for some $\vec{v}' \geq \vec{v}_f$ from $\ell_0(\vec{v}_0)$? That is, does she win the* coverability *game $(\mathscr{A}_{\mathbb{N}}(\mathscr{V}), \mathfrak{c}, \texttt{Reach})$ from $\ell_0(\vec{v}_0)$, where $\mathfrak{c}(e) =$ Win if and only if $In(e) = \ell_f(\vec{v}')$ for some $\vec{v}' \geq \vec{v}_f$?*

[1] The name 'coverability' comes from the the literature on Petri nets and vector addition systems with states, because Eve is attempting to *cover* $\ell_f(\vec{v}_f)$, i.e., to reach a configuration $\ell_f(\vec{v}')$ with $\vec{v}' \geq \vec{v}_f$.

Example 9 (Objectives)**.** *Consider the target configuration $\ell(2,2)$ in Figures 11.1 and 11.2. Eve's winning region in the configuration reachability vector game is $W_{\text{Eve}} = \{\ell(n+1,n+1) \mid n \in \mathbb{N}\} \cup \{\ell'(0,1)\}$, displayed on the left in Figure 11.3. Eve has indeed an obvious winning strategy from any configuration $\ell(n,n)$ with $n \geq 2$, which is to use the action $\ell \xrightarrow{-1,-1} \ell$ until she reaches $\ell(2,2)$. Furthermore, in $\ell'(0,1)$—due to the natural semantics—, Adam has no choice but to use the action $\ell' \xrightarrow{2,1} \ell$: therefore $\ell'(0,1)$ and $\ell(1,1)$ are also winning for Eve.*

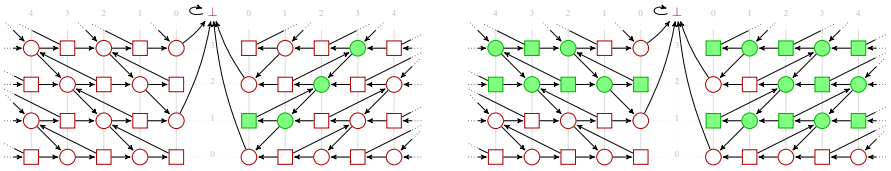


Figure 11.3: The winning regions of Eve in the configuration reachability game (left) and the coverability game (right) on the graphs of Figures 11.1 and 11.2 with target configuration $\ell(2,2)$. The winning vertices are in filled in green, while the losing ones are filled with white with a red border; the sink is always losing.

In the coverability vector game, Eve's winning region is $W_{\text{Eve}} = \{\ell(m+2,n+2), \ell'(m+2,n+2), \ell'(0,n+1), \ell(1,n+2), \ell'(2m+2,1), \ell(2m+3,1) \mid m,n \in \mathbb{N}\}$ displayed on the right in Figure 11.3. Observe in particular that Adam is forced to use the action $\ell' \xrightarrow{2,1} \ell$ from the configurations of the form $\ell'(0,n+1)$, which brings him to a configuration $\ell(2,n+2)$ coloured Win in the game, and thus the configurations of the form $\ell(1,n+1)$ are also winning for Eve since she can play $\ell \xrightarrow{-1,0} \ell'$. Thus the configurations of the form $\ell(2m+3,n+1)$ are also winning for Eve, as she can play the action $\ell \xrightarrow{-1,0} \ell'$ to a winning configuration $\ell'(2m+2,n+1)$ where all the actions available to Adam go into her winning region.

Remark 19 (Location reachability)**.** *One can notice that coverability is equivalent to* location reachability*, where we are given a target location ℓ_f but no target vector, and want to know whether Eve have a strategy to reach $\ell_f(\vec{v})$ for some $\vec{v}$.*

Indeed, in both configuration reachability and coverability, we can assume without loss of generality that $\ell_f \in \mathscr{L}_{\text{Eve}}$ is controlled by Eve and that $\vec{v}_f = \vec{0}$ is the zero vector. Here is a LOGSPACE *reduction to that case. If $\ell_0(\vec{v}_0) = \ell_f(\vec{v}_f)$ in the case of configuration reachability, or $\ell_0 = \ell_f$ and $\vec{v}_0 \geq \vec{v}_f$ in the case of coverability, the problem is trivial. Otherwise, any winning play must use at least one action. For each incoming action $a = (\ell \xrightarrow{\vec{u}} \ell_f)$ of ℓ_f, create a new location ℓ_a controlled by Eve and replace a by $\ell \xrightarrow{\vec{u}} \ell_a \xrightarrow{\vec{0}} \ell_f$, so that Eve gains the control right before any play reaches ℓ_f. Also add a new location $\Smiley$ controlled by Eve with actions $\ell_a \xrightarrow{-\vec{v}_f} \Smiley$, and use $\Smiley(\vec{0})$ as target configuration.*

Remark 20 (Coverability to reachability)**.** *There is a* LOGSPACE *reduction from coverability to configuration reachability. By Remark 19, we can assume without loss of generality that $\ell_f \in \mathscr{L}_{\text{Eve}}$ is controlled by Eve and that $\vec{v}_f = \vec{0}$ is the zero vector. It suffices therefore to add an action $\ell_f \xrightarrow{-\vec{e}_j} \ell_f$ for all $1 \leq j \leq k$.*

Departing from reachability games, the following is a very simple kind of safety games where $C \stackrel{\text{def}}{=} \{\varepsilon, \text{Lose}\}$ and $\Omega \stackrel{\text{def}}{=} \texttt{Safe}$; Figure 11.4 shows Eve's winning region in the case of the graphs of Figures 11.1 and 11.2.

Problem 15 (non-termination vector game with given initial credit)**.**

INPUT: *A vector system $\mathscr{V} = (\mathscr{L}, A, \mathscr{L}_{\text{Eve}}, \mathscr{L}_{\text{Adam}}, k)$, an initial location $\ell_0 \in \mathscr{L}$, and an initial credit $\vec{v}_0 \in \mathbb{N}^k$.*

OUTPUT: *Does Eve have a strategy to avoid the sink $\perp$ from $\ell_0(\vec{v}_0)$? That is, does she win the* non-termination *game $(\mathscr{A}_{\mathbb{N}}(\mathscr{V}), \mathfrak{c}, \texttt{Safe})$ from $\ell_0(\vec{v}_0)$, where $\mathfrak{c}(e) = \text{Lose}$ if and only if $In(e) = \perp$?*

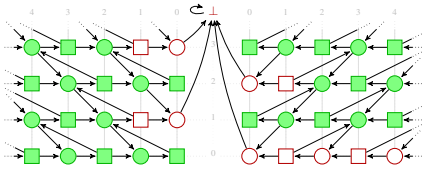


Figure 11.4: The winning region of Eve in the non-termination game on the graphs of Figures 11.1 and 11.2.

Finally, one of the most general vector games are parity games, where $C \stackrel{\text{def}}{=} \{1, \ldots, d\}$ and $\Omega \stackrel{\text{def}}{=} \texttt{Parity}$. In order to define a colouring of the natural semantics, we assume that we are provided with a *location colouring* $\text{lcol}: \mathscr{L} \to \{1, \ldots, d\}$.

Problem 16 (parity vector game with given initial credit)**.**

INPUT: *A vector system $\mathscr{V} = (\mathscr{L}, A, \mathscr{L}_{\text{Eve}}, \mathscr{L}_{\text{Adam}}, k)$, an initial location $\ell_0 \in \mathscr{L}$, an initial credit $\vec{v}_0 \in \mathbb{N}^k$, and a location colouring $\text{lcol}: \mathscr{L} \to \{1, \ldots, d\}$ for some $d > 0$.*

OUTPUT: *Does Eve have a strategy to simultaneously avoid the sink $\perp$ and fulfil the parity objective from $\ell_0(\vec{v}_0)$? That is, does she win the* parity *game $(\mathscr{A}_{\mathbb{N}}(\mathscr{V}), \mathfrak{c}, \texttt{Parity})$ from $\ell_0(\vec{v}_0)$, where $\mathfrak{c}(e) \stackrel{def}{=} \text{lcol}(\ell)$ if $In(e) = \ell(\vec{v})$ for some $\vec{v} \in \mathbb{N}^k$, and $\mathfrak{c}(e) \stackrel{def}{=} 1$ if $In(e) = \perp$?*

Remark 21 (Non termination to parity)**.** *There is a* LOGSPACE *reduction from non-termination to parity. Indeed, the two games coincide if we pick the constant location colouring defined by $\text{lcol}(\ell) \stackrel{def}{=} 2$ for all $\ell \in \mathscr{L}$ in the parity game.*

Remark 22 (Coverability to parity)**.** *There is a* LOGSPACE *reduction from coverability to parity. Indeed, by Remark 19, we can assume that $\ell_f \in \mathscr{L}_{\text{Eve}}$ is controlled by Eve and that the target credit is $\vec{v}_f = \vec{0}$ the zero vector. It suffices therefore to add an action $\ell_f \xrightarrow{\vec{0}} \ell_f$ and to colour every location $\ell \neq \ell_f$ with $\text{lcol}(\ell) \stackrel{def}{=} 1$ and to set $\text{lcol}(\ell_f) \stackrel{def}{=} 2$.*

The existential initial credit variants of Problems 13 to 16 are defined similarly, where $\vec{v}_0$ is not given as part of the input, but existentially quantified in the question.

11.1.2 Undecidability

The bad news is that, although Problems 13 to 16 are all decidable in the one-player case—see the bibliographic notes Section 11.5.3 at the end of the chapter—, they become undecidable in the two-player setting.

Theorem 112 (Undecidability of vector games)**.** *Configuration reachability, coverability, non-termination, and parity vector games, both with given and with existential initial credit, are undecidable in any dimension $k \geq 2$.*

Proof. By Remarks 20 and 21, it suffices to prove the undecidability of coverability and non-termination. For this, we exhibit reductions from the halting problem of deterministic Minsky machines with at least two counters.

Formally, a *deterministic Minsky machine* with k counters $\mathscr{M} = (\mathscr{L}, A, k)$ is defined similarly to a vector addition system with states with additional *zero test* actions $a = (\ell \xrightarrow{i \overset{?}{=} 0} \ell')$. The set of locations contains a distinguished 'halt' location $\ell_{\texttt{halt}}$, and for every $\ell \in \mathscr{L}$, exactly one of the following holds: either (i) $(\ell \xrightarrow{\vec{e}_i} \ell') \in A$ for some $0 < i \leq k$ and $\ell' \in \mathscr{L}$, or (ii) $(\ell \xrightarrow{i \overset{?}{=} 0} \ell') \in A$ and $(\ell \xrightarrow{-\vec{e}_i} \ell'') \in A$ for some $0 < i \leq k$ and $\ell', \ell'' \in \mathscr{L}$, or (iii) $\ell = \ell_{\texttt{halt}}$. The semantics of $\mathscr{M}$ extends the natural semantics by handling zero tests actions $a = (\ell \xrightarrow{i \overset{?}{=} 0} \ell')$: we define the domain as $\operatorname{dom} a \overset{\text{def}}{=} \{\ell(\vec{v}) \mid \vec{v}(i) = 0\}$ and the image by $a(\ell(\vec{v})) \overset{\text{def}}{=} \ell(\vec{v})$. This semantics is deterministic, and from any starting vertex of $\mathscr{A}_{\mathbb{N}}(\mathscr{M})$, there is a unique play, which either eventually visits $\ell_{\texttt{halt}}$ and then the sink in the next step, or keeps avoiding both $\ell_{\texttt{halt}}$ and the sink indefinitely.

The *halting problem* asks, given a deterministic Minsky machine and an initial location ℓ_0, whether it halts, that is, whether $\ell_{\texttt{halt}}(\vec{v})$ is reachable for some $\vec{v} \in \mathbb{N}^k$ starting from $\ell_0(\vec{0})$. The halting problem is undecidable in any dimension $k \geq 2$ [Min67]. Thus the halting problem is akin to the coverability of $\ell_{\texttt{halt}}(\vec{0})$ with given initial credit $\vec{0}$, but on the one hand there is only one player and on the other hand the machine can perform zero tests.

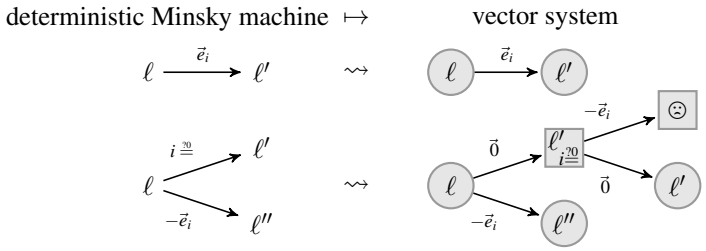


Figure 11.5: Schema of the reduction in the proof of Theorem 112.

Here is now a reduction to Problem 14. Given an instance of the halting problem, i.e., given a deterministic Minsky machine $\mathscr{M} = (\mathscr{L}, A, k)$ and an initial location ℓ_0, we construct a vector system $\mathscr{V} \overset{\text{def}}{=} (\mathscr{L} \uplus \mathscr{L}_{\overset{?}{=}0} \uplus \{☹\}, A', \mathscr{L}, \mathscr{L}_{\overset{?}{=}0} \uplus \{☹\}, k)$ where all the original locations are controlled by Eve and $\mathscr{L}_{\overset{?}{=}0} \uplus \{☹\}$ is a set of new locations

controlled by Adam. We use $\mathscr{L}_{\overset{?0}{=}}$ as a set of locations defined by

$$\mathscr{L}_{\overset{?0}{=}} \overset{\text{def}}{=} \{\ell'_{i\overset{?0}{=}} \mid \exists \ell \in \mathscr{L} \,.\, (\ell \xrightarrow{i\overset{?0}{=}} \ell') \in A\}$$

and define the set of actions by (see Figure 11.5)

$$\begin{aligned} A' \overset{\text{def}}{=} &\{\ell \xrightarrow{\vec{e}_i} \ell' \mid (\ell \xrightarrow{\vec{e}_i} \ell') \in A\} \cup \{\ell \xrightarrow{-\vec{e}_i} \ell'' \mid (\ell \xrightarrow{-\vec{e}_i} \ell'') \in A\} \\ &\cup \{\ell \xrightarrow{\vec{0}} \ell'_{i\overset{?0}{=}},\ \ell'_{i\overset{?0}{=}} \xrightarrow{\vec{0}} \ell',\ \ell'_{i\overset{?0}{=}} \xrightarrow{-\vec{e}_i} ☹ \mid (\ell \xrightarrow{i\overset{?0}{=}} \ell') \in A\}\,. \end{aligned}$$

We use $\ell_0(\vec{0})$ as initial configuration and $\ell_{\texttt{halt}}(\vec{0})$ as target configuration for the constructed coverability instance. Here is the crux of the argument why Eve has a winning strategy to cover $\ell_{\texttt{halt}}(\vec{0})$ from $\ell_0(\vec{0})$ if and only if the Minsky machine halts. Consider any configuration $\ell(\vec{v})$. If $(\ell \xrightarrow{\vec{e}_i} \ell') \in A$, Eve has no choice but to apply $\ell \xrightarrow{\vec{e}_i}_{?0} \ell'$ and go to the configuration $\ell'(\vec{v}+\vec{e}_i)$ also reached in one step in $\mathscr{M}$. If $\{\ell \xrightarrow{i\overset{?0}{=}} \ell', \ell \xrightarrow{-\vec{e}_i} \ell''\} \in A$ and $\vec{v}(i) = 0$, due to the natural semantics, Eve cannot use the action $\ell \xrightarrow{-\vec{e}_i} \ell''$, thus she must use $\ell \xrightarrow{\vec{0}} \ell'_{i\overset{?0}{=}}$. Still due to the natural semantics, Adam cannot use $\ell'_{i\overset{?0}{=}} \xrightarrow{-\vec{e}_i} ☹$, thus he must use $\ell'_{i\overset{?0}{=}} \xrightarrow{\vec{0}} \ell'$. Hence Eve regains the control in $\ell'(\vec{v})$, which was also the configuration reached in one step in $\mathscr{M}$. Finally, if $\{\ell \xrightarrow{i\overset{?0}{=}} \ell', \ell \xrightarrow{-\vec{e}_i} \ell''\} \in A$ and $\vec{v}(i) > 0$, Eve can choose: if she uses $\ell \xrightarrow{-\vec{e}_i} \ell''$, she ends in the configuration $\ell''(\vec{v}-\vec{e}_i)$ also reached in one step in $\mathscr{M}$. In fact, she should not use $\ell \xrightarrow{\vec{0}} \ell'_{i\overset{?0}{=}}$, because Adam would then have the opportunity to apply $\ell'_{i\overset{?0}{=}} \xrightarrow{-\vec{e}_i} ☹$ and to win, as ☹ is a deadlock location and all the subsequent moves end in the sink. Thus, if $\mathscr{M}$ halts, then Eve has a winning strategy that simply follows the unique play of $\mathscr{M}$, and conversely, if Eve wins, then necessarily she had to follow the play of $\mathscr{M}$ and thus the machine halts.

Further note that, in a deterministic Minsky machine the halting problem is similarly akin to the *complement* of non-termination with given initial credit $\vec{0}$. This means that, in the vector system $\mathscr{V} = (\mathscr{L} \uplus \mathscr{L}_{\overset{?0}{=}} \uplus \{☹\}, A', \mathscr{L}, \mathscr{L}_{\overset{?0}{=}} \uplus \{☹\}, k)$ defined earlier, Eve has a winning strategy to avoid the sink from $\ell_0(\vec{0})$ if and only if the given Minsky machine does not halt from $\ell_0(\vec{0})$, which shows the undecidability of Problem 15.

Finally, let us observe that both the existential and the universal initial credit variants of the halting problem are also undecidable. Indeed, given an instance of the halting problem, i.e., given a deterministic Minsky machine $\mathscr{M} = (\mathscr{L}, A, k)$ and an initial location ℓ_0, we add k new locations $\ell_k, \ell_{k-1}, \ldots, \ell_1$ with respective actions $\ell_j \xrightarrow{-\vec{e}_j} \ell_j$ and $\ell_j \xrightarrow{j\overset{?0}{=}} \ell_{j-1}$ for all $k \geq j > 0$. This modified machine first resets all its counters to zero before reaching $\ell_0(\vec{0})$ and then performs the same execution as the original machine. Thus there exists an initial credit $\vec{v}$ such that the modified machine reaches $\ell_{\texttt{halt}}$ from $\ell_k(\vec{v})$ if and only if for all initial credits $\vec{v}$ the modified machine reaches $\ell_{\texttt{halt}}$ from $\ell_k(\vec{v})$, if and only if $\ell_{\texttt{halt}}$ was reachable from $\ell_0(\vec{0})$ in the original machine. The previous construction of a vector system applied to the modified machine then shows the undecidability of the existential initial credit variants of Problems 14 and 15. □

11.2 Games in dimension one

Theorem 112 leaves open whether vector games might be decidable in dimension one. They are indeed decidable, and more generally we learned in Chapter 10 that one-counter games—with the additional ability to test the counter for zero—were decidable and in fact PSPACE-complete. This might seem to settle the case of vector games in dimension one, except that the one-counter games of Chapter 10 only allow integer weights in $\{-1,1\}$, whereas we allow arbitrary updates in $\mathbb{Z}$ with a binary encoding. Hence the PSPACE upper bound of Chapter 10 becomes an EXPSPACE one for *succinct one-counter games*.

Corollary 20 (One-dimensional vector games are in EXPSPACE)**.** *Configuration reachability, coverability, non-termination, and parity vector games, both with given and with existential initial credit, are in* EXPSPACE *in dimension one.*

The goal of this section is therefore to establish that this EXPSPACE upper bound is tight (in most cases), by proving a matching lower bound in Section 11.2.2. But first, we will study a class of one-dimensional vector games of independent interest in Section 11.2.1: countdown games.

11.2.1 Countdown Games

A one-dimensional vector system $\mathscr{V} = (\mathscr{L}, A, \mathscr{L}_{\text{Eve}}, \mathscr{L}_{\text{Adam}}, 1)$ is called a *countdown system* if $A \subseteq \mathscr{L} \times \mathbb{Z}_{<0} \times \mathscr{L}$, that is, if for all $(\ell \xrightarrow{z} \ell') \in A$, $z < 0$. We consider the games defined by countdown systems, both with given and with existential initial credit, and call the resulting games *countdown games*.

Theorem 113 (Countdown games are EXP-complete)**.** *Configuration reachability and coverability countdown games with given initial credit are* EXP*-complete.*

Proof. For the upper bound, consider an instance, i.e., a countdown system $\mathscr{V} = (\mathscr{L}, A, \mathscr{L}_{\text{Eve}}, \mathscr{L}_{\text{Adam}}, 1)$, an initial location $\ell_0 \in \mathscr{L}$, an initial credit $n_0 \in \mathbb{N}$, and a target configuration $\ell_f(n_f) \in \mathscr{L} \times \mathbb{N}$. Because every action decreases strictly the counter value, the reachable part of the natural semantics of $\mathscr{V}$ starting from $\ell_0(n_0)$ is finite and of size at most $1 + |\mathscr{L}| \cdot (n_0 + 1)$, and because n_0 is encoded in binary, this is at most exponential in the size of the instance. As seen in Chapter 2, such a reachability game can be solved in time polynomial in the size of the finite graph, thus in EXP overall.

For the lower bound, we start by considering a game played over an exponential-time Turing machine, before showing how to implement this game as a countdown game. Let us consider for this an arbitrary Turing machine $\mathscr{M}$ working in deterministic exponential time $2^{p(n)}$ for some fixed polynomial p and an input word $w = a_1 \cdots a_n$ of length n, which we assume to be positive. Let $m \stackrel{\text{def}}{=} 2^{p(n)} \geq n$. The computation of $\mathscr{M}$ on w is a sequence of configurations $C_1, C_2, \ldots, C_t$ of length $t \leq m$. Each configuration C_i is of the form $\triangleright \gamma_{i,1} \cdots \gamma_{i,m} \triangleleft$ where $\triangleright$ and $\triangleleft$ are endmarkers and the symbols $\gamma_{i,j}$ are either taken from the finite tape alphabet Γ (which includes a blank symbol $\square$) or a pair (q,a) of a state from Q and a tape symbol a. We assume that the set of states Q

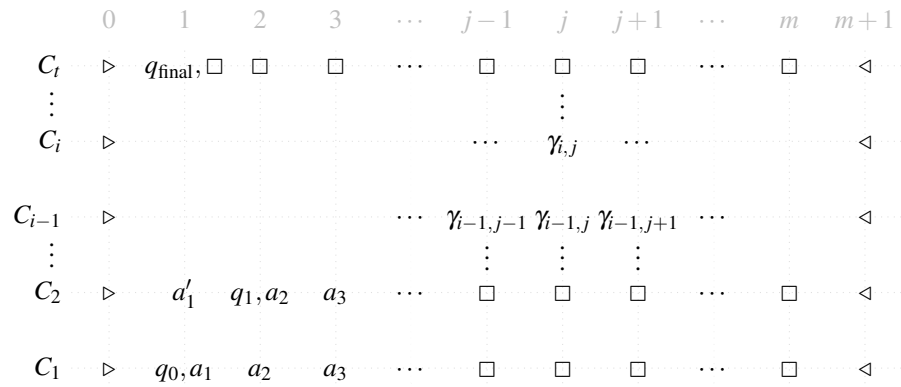


Figure 11.6: The computation of $\mathscr{M}$ on input $w = a_1 \cdots a_n$. This particular picture assumes $\mathscr{M}$ starts by rewriting a_1 into a'_1 and moving to the right in a state q_1, and empties its tape before accepting its input by going to state q_{final}.

contains a single accepting state q_{final}. The entire computation can be arranged over a $t \times m$ grid where each line corresponds to a configuration C_i, as shown in Figure 11.6.

We now set up a two-player game where Eve wants to prove that the input w is accepted. Let $\Gamma' \stackrel{\text{def}}{=} \{\triangleright, \triangleleft\} \cup \Gamma \cup (Q \times \Gamma)$. Rather than exhibiting the full computation from Figure 11.6, the game will be played over positions $(i, j, \gamma_{i,j})$ where $0 < i \leq m$, $0 \leq j \leq m+1$, and $\gamma_{i,j} \in \Gamma'$. Eve wants to show that, in the computation of $\mathscr{M}$ over w as depicted in Figure 11.6, the jth cell of the ith configuration C_i contains $\gamma_{i,j}$. In order to substantiate this claim, observe that the content of any cell $\gamma_{i,j}$ in the grid is determined by the actions of $\mathscr{M}$ and the contents of (up to) three cells in the previous configuration. Thus, if $i > 1$ and $0 < j < m+1$, Eve provides a triple $(\gamma_{i-1,j-1}, \gamma_{i-1,j}, \gamma_{i-1,j+1})$ of symbols in Γ' that yield $\gamma_{i,j}$ according to the actions of $\mathscr{M}$, which we denote by $\gamma_{i-1,j-1}, \gamma_{i-1,j}, \gamma_{i-1,j+1} \vdash \gamma_{i,j}$, and Adam chooses $j' \in \{j-1, j, j+1\}$ and returns the control to Eve in position $(i-1, j', \gamma_{i-1,j'})$. Regarding the boundary cases where $i = 0$ or $j = 0$ or $j = m+1$, Eve wins immediately if $j = 0$ and $\gamma = \triangleright$, or if $j = m+1$ and $\gamma = \triangleleft$, or if $i = 0$ and $0 < j \leq n$ and $\gamma = a_j$, or if $i = 0$ and $n < j \leq m$ and $\gamma = \square$, and otherwise Adam wins immediately. The game starts in a position $(t, j, (q_{\text{final}}, a))$ for some $0 < t \leq m$, $0 < j \leq m$, and $a \in \Gamma$ of Eve's choosing. It should be clear that Eve has a winning strategy in this game if and only if w is accepted by $\mathscr{M}$.

We now implement the previous game as a coverability game over a countdown system $\mathscr{V} \stackrel{\text{def}}{=} (\mathscr{L}, A, \mathscr{L}_{\text{Eve}}, \mathscr{L}_{\text{Adam}}, 1)$. The idea is that the pair (i, j) will be encoded as $(i-1) \cdot (m+2) + j + 2$ in the counter value, while the symbol $\gamma_{i,j}$ will be encoded in the location. For instance, the endmarker $\triangleright$ at position $(1,0)$ will be represented by configuration $\ell_{\triangleright}(2)$, the first input (q_0, a_1) at position $(1,1)$ by $\ell_{(q_0,a_1)}(3)$, and the endmarker $\triangleleft$ at position $(m, m+1)$ by $\ell_{\triangleleft}(m \cdot (m+2) + 1)$. The game starts from the initial configuration $\ell_0(n_0)$ where $n_0 \stackrel{\text{def}}{=} m \cdot (m+2) + 1$ and the target location is $\odot$.

We define for this the sets of locations

$$\mathscr{L}_{\text{Eve}} \stackrel{\text{def}}{=} \{\ell_0, ☺, ☹\} \cup \{\ell_\gamma \mid \gamma \in \Gamma'\} ,$$
$$\mathscr{L}_{\text{Adam}} \stackrel{\text{def}}{=} \{\ell_{(\gamma_1,\gamma_2,\gamma_3)} \mid \gamma_1,\gamma_2,\gamma_3 \in \Gamma'\} \cup \{\ell_{=j} \mid 0 < j \leq n\} \cup \{\ell_{1\leq ?\leq m-n+1}\} .$$

The intention behind the locations $\ell_{=j} \in \mathscr{L}_{\text{Adam}}$ is that Eve can reach ☺ from a configuration $\ell_{=j}(c)$ if and only if $c = j$; we accordingly define A with the following actions, where ☹ is a deadlock location:

$$\ell_{=j} \xrightarrow{-j-1} ☹ , \qquad \ell_{=j} \xrightarrow{-j} ☺ .$$

Similarly, Eve should be able to reach ☺ from $\ell_{1\leq ?\leq m-n+1}(c)$ if and only if $1 \leq c \leq m-n+1$, which is implemented by the actions

$$\ell_{1\leq ?\leq m-n+1} \xrightarrow{-m+n-2} ☹ , \qquad \ell_{1\leq ?\leq m-n+1} \xrightarrow{-1} ☺ , \qquad ☺ \xrightarrow{-1} ☺ .$$

Note this last action also ensures that Eve can reach the location ☺ if and only if she can reach the configuration $☺(0)$, thus the game can equivalently be seen as a configuration reachability game.

Regarding initialisation, Eve can choose her initial position, which we implement by the actions

$$\ell_0 \xrightarrow{-1} \ell_0 \qquad \ell_0 \xrightarrow{-1} \ell_{(q_{\text{final}},a)} \qquad \text{for } a \in \Gamma .$$

Outside the boundary cases, the game is implemented by the following actions:

$$\ell_\gamma \xrightarrow{-m} \ell_{(\gamma_1,\gamma_2,\gamma_3)} \qquad \text{for } \gamma_1,\gamma_2,\gamma_3 \vdash \gamma ,$$
$$\ell_{(\gamma_1,\gamma_2,\gamma_3)} \xrightarrow{-k} \ell_{\gamma_k} \qquad \text{for } k \in \{1,2,3\} .$$

We handle the endmarker positions via the following actions, where Eve proceeds along the left edge of Figure 11.6 until she reaches the initial left endmarker:

$$\ell_\triangleright \xrightarrow{-m-2} \ell_\triangleright , \qquad \ell_\triangleright \xrightarrow{-1} \ell_{=1} , \qquad \ell_\triangleleft \xrightarrow{-m-1} \ell_\triangleright .$$

For the positions inside the input word $w = a_1 \cdots a_n$, we use the actions

$$\ell_{(q_0,a_1)} \xrightarrow{-2} \ell_{=1} , \qquad \ell_{a_j} \xrightarrow{-2} \ell_{=j} \qquad \text{for } 1 < j \leq n .$$

Finally, for the blank symbols of C_1, which should be associated with a counter value c such that $n+3 \leq c \leq m+3$, we use the action

$$\ell_\square \xrightarrow{-n-2} \ell_{1\leq ?\leq m-n+1} .$$

□

Theorem 114 (Existential countdown games are EXPSPACE-complete)**.** *Configuration reachability and coverability countdown games with existential initial credit are* EXPSPACE*-complete.*

Proof. For the upper bound, consider an instance, i.e., a countdown system $\mathscr{V} = (\mathscr{L}, A, \mathscr{L}_{\text{Eve}}, \mathscr{L}_{\text{Adam}}, 1)$, an initial location ℓ_0, and a target configuration $\ell_f \in \mathscr{L}$. We reduce this to an instance of configuration reachability with given initial credit in a one-dimensional vector system by adding a new location ℓ'_0 controlled by Eve with actions $\ell'_0 \xrightarrow{1} \ell'_0$ and $\ell'_0 \xrightarrow{0} \ell_0$, and asking whether Eve has a winning strategy starting from $\ell'_0(0)$ in the new system. By Corollary 20, this configuration reachability game can be solved in EXPSPACE.

For the lower bound, we reduce from the acceptance problem of a deterministic Turing machine working in exponential space. The reduction is the same as in the proof of Theorem 113, except that now the length t of the computation is not bounded a priori, but this is compensated by the fact that we are playing the existential initial credit version of the countdown game. □

Originally, countdown games were introduced with a slightly different objective, which corresponds to the following decision problem.

Problem 17 (zero reachability with given initial credit)**.**

INPUT: *A countdown system $\mathscr{V} = (\mathscr{L}, T, \mathscr{L}_{\text{Eve}}, \mathscr{L}_{\text{Adam}}, 1)$, an initial location $\ell_0 \in \mathscr{L}$, and an initial credit $n_0 \in \mathbb{N}$.*

OUTPUT: *Does Eve have a strategy to reach a configuration $\ell(0)$ for some $\ell \in \mathscr{L}$? That is, does she win the* zero reachability *game $(\mathscr{A}_{\mathbb{N}}(\mathscr{V}), \mathfrak{c}, \texttt{Reach})$ from $\ell_0(n_0)$, where $\mathfrak{c}(e) =$ Win if and only if $In(e) = \ell(0)$ for some $\ell \in \mathscr{L}$?*

Theorem 115 (Countdown to zero games are EXP-complete)**.** *Zero reachability countdown games with given initial credit are* EXP*-complete.*

Proof. The upper bound of Theorem 113 applies in the same way. Regarding the lower bound, we modify the lower bound construction of Theorem 113 in the following way: we use $\ell_0(2 \cdot n_0 + 1)$ as initial configuration, multiply all the action weights in A by two, and add a new location ℓ_{zero} with an action $\ominus \xrightarrow{-1} \ell_{\text{zero}}$. Because all the counter values in the new game are odd unless we reach ℓ_{zero}, the only way for Eve to bring the counter to zero in this new game is to first reach $\ominus(1)$, which occurs if and only if she could reach $\ominus(0)$ in the original game. □

11.2.2 Vector Games in Dimension One

Countdown games are frequently employed to prove complexity lower bounds. Here, we use them to show that the EXPSPACE upper bounds from Corollary 20 are tight in most cases.

Theorem 116 (The complexity of vector games in dimension one)**.** *Configuration reachability, coverability, and parity vector games, both with given and with existential initial credit, are* EXPSPACE*-complete in dimension one; non-termination vector games in dimension one are* EXP*-hard with given initial credit and* EXPSPACE*-complete with existential initial credit.*

Proof. By Theorem 114, configuration reachability and coverability vector games with existential initial credit are EXPSPACE-hard in dimension one. Furthermore, Remark 22 allows to deduce that parity is also EXPSPACE-hard. Finally, we can argue as in the upper bound proof of Theorem 114 that all these games are also hard with given initial credit: we add a new initial location ℓ'_0 controlled by Eve with actions $\ell'_0 \xrightarrow{1} \ell'_0$ and $\ell'_0 \xrightarrow{0} \ell_0$ and play the game starting from $\ell'_0(0)$.

Regarding non-termination, we can add a self loop $\☺ \xrightarrow{0} \☺$ to the construction of Theorems 113 and 114: then the only way to build an infinite play that avoids the sink is to reach the target location ☺. This shows that the games are EXP-hard with given initial credit and EXPSPACE-hard with existential initial credit. Note that the trick of reducing existential to given initial credit with an initial incrementing loop $\ell'_0 \xrightarrow{1} \ell'_0$ does not work, because Eve would have a trivial winning strategy that consists in just playing this loop forever. □

11.3 Asymmetric games

Theorem 112 shows that vector games are too powerful to be algorithmically relevant, except in dimension one where Theorem 116 applies. This prompts the study of restricted kinds of vector games, which might be decidable in arbitrary dimension. This section introduces one such restriction, called *asymmetry*, which turns out to be very fruitful: it yields decidable games (see Section 11.5), and is related to another class of games on counter systems called energy games (see Section 11.4).

Asymmetric Games A vector system $\mathscr{V} = (\mathscr{L}, A, \mathscr{L}_{\text{Eve}}, \mathscr{L}_{\text{Adam}}, k)$ is *asymmetric* if, for all locations $\ell \in \mathscr{L}_{\text{Adam}}$ controlled by Adam and all actions $(\ell \xrightarrow{\vec{u}} \ell') \in A$ originating from those, $\vec{u} = \vec{0}$ the zero vector. In other words, Adam may only change the current location, and cannot interact directly with the counters.

Example 10 (Asymmetric vector system)**.** *Figure 11.7 presents an asymmetric vector system of dimension two with locations partitioned as* $\mathscr{L}_{\text{Eve}} = \{\ell, \ell_{2,1}, \ell_{\text{-}1,0}\}$ *and* $\mathscr{L}_{\text{Adam}} = \{\ell'\}$*. We omit the labels on the actions originating from Adamś locations, since those are necessarily the zero vector. It is worth observing that this vector system behaves quite differently from the one of Example 7 on p. 356: for instance, in* $\ell'(0,1)$*, Adam can now ensure that the sink will be reached by playing the action* $\ell' \xrightarrow{0,0} \ell_{\text{-}1,0}$*, whereas in Example 7, the action* $\ell' \xrightarrow{-1,0} \ell$ *was just inhibited by the natural semantics.*

11.3.1 The Case of Configuration Reachability

In spite of the restriction to asymmetric vector systems, configuration reachability remains undecidable.

Theorem 117 (Reachability in asymmetric vector games is undecidable)**.** *Configuration reachability asymmetric vector games, both with given and with existential initial credit, are undecidable in any dimension* $k \geq 2$*.*

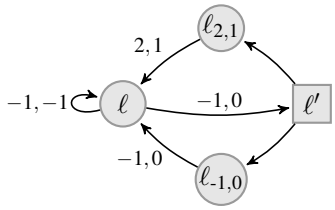


Figure 11.7: An asymmetric vector system.

Proof. We first reduce from the halting problem of deterministic Minsky machines to configuration reachability with given initial credit. Given an instance of the halting problem, i.e., given $\mathscr{M} = (\mathscr{L}, A, k)$ and an initial location ℓ_0 where we assume without loss of generality that $\mathscr{M}$ checks that all its counters are zero before going to $\ell_{\texttt{halt}}$, we construct an asymmetric vector system $\mathscr{V} \stackrel{\text{def}}{=} (\mathscr{L} \uplus \mathscr{L}_{\overline{=}^?0} \uplus \mathscr{L}_k, A', \mathscr{L} \uplus \mathscr{L}'_{\overline{=}^?0}, \mathscr{L}_{\overline{=}^?0}, k)$ where all the original locations and $\mathscr{L}_k$ are controlled by Eve and $\mathscr{L}_{\overline{=}^?0}$ is controlled by Adam.

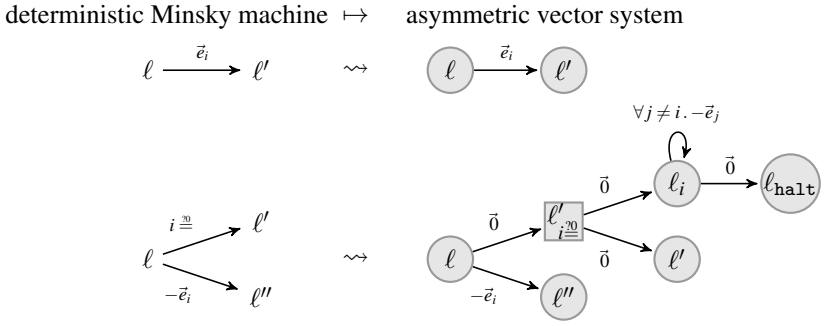


Figure 11.8: Schema of the reduction in the proof of Theorem 117.

We use $\mathscr{L}_{\overline{=}^?0}$ and $\mathscr{L}_k$ as two sets of locations disjoint from $\mathscr{L}$ defined by

$$\mathscr{L}_{\overline{=}^?0} \stackrel{\text{def}}{=} \{\ell'_{i\overline{=}^?0} \in \mathscr{L} \times \{1,\ldots,k\} \mid \exists \ell \in \mathscr{L} \,.\, (\ell \xrightarrow{i \overline{=}^? 0} \ell') \in A\}$$
$$\mathscr{L}_k \stackrel{\text{def}}{=} \{\ell_i \mid 1 \leq i \leq k\}$$

and define the set of actions by (see Figure 11.8)

$$\begin{aligned} A' \stackrel{\text{def}}{=} & \{\ell \xrightarrow{\vec{e}_i} \ell' \mid (\ell \xrightarrow{\vec{e}_i} \ell') \in A\} \cup \{\ell \xrightarrow{-\vec{e}_i} \ell'' \mid (\ell \xrightarrow{-\vec{e}_i} \ell'') \in A\} \\ & \cup \{\ell \xrightarrow{\vec{0}} \ell'_{i\overline{=}^?0},\ \ell'_{i\overline{=}^?0} \xrightarrow{\vec{0}} \ell',\ \ell'_{i\overline{=}^?0} \xrightarrow{\vec{0}} \ell_i \mid (\ell \xrightarrow{i \overline{=}^? 0} \ell') \in A\} \\ & \cup \{\ell_i \xrightarrow{-\vec{e}_j} \ell_i,\ \ell_i \xrightarrow{\vec{0}} \ell_{\texttt{halt}} \mid 1 \leq i,j \leq k, j \neq i\} \,. \end{aligned}$$

We use $\ell_0(\vec{0})$ as initial configuration and $\ell_{\texttt{halt}}(\vec{0})$ as target configuration for the constructed configuration reachability instance. Here is the crux of the argument why Eve has a winning strategy to reach $\ell_{\texttt{halt}}(\vec{0})$ from $\ell_0(\vec{0})$ if and only if the Minsky machine halts, i.e., if and only if the Minsky machine reaches $\ell_{\texttt{halt}}(\vec{0})$. Consider any configuration $\ell(\vec{v})$. If $(\ell \xrightarrow{\vec{e}_i} \ell') \in A$, Eve has no choice but to apply $\ell \xrightarrow{\vec{e}_i} \ell'$ and go to the

configuration $\ell'(\vec{v}+\vec{e}_i)$ also reached in one step in $\mathscr{M}$. If $\{\ell \xrightarrow[i=]{i \overset{?}{=} 0} \ell', \ell \xrightarrow{-\vec{e}_i} \ell''\} \in A$ and $\vec{v}(i) = 0$, due to the natural semantics, Eve cannot use the action $\ell \xrightarrow{-\vec{e}_i} \ell''$, thus she must use $\ell \xrightarrow{0} \ell'_{i \overset{?}{=} 0}$. Then, either Adam plays $\ell'_{i \overset{?}{=} 0} \xrightarrow{0} \ell'$ and Eve regains the control in $\ell'(\vec{v})$, which was also the configuration reached in one step in $\mathscr{M}$, or Adam plays $\ell'_{i \overset{?}{=} 0} \xrightarrow{0} \ell_i$ and Eve regains the control in $\ell_i(\vec{v})$ with $\vec{v}(i) = 0$. Using the actions $\ell_{i \overset{?}{=} 0} \xrightarrow{-\vec{e}_j} \ell_i$ for $j \neq i$, Eve can then reach $\ell_i(\vec{0})$ and move to $\ell_{\texttt{halt}}(\vec{0})$. Finally, if $\{\ell \xrightarrow[i=]{i \overset{?}{=} 0} \ell', \ell \xrightarrow{-\vec{e}_i} \ell''\} \in A$ and $\vec{v}(i) > 0$, Eve can choose: if she uses $\ell \xrightarrow{-\vec{e}_i} \ell''$, she ends in the configuration $\ell''(\vec{v}-\vec{e}_i)$ also reached in one step in $\mathscr{M}$. In fact, she should not use $\ell \xrightarrow{0} \ell'_{i \overset{?}{=} 0}$, because Adam would then have the opportunity to apply $\ell'_{i \overset{?}{=} 0} \xrightarrow{0} \ell_i$, and in $\ell_i(\vec{v})$ with $\vec{v}(i) > 0$, there is no way to reach a configuration with an empty ith component, let alone to reach $\ell_{\texttt{halt}}(\vec{0})$. Thus, if $\mathscr{M}$ halts, then Eve has a winning strategy that mimics the unique play of $\mathscr{M}$, and conversely, if Eve wins, then necessarily she had to follow the play of $\mathscr{M}$ and thus the machine halts.

Finally, regarding the existential initial credit variant, the arguments used in the proof of Theorem 112 apply similarly to show that it is also undecidable. □

In dimension one, Theorem 116 applies, thus configuration reachability is decidable in EXPSPACE. This bound is actually tight.

Theorem 118 (Asymmetric vector games are EXPSPACE-complete in dimension one)**.** *Configuration reachability asymmetric vector games, both with given and with existential initial credit, are* EXPSPACE*-complete in dimension one.*

Proof. Let us first consider the existential initial credit variant. We proceed as in Theorems 113 and 114 and reduce from the acceptance problem for a deterministic Turing machine working in exponential space $m = 2^{p(n)}$. The reduction is mostly the same as in Theorem 113, with a few changes. Consider the integer $m-n$ from that reduction. While this is an exponential value, it can be written as $m-n = \sum_{0 \leq e \leq p(n)} 2^e \cdot b_e$ for a polynomial number of bits $b_0, \ldots, b_{p(n)}$. For all $0 \leq d \leq p(n)$, we define $m_d \overset{\text{def}}{=} \sum_{0 \leq e \leq d} 2^e \cdot b_e$; thus $m-n+1 = m_{p(n)} + 1$.

We define now the sets of locations

$$\begin{aligned}\mathscr{L}_{\text{Eve}} &\overset{\text{def}}{=} \{\ell_0, ☺\} \cup \{\ell_\gamma \mid \gamma \in \Gamma'\} \cup \{\ell_\gamma^k \mid 1 \leq k \leq 3\} \cup \{\ell_{=j} \mid 0 < j \leq n\} \\ &\quad \cup \{\ell_{1 \leq ? \leq m_d+1} \mid 0 \leq d \leq p(n)\} \cup \{\ell_{1 \leq ? \leq 2^d} \mid 1 \leq d \leq p(n)\}\,, \\ \mathscr{L}_{\text{Adam}} &\overset{\text{def}}{=} \{\ell_{(\gamma_1,\gamma_2,\gamma_3)} \mid \gamma_1, \gamma_2, \gamma_3 \in \Gamma'\}\,.\end{aligned}$$

The intention behind the locations $\ell_{=j} \in \mathscr{L}_{\text{Eve}}$ is that Eve can reach $☺(0)$ from a configuration $\ell_{=j}(c)$ if and only if $c = j$; we define accordingly A with the action $\ell_{=j} \xrightarrow{-j} ☺$. Similarly, Eve should be able to reach $☺(0)$ from $\ell_{1 \leq ? \leq m_d+1}(c)$ for $0 \leq d \leq p(n)$ if and only if $1 \leq c \leq m_d + 1$, which is implemented by the following actions: if $b_{d+1} = 1$, then

$$\ell_{1 \leq ? \leq m_{d+1}+1} \xrightarrow{0} \ell_{1 \leq ? \leq 2^{d+1}}\,, \qquad \ell_{1 \leq ? \leq m_{d+1}+1} \xrightarrow{-2^{d+1}} \ell_{1 \leq ? \leq m_d+1}\,,$$

and if $b_{d+1} = 0$,

$$\ell_{1 \leq ? \leq m_{d+1}+1} \xrightarrow{0} \ell_{1 \leq ? \leq m_d+1}\,,$$

and finally

$$\ell_{1\leq ?\leq m_0+1} \xrightarrow{-b_0} \ell_{=1}\ , \qquad\qquad \ell_{1\leq ?\leq m_0+1} \xrightarrow{0} \ell_{=1}\ ,$$

where for all $1 \leq d \leq p(n)$, $\ell_{1\leq ?\leq 2^d}(c)$ allows to reach $\ominus(0)$ if and only if $1 \leq c \leq 2^d$:

$$\begin{array}{ll} \ell_{1\leq ?\leq 2^{d+1}} \xrightarrow{-2^d} \ell_{1\leq ?\leq 2^d}\ , & \ell_{1\leq ?\leq 2^{d+1}} \xrightarrow{0} \ell_{1\leq ?\leq 2^d}\ , \\ \ell_{1\leq ?\leq 2^1} \xrightarrow{-1} \ell_{=1}\ , & \ell_{1\leq ?\leq 2^1} \xrightarrow{0} \ell_{=1}\ . \end{array}$$

The remainder of the reduction is now very similar to the reduction shown in Theorem 113. Regarding initialisation, Eve can choose her initial position, which we implement by the actions

$$\ell_0 \xrightarrow{-1} \ell_0 \qquad\qquad \ell_0 \xrightarrow{-1} \ell_{(q_{\text{final}},a)} \qquad \text{for } a \in \Gamma\ .$$

Outside the boundary cases, the game is implemented by the following actions:

$$\begin{array}{lll} \ell_\gamma \xrightarrow{-m} \ell_{(\gamma_1,\gamma_2,\gamma_3)} & & \text{for } \gamma_1,\gamma_2,\gamma_3 \vdash \gamma\ , \\ \ell_{(\gamma_1,\gamma_2,\gamma_3)} \xrightarrow{0} \ell^k_{\gamma_k} & \ell^k_{\gamma_k} \xrightarrow{-k} \ell_{\gamma_k} & \text{for } k \in \{1,2,3\}\ . \end{array}$$

We handle the endmarker positions via the following actions, where Eve proceeds along the left edge of Figure 11.6 until she reaches the initial left endmarker:

$$\ell_\triangleright \xrightarrow{-m-2} \ell_\triangleright\ , \qquad\qquad \ell_\triangleright \xrightarrow{-1} \ell_{=1}\ , \qquad\qquad \ell_\triangleleft \xrightarrow{-m-1} \ell_\triangleright\ .$$

For the positions inside the input word $w = a_1 \cdots a_n$, we use the actions

$$\ell_{(q_0,a_1)} \xrightarrow{-2} \ell_{=1}\ , \qquad\qquad \ell_{a_j} \xrightarrow{-2} \ell_{=j} \qquad \text{for } 1 < j \leq n\ .$$

Finally, for the blank symbols of C_1, which should be associated with a counter value c such that $n+3 \leq c \leq m+3$, i.e., such that $1 \leq c-n-2 \leq m-n+1 = m_{p(n)}+1$, we use the action

$$\ell_\square \xrightarrow{-n-2} \ell_{1\leq ?\leq m_{p(n)}+1}\ .$$

Regarding the given initial credit variant, we add a new location ℓ'_0 controlled by Eve and let her choose her initial credit when starting from $\ell'_0(0)$ by using the new actions $\ell'_0 \xrightarrow{1} \ell'_0$ and $\ell'_0 \xrightarrow{0} \ell_0$. □

11.3.2 Asymmetric Monotone Games

The results on configuration reachability might give the impression that asymmetry does not help much for solving vector games: we obtained in Section 11.3.1 exactly the same results as in the general case. Thankfully, the situation changes drastically if we consider the other types of vector games: coverability, non-termination, and parity become decidable in asymmetric vector games. The main rationale for this comes from order theory, which prompts the following definitions.

Quasi-orders A *quasi-order* $(X, \leq)$ is a set X together with a reflexive and transitive relation $\leq \subseteq X \times X$. Two elements $x, y \in X$ are incomparable if $x \not\leq y$ and $y \not\leq x$, and they are equivalent if $x \leq y$ and $y \leq x$. The associated strict relation $x < y$ holds if $x \leq y$ and $y \not\leq x$.

The *upward closure* of a subset $S \subseteq X$ is the set of elements greater or equal to the elements of S: $\uparrow S \stackrel{\text{def}}{=} \{x \in X \mid \exists y \in S \,.\, y \leq x\}$. A subset $U \subseteq X$ is *upwards closed* if $\uparrow U = U$. When $S = \{x\}$ is a singleton, we write more simply $\uparrow x$ for its upward closure and call the resulting upwards closed set a *principal filter*. Dually, the *downward closure* of S is $\downarrow S \stackrel{\text{def}}{=} \{x \in X \mid \exists y \in S \,.\, x \leq y\}$, a *downwards closed* set is a subset $D \subseteq X$ such that $D = \downarrow D$, and $\downarrow x$ is called a *principal ideal*. Note that the complement $X \setminus U$ of an upwards closed set U is downwards closed and vice versa.

Monotone Games Let us consider again the natural semantics $\mathscr{A}_{\mathbb{N}}(\mathscr{V})$ of a vector system. The set of vertices $V = \mathscr{L} \times \mathbb{N}^k \cup \{\bot\}$ is naturally equipped with a partial ordering: $v \leq v'$ if either $v = v' = \bot$, or $v = \ell(\vec{v})$ and $v' = \ell(\vec{v}')$ are two configurations that share the same location and satisfy $\vec{v}(i) \leq \vec{v}'(i)$ for all $1 \leq i \leq k$, i.e., if $\vec{v} \leq \vec{v}'$ for the componentwise ordering.

Consider a set of colours C and a vertex colouring $\mathrm{vcol}: V \to C$ of the natural semantics $\mathscr{A}_{\mathbb{N}}(\mathscr{V})$ of a vector system, which defines a colouring $\mathfrak{c}: E \to C$ where $\mathfrak{c}(e) \stackrel{\text{def}}{=} \mathrm{vcol}(\mathrm{In}(e))$. We say that the colouring vcol is *monotonic* if C is finite and, for every colour $p \in C$, the set $\mathrm{vcol}^{-1}(p)$ of vertices coloured by p is upwards closed with respect to $\leq$. Clearly, the colourings of coverability, non-termination, and parity vector games are monotonic, whereas those of configuration reachability vector games are not. By extension, we call a vector game *monotonic* if its underlying colouring is monotonic.

Lemma 132 (Simulation)**.** *In a monotonic asymmetric vector game, if Eve wins from a vertex v_0, then she also wins from v_0' for all $v_0' \geq v_0$.*

Proof. It suffices for this to check that, for all $v_1 \leq v_2$ in V,

(colours) $\mathrm{vcol}(v_1) = \mathrm{vcol}(v_2)$ since vcol is monotonic;

(zig Eve) if $v_1, v_2 \in V_{\mathrm{Eve}}$, $a \in A$, and $\Delta(v_1, a) = v_1' \neq \bot$ is defined, then $v_2' \stackrel{\text{def}}{=} \Delta(v_2, a)$ is such that $v_2' \geq v_1'$: indeed, $v_1' \neq \bot$ entails that v_1 is a configuration $\ell(\vec{v}_1)$ and $v_1' = \ell'(\vec{v}_1 + \vec{u})$ for the action $a = (\ell \xrightarrow{\vec{u}} \ell') \in A$, but then $v_2 = \ell(\vec{v}_2)$ for some $\vec{v}_2 \geq \vec{v}_1$ and $v_2' = \ell'(\vec{v}_2 + \vec{u}) \geq v_1'$;

(zig Adam) if $v_1, v_2 \in V_{\mathrm{Adam}}$, $a \in A$, and $\Delta(v_2, a) = v_2'$ is defined, then $v_1' \stackrel{\text{def}}{=} \Delta(v_1, a) \leq v_2'$: indeed, either $v_2' = \bot$ and then $v_1' = \bot$, or $v_2' \neq \bot$, thus $v_2 = \ell(\vec{v}_2)$, $v_2' = \ell'(\vec{v}_2)$, and $a = (\ell \xrightarrow{\vec{0}} \ell') \in A$ (recall that the game is asymmetric), but then $v_1 = \ell(\vec{v}_1)$ for some $\vec{v}_1 \leq \vec{v}_2$ and thus $v_1' = \ell'(\vec{v}_1) \leq v_2'$.

The above conditions show that, if $\sigma: E^* \to A$ is a strategy of Eve that wins from v_0, then by *simulating* σ starting from v_0'—i.e., by applying the same actions when given a pointwise larger or equal history—she will also win. □

Note that Lemma 132 implies that W_{Eve} is upwards closed: $v_0 \in W_{\mathrm{Eve}}$ and $v_0 \leq v_0'$ imply $v_0' \in W_{\mathrm{Eve}}$. Lemma 132 does not necessarily hold in vector games without the

asymmetry condition. For instance, in both Figures 11.3 and 11.4 on p. 359, $\ell'(0,1) \in W_{\text{Eve}}$ but $\ell'(1,2) \in W_{\text{Adam}}$ for the coverability and non-termination objectives. This is due to the fact that the action $\ell' \xrightarrow{-1,0} \ell$ is available in $\ell'(1,2)$ but not in $\ell'(0,1)$.

Well-quasi-orders What makes monotonic vector games so interesting is that the partial order $(V,\leq)$ associated with the natural semantics of a vector system is a *well-quasi-order*. A quasi-order $(X,\leq)$ is well (a *wqo*) if any of the following equivalent characterisations hold [Kru72, SS12]:

- in any infinite sequence $x_0, x_1, \cdots$ of elements of X, there exists an infinite sequence of indices $n_0 < n_1 < \cdots$ such that $x_{n_0} \leq x_{n_1} \leq \cdots$—infinite sequences in X are *good*—,
- any strictly ascending sequence $U_0 \subsetneq U_1 \subsetneq \cdots$ of upwards closed sets $U_i \subseteq X$ is finite—X has the *ascending chain condition*—,
- any non-empty upwards closed $U \subseteq X$ has at least one, and at most finitely many minimal elements up to equivalence; therefore any upwards closed $U \subseteq X$ is a finite union $U = \bigcup_{1\leq j\leq n} \uparrow x_j$ of finitely many principal filters $\uparrow x_j$—X has the *finite basis property*.

The fact that $(V,\leq)$ satisfies all of the above is an easy consequence of *Dickson's Lemma* [Dic13].

Pareto Limits By the finite basis property of $(V,\leq)$ and Lemma 132, in a monotonic asymmetric vector game, $W_{\text{Eve}} = \bigcup_{1\leq j\leq n} \uparrow \ell_j(\vec{v}_j)$ is a finite union of principal filters. The set $\mathsf{Pareto} \stackrel{\text{def}}{=} \{\ell_1(\vec{v}_1), \ldots, \ell_n(\vec{v}_n)\}$ is called the *Pareto limit* or *Pareto frontier* of the game. Both the existential and the given initial credit variants of the game can be reduced to computing this Pareto limit: with existential initial credit and an initial location ℓ_0, check whether $\ell_0(\vec{v}) \in \mathsf{Pareto}$ for some $\vec{v}$, and with given initial credit and an initial configuration $\ell_0(\vec{v}_0)$, check whether $\ell_0(\vec{v}) \in \mathsf{Pareto}$ for some $\vec{v} \leq \vec{v}_0$.

Example 11 (Pareto limit)**.** *Consider the asymmetric vector system from Figure 11.7 on p. 368. For the coverability game with target configuration $\ell(2,2)$, the Pareto limit is* $\mathsf{Pareto} = \{\ell(2,2), \ell'(3,2), \ell_{2,1}(0,1), \ell_{\text{-}1,0}(3,2)\}$*, while for the non-termination game,* $\mathsf{Pareto} = \emptyset$*: Eve loses from all the vertices. Observe that this is consistent with Eve's winning region in the coverability energy game shown in Figure 11.12.*

Example 12 (Doubly exponential Pareto limit)**.** *Consider the one-player vector system of Figure 11.9, where the meta-decrement from ℓ_0 to ℓ_1 can be implemented using $O(n)$ additional counters and a set $\mathscr{L}'$ of $O(n)$ additional locations by the arguments of the forthcoming Theorem 125.*

For the coverability game with target configuration $\ell_f(\vec{0})$, if ℓ_0 is the initial location and we are given initial credit $m \cdot \vec{e}_1$, Eve wins if and only if $m \geq 2^{2^n}$, but with existential initial credit she can start from $\ell_0(\vec{e}_2)$ instead. We have indeed $\mathsf{Pareto} \cap (\{\ell_0, \ell_1, \ell_f\} \times \mathbb{N}^k) = \{\ell_0(\vec{e}_2), \ell_0(2^{2^n} \cdot \vec{e}_1), \ell_1(\vec{0}), \ell_f(\vec{0})\}$*. Looking more in-depth into the construction of Theorem 125, there is also an at least double exponential number of distinct minimal configurations in* Pareto.

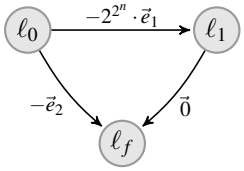


Figure 11.9: A one-player vector system with a large Pareto limit.

Finite Memory Besides having a finitely represented winning region, Eve also has finite memory strategies in asymmetric vector games with parity objectives; the following argument is straightforward to adapt to the other regular objectives from Chapter 2.

Lemma 133 (Finite memory suffices in parity asymmetric vector games)**.** *If Eve has a strategy winning from some vertex v_0 in a parity asymmetric vector game, then she has a finite-memory one.*

Proof. Assume σ is a winning strategy from v_0. Consider the tree of vertices visited by plays consistent with σ: each branch is an infinite sequence $v_0, v_1, \dots$ of elements of V where the maximal priority occuring infinitely often is some even number p. Since $(V, \leq)$ is a wqo, this is a good sequence: there exists infinitely many indices $n_0 < n_1 < \cdots$ such that $v_{n_0} \leq v_{n_1} \leq \cdots$. There exists $i < j$ such that $p = \max_{n_i \leq n < n_j} \mathrm{vcol}(v_n)$ is the maximal priority occurring in some interval $v_{n_i}, v_{n_{i+1}}, \dots, v_{n_{j-1}}$. Then Eve can play in v_{n_j} as if she were in v_{n_i}, in v_{n_j+1} as if she were in v_{n_i+1} and so on, and we prune the tree at index n_j along this branch so that v_{n_j} is a leaf, and we call v_{n_i} the *return node* of that leaf. We therefore obtain a finitely branching tree with finite branches, which by König's Lemma is finite.

The finite tree we obtain this way is sometimes called a *self-covering tree*. It is relatively straightforward to construct a finite memory structure (M, m_0, δ) (as defined in Section 1.5) from a self-covering tree, using its internal nodes as memory states plus an additional sink memory state $m_\perp$; the initial memory state m_0 is the root of the tree. In a node m labelled by $\ell(\vec{v})$, given an edge $e = (\ell(\vec{v}'), \ell'(\vec{v}' + \vec{u}))$ arising from an action $\ell \xrightarrow{\vec{u}} \ell' \in A$, if $\vec{v}' \geq \vec{v}$ and m has a child m' labelled by $\ell'(\vec{v} + \vec{u})$ in the self-covering tree, then either m' is a leaf with return node m'' and we set $\delta(m, e) \stackrel{\text{def}}{=} m''$, or m' is an internal node and we set $\delta(m, e) \stackrel{\text{def}}{=} m'$; in all the other cases, $\delta(m, e) \stackrel{\text{def}}{=} m_\perp$. □

Example 13 (doubly exponential memory)**.** *Consider the one-player vector system of Figure 11.10, where the meta-decrement from ℓ_1 to ℓ_0 can be implemented using $O(n)$ additional counters and $O(n)$ additional locations by the arguments of the forthcoming Theorem 125 on p. 387.*

For the parity game with location colouring $\mathrm{lcol}(\ell_0) \stackrel{def}{=} 2$ *and* $\mathrm{lcol}(\ell_1) \stackrel{def}{=} 1$*, note that Eve must visit ℓ_0 infinitely often in order to fulfil the parity requirements. Starting from the initial configuration $\ell_0(\vec{0})$, any winning play of Eve begins by*

$$\ell_0(\vec{0}) \xrightarrow{0} \ell_1(\vec{0}) \xrightarrow{\vec{e}_1} \ell_1(\vec{e}_1) \xrightarrow{\vec{e}_1} \cdots \xrightarrow{\vec{e}_1} \ell_1(m \cdot \vec{e}_1) \xrightarrow{-2^{2^n}} \ell_0((m - 2^{2^n}) \cdot \vec{e}_1)$$

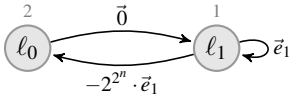


Figure 11.10: A one-player vector system witnessing the need for double exponential memory.

*for some $m \geq 2^{2^n}$ before she visits again a configuration—namely $\ell_0((m-2^{2^n}) \cdot \vec{e}_1)$—greater or equal than a previous configuration—namely $\ell_0(\vec{0})$—*and *witnesses a maximal even parity in the meantime. She then has a winning strategy that simply repeats this sequence of actions, allowing her to visit successively $\ell_0(2(m-2^{2^n}) \cdot \vec{e}_1)$, $\ell_0(3(m-2^{2^n}) \cdot \vec{e}_1)$, etc. In this example, she needs at least 2^{2^n} memory to remember how many times the $\ell_1 \xrightarrow{\vec{e}_1} \ell_1$ loop should be taken.*

Attractor Computation for Coverability

So far, we have not seen how to compute the Pareto limit derived from Lemma 132 nor the finite memory structure derived from Lemma 133. These objects are not merely finite but also computable. The simplest case is the one of coverability asymmetric monotonic vector games: the fixed point computation of Section 2.1 for reachability objectives can be turned into an algorithm computing the Pareto limit of the game.

Fact 20 (Computable Pareto limit)**.** *The Pareto limit of a coverability asymmetric vector game is computable.*

Proof. Let $\ell_f(\vec{v}_f)$ be the target configuration. We define a chain $U_0 \subseteq U_1 \subseteq \cdots$ of sets $U_i \subseteq V$ by

$$U_0 \stackrel{\text{def}}{=} \uparrow \ell_f(\vec{v}_f)\ , \qquad\qquad U_{i+1} \stackrel{\text{def}}{=} U_i \cup \mathrm{Pre}(U_i)\ .$$

Observe that for all i, U_i is upwards closed. This can be checked by induction over i: it holds initially in U_0, and for the induction step, if $v \in U_{i+1}$ and $v' \geq v$, then either

- $v = \ell(\vec{v}) \in \mathrm{Pre}(U_i) \cap V_{\mathrm{Eve}}$ thanks to some $\ell \xrightarrow{\vec{u}} \ell' \in A$ such that $\ell'(\vec{v}+\vec{u}) \in U_i$; therefore $v' = \ell(\vec{v}')$ for some $\vec{v}' \geq \vec{v}$ is such that $\ell'(\vec{v}'+\vec{u}) \in U_i$ as well, thus $v' \in \mathrm{Pre}(U_i) \subseteq U_{i+1}$, or
- $v = \ell(\vec{v}) \in \mathrm{Pre}(U_i) \cap V_{\mathrm{Adam}}$ because for all $\ell \xrightarrow{\vec{0}} \ell' \in A$, $\ell'(\vec{v}) \in U_i$; therefore $v' = \ell(\vec{v}')$ for some $\vec{v}' \geq \vec{v}$ is such that $\ell'(\vec{v}') \in U_i$ as well, thus $v' \in \mathrm{Pre}(U_i) \subseteq U_{i+1}$, or
- $v \in U_i$ and therefore $v' \in U_i \subseteq U_{i+1}$.

By the ascending chain condition, there is a finite rank i such that $U_{i+1} \subseteq U_i$ and then $W_{\mathrm{Eve}} = U_i$. Thus the Pareto limit is obtained after finitely many steps. In order to turn this idea into an algorithm, we need a way of representing those infinite upwards

closed sets U_i. Thankfully, by the finite basis property, each U_i has a finite basis B_i such that $\uparrow B_i = U_i$. We therefore compute the following sequence of sets

$$B_0 \stackrel{\text{def}}{=} \{\ell_f(\vec{v}_f)\} \qquad\qquad B_{i+1} \stackrel{\text{def}}{=} B_i \cup \min \mathrm{Pre}(\uparrow B_i)\ .$$

Indeed, given a finite basis B_i for U_i, it is straightforward to compute a finite basis for the upwards closed $\mathrm{Pre}(U_i)$. This results in Algorithm 11.1 below. □

Algorithm 11.1: Fixed point algorithm for coverability in asymmetric vector games.

Data: A vector system and a target configuration $\ell_f(\vec{v}_f)$
$B_0 \leftarrow \{\ell_f(\vec{v}_f)\}$;
$i \leftarrow 0$;
repeat
  $B_{i+1} \leftarrow B_i \cup \min \mathrm{Pre}(\uparrow B_i)$;
  $i \leftarrow i+1$;
until $\uparrow B_i \supseteq B_{i+1}$;
return $\min B_i = \mathsf{Pareto}(\mathscr{G})$

While this algorithm terminates thanks to the ascending chain condition, it may take quite long. For instance, in Example 12, it requires at least 2^{2^n} steps before it reaches its fixed point. This is a worst-case instance, as it turns out that this algorithm works in 2EXP; see the bibliographic notes at the end of the chapter. Note that such a fixed point computation does not work directly for non-termination or parity vector games, due to the need for greatest fixed points.

11.4 Resource-conscious games

Vector games are very well suited for reasoning about systems manipulating discrete resources, modelled as counters. However, in the natural semantics, actions that would deplete some resource, i.e., that would make some counter go negative, are simply inhibited. In models of real-world systems monitoring resources like a gas tank or a battery, a depleted resource would be considered as a system failure. In the energy games of Section 11.4.1, those situations are accordingly considered as winning for Adam. Moreover, if we are modelling systems with a bounded capacity for storing resources, a counter exceeding some bound might also be considered as a failure, which will be considered with bounding games in Section 11.4.2.

These resource-conscious games can be seen as providing alternative semantics for vector systems. They will also be instrumental in establishing complexity upper bounds for monotonic asymmetric vector games later in Section 11.5, and are strongly related to multidimensional mean payoff games, as will be explained in Section 12.2 of Chapter 12.

11.4.1 Energy Semantics

Energy games model systems where the depletion of a resource allows Adam to win. This is captured by an *energy semantics* $\mathscr{A}_{\mathbb{E}}(\mathscr{V}) \stackrel{\text{def}}{=} (V, E_{\mathbb{E}}, V_{\text{Eve}}, V_{\text{Adam}})$ associated with a vector system $\mathscr{V}$: we let as before $V \stackrel{\text{def}}{=} (\mathscr{L} \times \mathbb{N}^k) \uplus \{\bot\}$, but define instead

$$E_{\mathbb{E}} \stackrel{\text{def}}{=} \{(\ell(\vec{v}), \ell'(\vec{v}+\vec{u}) \mid \ell \xrightarrow{\vec{u}} \ell' \in A \text{ and } \vec{v}+\vec{u} \geq \vec{0}\}$$
$$\cup \{(\ell(\vec{v}), \bot) \mid \forall \ell \xrightarrow{\vec{u}} \ell' \in A \,.\, \vec{v}+\vec{u} \not\geq \vec{0}\} \cup \{(\bot, \bot)\} \,.$$

In the energy semantics, moves that would result in a negative component lead to the sink instead of being inhibited.

Example 14 (Energy semantics)**.** *Figure 11.11 illustrates the energy semantics of the vector system depicted in Figure 11.1 on p. 356. Observe that, by contrast with the natural semantics of the same system depicted in Figure 11.2, all the configurations $\ell'(0,n)$ controlled by Adam can now move to the sink.*

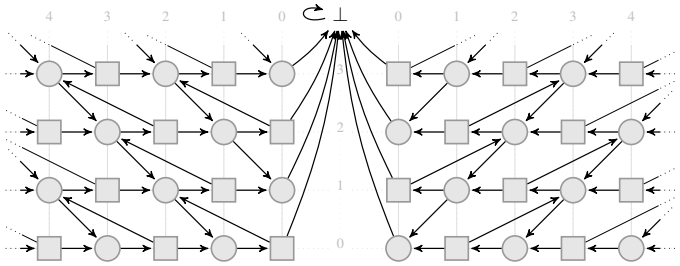


Figure 11.11: The energy semantics of the vector system of Figure 11.1: a circle (resp. a square) at position (i, j) of the grid denotes a configuration $\ell(i, j)$ (resp. $\ell'(i, j)$) controlled by Eve (resp. Adam).

Given a colouring $\mathfrak{c}: E \to C$ and an objective Ω, we call the resulting game $(\mathscr{A}_{\mathbb{E}}(\mathscr{V}), \mathfrak{c}, \Omega)$ an *energy game*. In particular, we shall speak of configuration reachability, coverability, non-termination, and parity energy games when replacing $\mathscr{A}_{\mathbb{N}}(\mathscr{V})$ by $\mathscr{A}_{\mathbb{E}}(\mathscr{V})$ in Problems 13 to 16; the existential initial credit variants are defined similarly.

Example 15 (Energy games)**.** *Consider the target configuration $\ell(2,2)$ in Figures 11.1 and 11.11. Eve's winning region in the configuration reachability energy game is $W_{\text{Eve}} = \{\ell(n+2, n+2) \mid n \in \mathbb{N}\}$, displayed on the left in Figure 11.12. In the coverability energy game, Eve's winning region is $W_{\text{Eve}} = \{\ell(m+2, n+2), \ell'(m+3, n+2) \mid m, n \in \mathbb{N}\}$ displayed on the right in Figure 11.12.*

The reader might have noticed that the natural semantics of the asymmetric system of Figure 11.7 and the energy semantics of the system of Figure 11.1 are essentially the same. This correspondence is quite general.

Lemma 134 (Energy vs. asymmetric vector games)**.** *Energy games and asymmetric vector games are* LOGSPACE*-equivalent for configuration reachability, coverability, non-termination, and parity, both with given and with existential initial credit.*

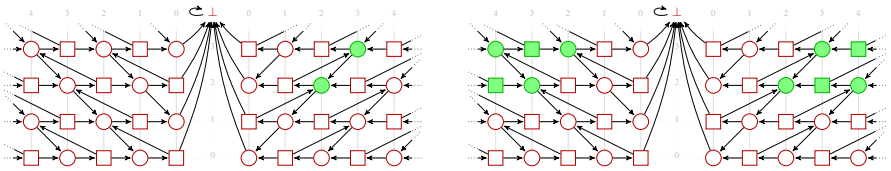


Figure 11.12: The winning regions of Eve in the configuration reachability energy game (left) and the coverability energy game (right) on the graphs of Figures 11.1 and 11.11 with target configuration $\ell(2,2)$. The winning vertices are in filled in green, while the losing ones are filled with white with a red border; the sink is always losing.

Proof. Let us first reduce asymmetric vector games to energy games. Given $\mathscr{V}$, $\mathfrak{c}$, and Ω where $\mathscr{V}$ is asymmetric and Eve loses if the play ever visits the sink $\bot$, we see that Eve wins $(\mathscr{A}_{\mathbb{N}}(\mathscr{V}),\mathfrak{c},\Omega)$ from some $v \in V$ if and only if she wins $(\mathscr{A}_{\mathbb{E}}(\mathscr{V}),\mathfrak{c},\Omega)$ from v. Of course, this might not be true if $\mathscr{V}$ is not asymmetric, as seen for instance in Examples 9 and 15.

Conversely, let us reduce energy games to asymmetric vector games. Consider $\mathscr{V} = (\mathscr{L},A,\mathscr{L}_{\text{Eve}},\mathscr{L}_{\text{Adam}},k)$, a colouring $\mathfrak{c}$ defined from a vertex colouring vcol by $\mathfrak{c}(e) \stackrel{\text{def}}{=} \text{vcol}(\text{In}(e))$, and an objective Ω, where vcol and Ω are such that Eve loses if the play ever visits the sink $\bot$ and such that, for all $\pi \in C^*$, $p \in C$, and $\pi' \in C^\omega$, $\pi p \pi' \in \Omega$ if and only if $\pi pp\pi' \in \Omega$ (we shall call Ω *stutter-invariant*, and the objectives in the statement are indeed stutter-invariant). We construct an asymmetric vector system $\mathscr{V}' \stackrel{\text{def}}{=} (\mathscr{L} \uplus \mathscr{L}_A, A', \mathscr{L}_{\text{Eve}} \uplus \mathscr{L}_A, \mathscr{L}_{\text{Adam}}, k)$ where we add the following locations controlled by Eve:

$$\mathscr{L}_A \stackrel{\text{def}}{=} \{\ell_a \mid a = (\ell \xrightarrow{\vec{u}} \ell') \in A \text{ and } \ell \in \mathscr{L}_{\text{Adam}}\} .$$

We also modify the set of actions:

$$\begin{aligned} A' \stackrel{\text{def}}{=} &\{\ell \xrightarrow{\vec{u}} \ell' \mid \ell \xrightarrow{\vec{u}} \ell' \in A \text{ and } \ell \in \mathscr{L}_{\text{Eve}}\} \\ &\cup \{\ell \xrightarrow{\vec{0}} \ell_a,\ \ell_a \xrightarrow{\vec{u}} \ell' \mid a = (\ell \xrightarrow{\vec{u}} \ell') \in A \text{ and } \ell \in \mathscr{L}_{\text{Adam}}\} . \end{aligned}$$

Figure 11.7 presents the result of this reduction on the system of Figure 11.1. We define a vertex colouring vcol$'$ of $\mathscr{A}_{\mathbb{N}}(\mathscr{V}')$ with vcol$'(v) \stackrel{\text{def}}{=} \text{vcol}(v)$ for all $v \in \mathscr{L} \times \mathbb{N}^k \uplus \{\bot\}$ and vcol$'(\ell_a(\vec{v})) \stackrel{\text{def}}{=} \text{vcol}(\ell(\vec{v}))$ if $a = (\ell \xrightarrow{\vec{u}} \ell') \in A$. Then, for all vertices $v \in V$, Eve wins from v in the energy game $(\mathscr{A}_{\mathbb{E}}(\mathscr{V}),\mathfrak{c},\Omega)$ if and only if she wins from v in the vector game $(\mathscr{A}_{\mathbb{N}}(\mathscr{V}'),\mathfrak{c}',\Omega)$. The crux of the argument is that, in a configuration $\ell(\vec{v})$ where $\ell \in \mathscr{L}_{\text{Adam}}$, if $a = (\ell \xrightarrow{\vec{u}} \ell') \in A$ is an action with $\vec{v} + \vec{u} \not\geq \vec{0}$, in the energy semantics, Adam can force the play into the sink by playing a; the same occurs in $\mathscr{V}'$ with the natural semantics, as Adam can now choose to play $\ell \xrightarrow{\vec{0}} \ell_a$ where Eve has only $\ell_a \xrightarrow{\vec{u}} \ell'$ at her disposal, which leads to the sink. □

In turn, energy games with existential initial credit are related to the multi-dimensional mean payoff games of Chapter 12.

11.4.2 Bounded Semantics

While Adam wins immediately in an energy game if a resource gets depleted, he also wins in a bounding game if a resource reaches a certain bound B. This is a *hard upper bound*, allowing to model systems where exceeding a capacity results in failure, like a dam that overflows and floods the area. We define for a bound $B \in \mathbb{N}$ the *bounded semantics* $\mathscr{A}_B(\mathscr{V}) = (V^B, E^B, V^B_{\text{Eve}}, V^B_{\text{Adam}})$ of a vector system $\mathscr{V}$ by

$$
\begin{aligned}
V^B &\stackrel{\text{def}}{=} \{\ell(\vec{v}) \mid \ell \in \mathscr{L} \text{ and } \|\vec{v}\| < B\}\ , \\
E^B &\stackrel{\text{def}}{=} \{(\ell(\vec{v}), \ell'(\vec{v}+\vec{u})) \mid \ell \xrightarrow{\vec{u}} \ell' \in A, \vec{v}+\vec{u} \geq \vec{0}, \text{ and } \|\vec{v}+\vec{u}\| < B\} \\
&\quad \cup \{(\ell(\vec{v}), \bot) \mid \forall \ell \xrightarrow{\vec{u}} \ell' \in A\,.\, \vec{v}+\vec{u} \not\geq \vec{0} \text{ or } \|\vec{v}+\vec{u}\| \geq B\} \cup \{(\bot, \bot)\}\ .
\end{aligned}
$$

As usual, $V^B_{\text{Eve}} \stackrel{\text{def}}{=} V^B \cap \mathscr{L}_{\text{Eve}} \times \mathbb{N}^k$ and $V^B_{\text{Adam}} \stackrel{\text{def}}{=} V^B \cap \mathscr{L}_{\text{Adam}} \times \mathbb{N}^k$. Any edge from the energy semantics that would bring to a configuration $\ell(\vec{v})$ with $\vec{v}(i) \geq B$ for some $1 \leq i \leq k$ leads instead to the sink. All the configurations in this arena have norm less than B, thus $|V^B| = |\mathscr{L}|B^k + 1$, and the qualitative games of Chapter 2 are decidable over this arena.

Our focus here is on non-termination games played on the bounded semantics where B is not given as part of the input, but quantified existentially. As usual, the existential initial credit variant of Problem 18 is obtained by quantifying $\vec{v}_0$ existentially in the question.

Problem 18 (bounding game with given initial credit)**.**

INPUT: *A vector system $\mathscr{V} = (\mathscr{L}, A, \mathscr{L}_{\text{Eve}}, \mathscr{L}_{\text{Adam}}, k)$, an initial location $\ell_0 \in \mathscr{L}$, and an initial credit $\vec{v}_0 \in \mathbb{N}^k$.*

OUTPUT: *Does there exist $B \in \mathbb{N}$ such that Eve has a strategy to avoid the sink $\bot$ from $\ell_0(\vec{v}_0)$ in the bounded semantics? That is, does there exist $B \in \mathbb{N}$ such that she wins the* bounding *game $(\mathscr{A}_B(\mathscr{V}), \mathtt{c}, \mathtt{Safe})$ from $\ell_0(\vec{v}_0)$, where $\mathtt{c}(e) \stackrel{def}{=}$ Lose if and only if $In(e) = \bot$?*

Lemma 135. *There is a* LOGSPACE *reduction from parity asymmetric vector games to bounding games, both with given and with existential initial credit.*

Proof. Given an asymmetric vector system $\mathscr{V} = (\mathscr{L}, A, \mathscr{L}_{\text{Eve}}, \mathscr{L}_{\text{Adam}}, k)$, a location colouring $\text{lcol}: \mathscr{L} \to \{1, \ldots, 2d\}$, and an initial location $\ell_0 \in \mathscr{L}$, we construct a vector system $\mathscr{V}'$ of dimension $k' \stackrel{\text{def}}{=} k + d$ as described in Figure 11.13, where the priorities in $\mathscr{V}$ for $p \in \{1, \ldots, d\}$ are indicated above the corresponding locations.

If Eve wins the bounding game played over $\mathscr{V}'$ from some configuration $\ell_0(\vec{v}_0)$, then she also wins the parity vector game played over $\mathscr{V}$ from the configuration $\ell_0(\vec{v}'_0)$ where $\vec{v}'_0$ is the projection of $\vec{v}_0$ to $\mathbb{N}^k$. Indeed, she can play essentially the same strategy: by Lemma 132 she can simply ignore the new decrement self loops, while the actions on the components in $\{k+1, \ldots, k+d\}$ ensure that the maximal priority visited infinitely often is even—otherwise some decrement $-\vec{e}_{k+p}$ would be played infinitely often but the increment $\vec{e}_{k+p}$ only finitely often.

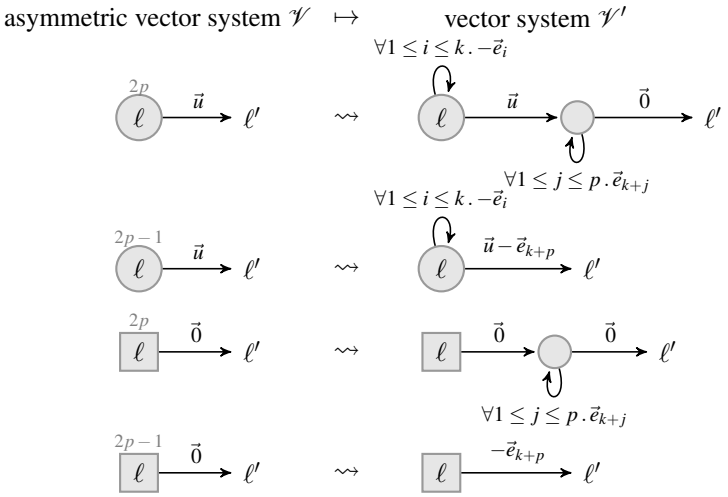


Figure 11.13: Schema of the reduction to bounding games in the proof of Lemma 135.

Conversely, consider the parity game $\mathscr{G}$ played over $\mathscr{A}_{\mathbb{N}}(\mathscr{V})$ with the colouring defined by lcol. Then the Pareto limit of the game is finite, thus there exists a natural number

$$B_0 \stackrel{\text{def}}{=} 1 + \max_{\ell_0(\vec{v}_0) \in \mathsf{Pareto}(\mathscr{G})} \|\vec{v}_0\| \tag{11.1}$$

bounding the norms of the minimal winning configurations. For a vector $\vec{v}$ in $\mathbb{N}^k$, let us write $\overline{\vec{v}}^{B_0}$ for the vector 'capped' at B: for all $1 \le i \le k$, $\overline{\vec{v}}^{B_0}(i) \stackrel{\text{def}}{=} \vec{v}(i)$ if $\vec{v}(i) < B_0$ and $\overline{\vec{v}}^{B_0} \stackrel{\text{def}}{=} B_0$ if $\vec{v}(i) \ge B_0$.

Consider now some configuration $\ell_0(\vec{v}_0) \in \mathsf{Pareto}(\mathscr{G})$. As seen in Lemma 133, since $\ell_0(\vec{v}_0) \in W_{\text{Eve}}(\mathscr{G})$, there is a finite self-covering tree witnessing the fact, and an associated winning strategy. Let $H(\ell_0(\vec{v}_0))$ denote the height of this self-covering tree and observe that all the configurations in this tree have norm bounded by $\|\vec{v}_0\| + \|A\| \cdot H(\ell_0(\vec{v}_0))$. Let us define

$$B \stackrel{\text{def}}{=} B_0 + (\|A\| + 1) \cdot \max_{\ell_0(\vec{v}_0) \in \mathsf{Pareto}(\mathscr{G})} H(\ell_0(\vec{v}_0)) \,. \tag{11.2}$$

This is a bound on the norm of the configurations appearing on the (finitely many) self-covering trees spawned by the elements of $\mathsf{Pareto}(\mathscr{G})$. Note that $B \ge B_0 + (\|A\| + 1)$ since a self-covering tree has height at least one.

Consider the non-termination game $\mathscr{G}_B \stackrel{\text{def}}{=} (\mathscr{A}_B(\mathscr{V}'), \mathfrak{c}', \texttt{Safe})$ played over the bounded semantics defined by B, where $\mathfrak{c}'(e) = \text{Lose}$ if and only if $\text{In}(e) = \bot$. Let $\vec{b} \stackrel{\text{def}}{=} \sum_{1 \le p \le d} (B-1) \cdot \vec{e}_{k+p}$.

Claim 3. *If $\ell_0(\vec{v}) \in W_{\text{Eve}}(\mathscr{G})$, then $\ell_0(\overline{\vec{v}}^{B_0} + \vec{b}) \in W_{\text{Eve}}(\mathscr{G}_B)$.*

Indeed, by definition of the Pareto limit $\mathsf{Pareto}(\mathscr{G})$, if $\ell_0(\vec{v}) \in W_{\text{Eve}}(\mathscr{G})$, then there exists $\vec{v}_0 \le \vec{v}$ such that $\ell_0(\vec{v}_0) \in \mathsf{Pareto}(\mathscr{G})$. By definition of the bound B_0, $\|\vec{v}_0\| < B_0$,

thus $\vec{v}_0 \leq \vec{v}^{B_0}$. Consider the self-covering tree of height $H(\ell_0(\vec{v}_0))$ associated to $\ell_0(\vec{v}_0)$, and the strategy σ' defined by the memory structure from the proof of Lemma 133. This is a winning strategy for Eve in $\mathscr{G}$ starting from $\ell_0(\vec{v}_0)$, and by Lemma 132, it is also winning from $\ell_0(\vec{v}^{B_0})$.

Here is how Eve wins $\mathscr{G}_B$ from $\ell_0(\vec{v}^{B_0} + \vec{b})$. She essentially follows the strategy σ', with two modifications. First, whenever σ' goes to a return node $\ell(\vec{v})$ instead of a leaf $\ell(\vec{v}')$—thus $\vec{v} \leq \vec{v}'$—, the next time Eve has the control, she uses the self loops to decrement the current configuration by $\vec{v}' - \vec{v}$. This ensures that any play consistent with the modified strategy remains between zero and $B-1$ on the components in $\{1,\ldots,k\}$. (Note that if she never regains the control, the current vector never changes any more since $\mathscr{V}$ is asymmetric.)

Second, whenever a play in $\mathscr{G}$ visits a location with even parity $2p$ for some p in $\{1,\ldots,d\}$, Eve has the opportunity to increase the coordinates in $\{k+1,\ldots,k+p\}$ in $\mathscr{G}_B$. She does so and increments until all these components reach $B-1$. This ensures that any play consistent with the modified strategy remains between zero and $B-1$ on the components in $\{k+1,\ldots,k+p\}$. Indeed, σ' guarantees that the longest sequence of moves before a play visits a location with maximal even priority is bounded by $H(\ell_0(\vec{v}_0))$, thus the decrements $-\vec{e}_{k+p}$ introduced in $\mathscr{G}_B$ by the locations from $\mathscr{G}$ with odd parity $2p-1$ will never force the play to go negative. □

The bound B defined in Equation (11.2) in the previous proof is not constructive, and possibly much larger than really required. Nevertheless, one can sometimes show that an explicit B suffices in a bounding game. A simple example is provided by the coverability asymmetric vector games with existential initial credit arising from Remark 22, i.e., where the objective is to reach some location ℓ_f. Indeed, it is rather straightforward that there exists a suitable initial credit such that Eve wins the game if and only if she wins the finite reachability game played over the underlying directed graph over $\mathscr{L}$ where we ignore the counters. Thus, for an initial location ℓ_0, $B_0 = |\mathscr{L}| \cdot \|A\| + 1$ bounds the norm of the necessary initial credit, while a simple path may visit at most $|\mathscr{L}|$ locations, thus $B = B_0 + |\mathscr{L}| \cdot \|A\|$ suffices for Eve to win the constructed bounding game.

In the general case of bounding games with existential initial credit, an explicit bound can be established. The proof goes along very different lines and is too involved to fit in this chapter, but we refer the reader to [JLS15, CJLS17] for details.

Theorem 119 (Bounds on bounding)**.** *If Eve wins a bounding game with existential initial credit defined by a vector system $\mathscr{V} = (\mathscr{L}, A, \mathscr{L}_{\text{Eve}}, \mathscr{L}_{\text{Adam}}, k)$, then an initial credit $\vec{v}_0$ with $\|\vec{v}_0\| = (4|\mathscr{L}| \cdot \|A\|)^{2(k+2)^3}$ and a bound $B = 2(4|\mathscr{L}| \cdot \|A\|)^{2(k+2)^3} + 1$ suffice for this.*

Theorem 119 also yields a way of handling bounding games with given initial credit.

11.5 The complexity of asymmetric monotone games

Unlike general vector games and configuration reachability asymmetric ones, coverability, non-termination, and parity asymmetric vector games are decidable. We survey in this section the best known complexity bounds for every case; see Table 11.1 at the end of the chapter for a summary.

11.5.1 Upper Bounds

We begin with complexity upper bounds. The main results are that parity games with existential initial credit can be solved in coNP, but are in 2EXP with given initial credit. In both cases however, the complexity is pseudo-polynomial if both the dimension k and the number of priorities d are fixed, which is rather good news: one can hope that, in practice, both the number of different resources (encoded by the counters) and the complexity of the functional specification (encoded by the parity condition) are tiny compared to the size of the system.

Existential Initial Credit

Counterless Strategies Consider a strategy τ of Adam in a vector game. In all the games we consider, uniform positional strategies suffice over the infinite arena $\mathscr{A}_{\mathbb{N}}(\mathscr{V}) = (V, E, V_{\text{Eve}}, V_{\text{Adam}})$: τ maps vertices in V to edges in E. We call τ *counterless* if, for all locations $\ell \in \mathscr{L}_{\text{Adam}}$ and all vectors $\vec{v}, \vec{v}' \in \mathbb{N}^k$, $\tau(\ell(\vec{v})) = \tau(\ell(\vec{v}'))$. A counterless strategy thus only considers the current location of the play.

Lemma 136 (Counterless strategies)**.** *Let $\mathscr{V} = (\mathscr{L}, A, \mathscr{L}_{\text{Eve}}, \mathscr{L}_{\text{Adam}}, k)$ be an asymmetric vector system, $\ell_0 \in \mathscr{L}$ be a location, and* lcol$: \mathscr{L} \to \{1, \dots, d\}$ *be a location colouring. If Adam wins from $\ell_0(\vec{v})$ for every initial credit $\vec{v}$ in the parity game played over $\mathscr{V}$ with* lcol*, then he has a single counterless strategy such that he wins from $\ell_0(\vec{v})$ for every initial credit $\vec{v}$.*

Proof. Let $A_{\text{Adam}} \stackrel{\text{def}}{=} \{(\ell \xrightarrow{\vec{u}} \ell') \in A \mid \ell \in \mathscr{L}_{\text{Adam}}\}$ be the set of actions controlled by Adam. We assume without loss of generality that every location $\ell \in \mathscr{L}_{\text{Adam}}$ has either one or two outgoing actions, thus $|\mathscr{L}_{\text{Adam}}| \leq |A_{\text{Adam}}| \leq 2|\mathscr{L}_{\text{Adam}}|$. We proceed by induction over $|A_{\text{Adam}}|$. For the base case, if $|A_{\text{Adam}}| = |\mathscr{L}_{\text{Adam}}|$ then every location controlled by Adam has a single outgoing action, thus any strategy for Adam is trivially counterless.

For the induction step, consider some location $\hat{\ell} \in \mathscr{L}_{\text{Adam}}$ with two outgoing actions $a_l \stackrel{\text{def}}{=} \hat{\ell} \xrightarrow{\vec{0}} \ell_l$ and $a_r \stackrel{\text{def}}{=} \hat{\ell} \xrightarrow{\vec{0}} \ell_r$. Let $\mathscr{V}_l$ and $\mathscr{V}_r$ be the vector systems obtained from $\mathscr{V}$ by removing respectively a_r and a_l from A, i.e., by using $A_l \stackrel{\text{def}}{=} A \setminus \{a_r\}$ and $A_r \stackrel{\text{def}}{=} A \setminus \{a_l\}$. If Adam wins the parity game from $\ell(\vec{v})$ for every initial credit $\vec{v}$ in either $\mathscr{V}_l$ or $\mathscr{V}_r$, then by induction hypothesis he has a counterless winning strategy winning from $\ell(\vec{v})$ for every initial credit $\vec{v}$, and the same strategy is winning in $\mathscr{V}$ from $\ell(\vec{v})$ for every initial credit $\vec{v}$.

In order to conclude the proof, we show that, if Adam loses in $\mathscr{V}_l$ from $\ell_0(\vec{v}_l)$ for some $\vec{v}_l \in \mathbb{N}^k$ and in $\mathscr{V}_r$ from $\ell_0(\vec{v}_r)$ for some $\vec{v}_r \in \mathbb{N}^k$, then there exists $\vec{v}_0 \in \mathbb{N}^k$ such that Eve wins from $\ell_0(\vec{v}_0)$ in $\mathscr{V}$. Let σ_l and σ_r denote Eve's winning strategies in the two

games. By a slight abuse of notations (justified by the fact that we are only interested in a few initial vertices), we see plays as sequences of actions and strategies as maps $A^* \to A$.Consider the set of plays consistent with σ_r starting from $\ell_0(\vec{v}_r)$. If none of those plays visits $\hat{\ell}$, then Eve wins in $\mathscr{V}$ from $\ell_0(\vec{v}_r)$ and we conclude. Otherwise, there is some finite prefix $\hat{\pi}$ of a play that visits $\hat{\ell}(\hat{\vec{v}})$ for some vector $\hat{\vec{v}} = \vec{v}_r + w(\hat{\pi})$. We let $\vec{v}_0 \stackrel{\text{def}}{=} \vec{v}_l + \hat{\vec{v}}$ and show that Eve wins from $\ell_0(\vec{v}_0)$.

We define now a strategy σ for Eve over $\mathscr{V}$ that switches between applying σ_l and σ_r each time a_r is used and switches back each time a_l is used. More precisely, given a finite or infinite sequence π of actions, we decompose π as $\pi_1 a_1 \pi_2 a_2 \pi_3 \cdots$ where each segment $\pi_j \in (A \setminus \{a_l, a_r\})^*$ does not use either a_l nor a_r and each $a_j \in \{a_l, a_r\}$. The associated *mode* $m(j) \in \{l, r\}$ of a segment π_j is $m(1) \stackrel{\text{def}}{=} l$ for the initial segment and otherwise $m(j) \stackrel{\text{def}}{=} l$ if $e_{j-1} = a_l$ and $m(j) \stackrel{\text{def}}{=} r$ otherwise. The l-subsequence associated with π is the sequence of segments $\pi(l) \stackrel{\text{def}}{=} \pi_{l_1} a_{l_2-1} \pi_{l_2} a_{l_3-1} \pi_{l_3} \cdots$ with mode $m(l_i) = l$, while the r-subsequence is the sequence $\pi(r) \stackrel{\text{def}}{=} \hat{\pi} a_{r_1-1} \pi_{r_1} a_{r_2-1} \pi_{r_2} \cdots$ with mode $m(r_i) = r$ prefixed by $\hat{\pi}$. Then we let $\sigma(\pi) \stackrel{\text{def}}{=} \sigma_m(\pi(m))$ where $m \in \{l, r\}$ is the mode of the last segment of π.

Consider an infinite play π consistent with σ starting from $\ell_0(\vec{v}_0)$. Since $\vec{v}_0 \geq \vec{v}_l$ and $\vec{v}_0 \geq \vec{v}_r + w(\hat{\pi})$, $\pi(l)$ and $\pi(r)$ starting from $\ell_0(\vec{v}_0)$ are consistent with simulating—in the sense of Lemma 132—σ_l from $\ell_0(\vec{v}_l)$ and σ_r from $\ell_0(\vec{v}_r)$. Let π' be a finite prefix of π. Then $w(\pi') = w(\pi'(l)) + w(\pi'(r))$ where $\pi'(l)$ is a prefix of $\pi(l)$ and $\pi'(r)$ of $\pi(r)$, thus $w(\pi'(l)) \leq \vec{v}_l$ and $w(\pi'(r)) \leq \vec{v}_r + w(\hat{\pi})$, thus $w(\pi') \leq \vec{v}_0$: the play π avoids the sink. Furthermore, the maximal priority seen infinitely often along $\pi(l)$ and $\pi(r)$ is even (note that one of $\pi(l)$ and $\pi(r)$ might not be infinite), thus the maximal priority seen infinitely often along π is also even. This shows that σ is winning for Eve from $\ell_0(\vec{v}_0)$. □

We are going to exploit Lemma 136 in Theorem 122 in order to prove a coNP upper bound for asymmetric games with existential initial credit: it suffices in order to decide those games to guess a counterless winning strategy τ for Adam and check that it is indeed winning by checking that Eve loses the one-player game arising from τ. This last step requires an algorithmic result of independent interest.

One-player Case Let $\mathscr{V} = (\mathscr{L}, A, k)$ be a vector addition system with states, lcol: $\mathscr{L} \to \{1, \ldots, d\}$ a location colouring, and $\ell_0 \in \mathscr{L}$ an initial location. Then Eve wins the parity one-player game from $\ell_0(\vec{v}_0)$ for some initial credit $\vec{v}_0$ if and only if there exists some location such that

- ℓ is reachable from ℓ_0 in the directed graph underlying $\mathscr{V}$ and
- there is a cycle $\pi \in A^*$ from ℓ to itself such that $w(\pi) \geq 0$ and the maximal priority occurring along π is even.

Indeed, assume we can find such a location ℓ. Let $\hat{\pi} \in A^*$ be a path from ℓ_0 to ℓ and $\vec{v}_0(i) \stackrel{\text{def}}{=} \max\{\|w(\pi')\| \mid \pi' \text{ is a prefix of } \hat{\pi}\pi\}$ for all $1 \leq i \leq k$. Then $\ell_0(\vec{v}_0)$ can reach $\ell(\vec{v}_0 + w(\hat{\pi}))$ in the natural semantics of $\mathscr{V}$ by following $\hat{\pi}$, and then $\ell(\vec{v}_0 + \vec{W}(\hat{\pi}) + nw(\pi)) \geq \ell(\vec{v}_0 + w(\hat{\pi}))$ after n repetitions of the cycle π. The infinite play arising from this strategy has an even maximal priority.

Conversely, if Eve wins, then there is a winning play $\pi \in A^\omega$ from $\ell_0(\vec{v}_0)$ for some $\vec{v}_0$. Recall that $(V, \leq)$ is a wqo, and we argue as in Lemma 133 that there is indeed such a location ℓ.

Therefore, solving one-player parity vector games boils down to determining the existence of a cycle with non-negative effect and even maximal priority. We shall use linear programming techniques in order to check the existence of such a cycle in polynomial time [KS88].

Let us start with a relaxed problem: we call a *multi-cycle* a non-empty finite set of cycles Π and let $w(\Pi) \stackrel{\text{def}}{=} \sum_{\pi \in \Pi} w(\pi)$ be its weight; we write $t \in \Pi$ if $t \in \pi$ for some $\pi \in \Pi$. Let $M \in 2^A$ be a set of 'mandatory' subsets of actions and $F \subseteq A$ a set of 'forbidden' actions. Then we say that Π is *non-negative* if $w(\Pi) \geq \vec{0}$, and that it is *suitable* for (M, F) if for all $A' \in M$ there exists $t \in A'$ such that $t \in \Pi$, and if for all $t \in F$, $t \notin \Pi$. We use the same terminology for a single cycle π.

Lemma 137 (Linear programs for suitable non-negative multi-cycles)**.** *Let $\mathscr{V}$ be a vector addition system with states, $M \in 2^A$, and $F \subseteq A$. We can check in polynomial time whether $\mathscr{V}$ contains a non-negative multi-cycle Π suitable for (M, F).*

Proof. We reduce the problem to solving a linear program. For a location ℓ, let $\text{in}(\ell) \stackrel{\text{def}}{=} \{(\ell' \xrightarrow{\vec{u}} \ell) \in A \mid \ell' \in \mathscr{L}\}$ and $\text{out}(\ell) \stackrel{\text{def}}{=} \{(\ell \xrightarrow{\vec{u}} \ell') \in A \mid \ell' \in \mathscr{L}\}$ be its sets of incoming and outgoing actions. The linear program has a variable x_a for each action $a \in A$, which represents the number of times the action a occurs in the multi-cycle. It consists of the following constraints:

$$\forall \ell \in \mathscr{L}, \qquad \sum_{a \in \text{in}(\ell)} x_a = \sum_{a \in \text{out}(\ell)} x_a \,, \qquad \text{(multi-cycle)}$$

$$\forall a \in A, \qquad x_a \geq 0 \,, \qquad \text{(non-negative uses)}$$

$$\forall i \in \{1, \ldots, k\}, \qquad \sum_{a \in A} x_a \cdot w(t)(i) \geq 0 \,, \qquad \text{(non-negative weight)}$$

$$\sum_{a \in A} x_a \geq 0 \qquad \text{(non empty)}$$

$$\forall A' \in M, \qquad \sum_{a \in A'} x_a \geq 0 \,, \qquad \text{(every subset in } M \text{ is used)}$$

$$\forall a \in F, \qquad x_a = 0 \,. \qquad \text{(no forbidden actions)}$$

As solving a linear program is in polynomial time Theorem 2, the result follows. □

Of course, what we are aiming for is finding a non-negative *cycle* suitable for (M, F) rather than a multi-cycle. Let us define for this the relation $\ell \sim \ell'$ over $\mathscr{L}$ if $\ell = \ell'$ or if there exists a non-negative multi-cycle Π suitable for (M, F) such that ℓ and ℓ' belong to some cycle $\pi \in \Pi$.

Fact 21. *The relation $\sim$ is an equivalence relation.*

Proof. Symmetry and reflexivity are trivial, and if $\ell \sim \ell'$ and $\ell' \sim \ell''$ because ℓ and ℓ' appear in some cycle $\pi \in \Pi$ and ℓ' and ℓ'' in some cycle $\pi' \in \Pi'$ for two non-negative

multi-cycles Π and Π' suitable for (M,F), then up to a circular shift π and π' can be assumed to start and end with ℓ', and then $(\Pi \setminus \{\pi\}) \cup (\Pi' \setminus \{\pi'\}) \cup \{\pi\pi'\}$ is also a non-negative multi-cycle suitable for (M,F). □

Thus $\sim$ defines a partition $\mathscr{L}/\!\sim$ of $\mathscr{L}$. In order to find a non-negative cycle π suitable for (M,F), we are going to compute the partition $\mathscr{L}/\!\sim$ of $\mathscr{L}$ according to $\sim$. If we obtain a partition with a single equivalence class, we are done: there exists such a cycle. Otherwise, such a cycle if it exists must be included in one of the subsystems $(P, A \cap (P \times \mathbb{Z}^k \times P), k)$ induced by the equivalence classes $P \in \mathscr{L}/\!\sim$. This yields Algorithm 11.2, which assumes that we know how to compute the partition $\mathscr{L}/\!\sim$. Note that the depth of the recursion in Algorithm 11.2 is bounded by $|\mathscr{L}|$ and that recursive calls operate over disjoint subsets of $\mathscr{L}$, thus assuming that we can compute the partition in polynomial time, then Algorithm 11.2 also works in polynomial time.

Algorithm 11.2: cycle$(\mathscr{V},M,F)$

```
Data: A vector addition system with states V = (L, A, k), M ∈ 2^A, F ⊆ A
if |L| = 1 then
    if V has a non-negative multi-cycle suitable for (M, F) then
        return true
L/∼ ← partition(V, M, F) ;
if |L/∼| = 1 then
    return true
foreach P ∈ L/∼ do
    if cycle((P, A ∩ (P × Z^k × P), k), M, F) then
        return true
return false
```

It remains to see how to compute the partition $\mathscr{L}/\!\sim$. Consider for this the set of actions $A' \stackrel{\text{def}}{=} \{a \mid \exists \Pi$ a non-negative multi-cycle suitable for (M,F) with $a \in \Pi\}$ and $\mathscr{V}' = (\mathscr{L}', A', k)$ the subsystem induced by A'.

Claim 4. *There exists a path from ℓ to ℓ' in $\mathscr{V}'$ if and only if $\ell \sim \ell'$.*

Proof. If $\ell \sim \ell'$, then either $\ell = \ell'$ and there is an empty path, or there exist Π and $\pi \in \Pi$ such that ℓ and ℓ' belong to π and Π is a non-negative multi-cycle suitable for (M,F), thus every action of π is in A' and there is a path in $\mathscr{V}'$.

Conversely, if there is a path $\pi \in A'^*$ from ℓ to ℓ', then $\ell \sim \ell'$ by induction on π. Indeed, if $|\pi| = 0$ then $\ell = \ell'$. For the induction step, $\pi = \pi' a$ with $\pi' \in A'^*$ a path from ℓ to ℓ'' and $a = (\ell'' \xrightarrow{\vec{u}} \ell') \in A'$ for some $\vec{u}$. By induction hypothesis, $\ell \sim \ell''$ and since $a \in A'$, $\ell'' \sim \ell'$, thus $\ell \sim \ell'$ by transitivity shown in Fact 21. □

By Claim 4, the equivalence classes of $\sim$ are the strongly connected components of $\mathscr{V}'$. This yields the following polynomial time algorithm for computing $\mathscr{L}/\!\sim$.

Together, Lemma 137 and Algorithms 11.2 and 11.3 yield the following.

Algorithm 11.3: partition$(\mathscr{V}, M, F)$

Data: A vector addition system with states $\mathscr{V} = (\mathscr{L}, A, k)$, $M \in 2^A$, $F \subseteq A$
$A' \leftarrow \emptyset$;
foreach $a \in A$ **do**
  if *$\mathscr{V}$ has a non-negative multi-cycle suitable for* $(M \cup \{\{a\}\}, F)$ **then**
    $A' \leftarrow A' \cup \{a\}$
$\mathscr{V}' \leftarrow$ subsystem induced by A' ;
return SCC$(\mathscr{V}')$

Lemma 138 (Polynomial-time detection of suitable non-negative cycles)**.** *Let $\mathscr{V}$ be a vector addition system with states, $M \in 2^A$, and $F \subseteq A$. We can check in polynomial time whether $\mathscr{V}$ contains a non-negative cycle π suitable for (M, F).*

Finally, we obtain the desired polynomial time upper bound for parity in vector addition systems with states.

Theorem 120 (Existential one-player parity vector games are in P)**.** *Whether Eve wins a one-player parity vector game with existential initial credit is in* P.

Proof. Let $\mathscr{V} = (\mathscr{L}, A, k)$ be a vector addition system with states, lcol: $\mathscr{L} \to \{1, \ldots, d\}$ a location colouring, and $\ell_0 \in \mathscr{L}$ an initial location. We start by trimming $\mathscr{V}$ to only keep the locations reachable from ℓ_0 in the underlying directed graph. Then, for every even priority $p \in \{1, \ldots, d\}$, we use Lemma 138 to check for the existence of a non-negative cycle with maximal priority p: it suffices for this to set $M \stackrel{\text{def}}{=} \{\text{lcol}^{-1}(p)\}$ and $F \stackrel{\text{def}}{=} \text{lcol}^{-1}(\{p+1, \ldots, d\})$. □

Upper Bounds We are now equipped to prove our upper bounds. We begin with a nearly trivial case. In a coverability asymmetric vector game with existential initial credit, the counters play no role at all: Eve has a winning strategy for some initial credit in the vector game if and only if she has one to reach the target location ℓ_f in the finite game played over $\mathscr{L}$ and edges (ℓ, ℓ') whenever $\ell \xrightarrow{\vec{u}} \ell' \in A$ for some $\vec{u}$. This entails that coverability asymmetric vector games are quite easy to solve.

Theorem 121 (Existential coverability asymmetric vector games are in P)**.** *Coverability asymmetric vector games with existential initial credit are* P*-complete.*

Regarding non-termination and parity, we exploit Lemma 136 and thm. 120.

Theorem 122 (Existential parity asymmetric vector games are in coNP)**.** *Non-termination and parity asymmetric vector games with existential initial credit are in* coNP.

Proof. By Remark 21, it suffices to prove the statement for parity games. By Lemma 136, if Adam wins the game, we can guess a counterless winning strategy τ telling which action to choose for every location. This strategy yields a one-player game, and by Theorem 120 we can check in polynomial time that τ was indeed winning for Adam. □

Finally, in fixed dimension and with a fixed number of priorities, we can simply apply the results of Section 11.4.2.

Corollary 21 (Existential fixed-dimensional parity asymmetric vector games are pseudo-polynomial)**.** *Parity asymmetric vector games with existential initial credit are in pseudo-polynomial time if the dimension and the number of priorities are fixed.*

Proof. Consider an asymmetric vector system $\mathscr{V} = (\mathscr{L}, A, \mathscr{L}_{\text{Eve}}, \mathscr{L}_{\text{Adam}}, k)$ and a location colouring lcol: $\mathscr{L} \to \{1, \dots, 2d\}$. By Lemma 135, the parity vector game with existential initial credit over $\mathscr{V}$ problem reduces to a bounding game with existential initial credit over a vector system $\mathscr{V}' = (\mathscr{L}', A', \mathscr{L}'_{\text{Eve}}, \mathscr{L}'_{\text{Adam}}, k+d)$ where $\mathscr{L}' \in O(|\mathscr{L}|)$ and $\|A'\| = \|A\|$. By Theorem 119, it suffices to consider the case of a non-termination game with existential initial credit played over the bounded semantics $\mathscr{A}_B(\mathscr{V}')$ where B is in $(|\mathscr{L}'| \cdot \|A'\|)^{O(k+d)^3}$. Such a game can be solved in linear time in the size of the bounded arena using attractor techniques, thus in $O(|\mathscr{L}| \cdot B)^{k+d}$, which is in $(|\mathscr{L}| \cdot \|A\|)^{O(k+d)^4}$ in terms of the original instance. □

Given Initial Credit

Theorem 123 (Upper bounds for asymmetric vector games)**.** *Coverability, non-termination, and parity asymmetric vector games with given initial credit are in* 2EXP*. If the dimension is fixed, they are in* EXP*, and if the number of priorities is also fixed, they are in pseudo-polynomial time.*

11.5.2 Lower Bounds

Let us turn our attention to complexity lower bounds for monotonic asymmetric vector games. It turns out that most of the upper bounds shown in Section 11.5.1 are tight.

Existential Initial Credit

In the existential initial credit variant of our games, we have the following lower bound matching Theorem 122, already with a unary encoding.

Theorem 124 (Existential non-termination asymmetric vector games are coNP-hard)**.** *Non-termination, and parity asymmetric vector games with existential initial credit are* coNP*-hard.*

Proof. By Remark 21, it suffices to show hardness for non-termination games. We reduce from the 3SAT problem: given a formula $\varphi = \bigwedge_{1 \le i \le m} C_i$ where each clause C_i is a disjunction of the form $\ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3}$ of literals taken from $X = \{x_1, \neg x_1, x_2, \neg x_2, \dots, x_k, \neg x_k\}$, we construct an asymmetric vector system $\mathscr{V}$ where Eve wins the non-termination game with existential initial credit if and only if φ is not satisfiable; since the game is determined, we actually show that Adam wins the game if and only if φ is satisfiable.

Our vector system has dimension $2k$, and for a literal $\ell \in X$, we define the vector

$$\vec{u}_\ell \stackrel{\text{def}}{=} \begin{cases} \vec{e}_{2n-1} - \vec{e}_{2n} & \text{if } \ell = x_n \,, \\ \vec{e}_{2n} - \vec{e}_{2n-1} & \text{if } \ell = \neg x_n \,. \end{cases}$$

We define $\mathscr{V} \stackrel{\text{def}}{=} (\mathscr{L}, A, \mathscr{L}_{\text{Eve}}, \mathscr{L}_{\text{Adam}}, 2k)$ where

$$\mathscr{L}_{\text{Eve}} \stackrel{\text{def}}{=} \{\varphi\} \cup \{\ell_{i,j} \mid 1 \leq i \leq m, 1 \leq j \leq 3\} ,$$
$$\mathscr{L}_{\text{Adam}} \stackrel{\text{def}}{=} \{C_i \mid 1 \leq i \leq m\} ,$$
$$A \stackrel{\text{def}}{=} \{\varphi \xrightarrow{\vec{0}} C_i \mid 1 \leq i \leq m\} \cup \{C_i \xrightarrow{\vec{0}} \ell_{i,j},\ \ell_{i,j} \xrightarrow{\vec{u}_{\ell_{i,j}}} \varphi \mid 1 \leq i \leq m, 1 \leq j \leq 3\} .$$

We use φ as our initial location. Let us call a map $v : X \to \{0,1\}$ a *literal assignment*; we call it *conflicting* if there exists $1 \leq n \leq k$ such that $v(x_n) = v(\neg x_n)$.

Assume that φ is satisfiable. Then there exists a non-conflicting literal assignment v that satisfies all the clauses: for each $1 \leq i \leq m$, there exists $1 \leq j \leq 3$ such that $v(\ell_{i,j}) = 1$; this yields a counterless strategy for Adam, which selects $(C_i, \ell_{i,j})$ for each $1 \leq i \leq m$. Consider any infinite play consistent with this strategy. This play only visits literals ℓ where $v(\ell) = 1$. There exists a literal $\ell \in X$ that is visited infinitely often along the play, say $\ell = x_n$. Because v is non-conflicting, $v(\neg x_n) = 0$, thus the location $\neg x_n$ is never visited. Thus the play uses the action $\ell \xrightarrow{\vec{e}_{2n-1} - \vec{e}_{2n}} \varphi$ infinitely often, and never uses any action with a positive effect on component $2n$. Hence the play is losing from any initial credit.

Conversely, assume that φ is not satisfiable. By contradiction, assume that Adam wins the game for all initial credits. By Lemma 136, he has a counterless winning strategy τ that selects a literal in every clause. Consider a literal assignment that maps each one of the selected literals to 1 and the remaining ones in a non-conflicting manner. By definition, this literal assignment satisfies all the clauses, but because φ is not satisfiable, it is conflicting: necessarily, there exist $1 \leq n \leq k$ and $1 \leq i, i' \leq m$, such that τ selects x_n in C_i and $\neg x_n$ in $C_{i'}$. But this yields a winning strategy for Eve, which alternates in the initial location φ between C_i and $C_{i'}$, and for which an initial credit $\vec{e}_{2n-1} + \vec{e}_{2n}$ suffices: a contradiction. □

Note that Theorem 124 does not apply to fixed dimensions $k \geq 2$. We know by Corollary 21 that those games can be solved in pseudo-polynomial time if the number of priorities is fixed, and by Theorem 122 that they are in coNP.

Given Initial Credit

With given initial credit, we have a lower bound matching the 2EXP upper bound of Theorem 123, already with a unary encoding. The proof itself is an adaptation of the proof by Lipton [Lip76] of EXPSPACE-hardness of coverability in the one-player case.

Theorem 125 (Coverability and non-termination asymmetric vector games are 2EXP-hard)**.** *Coverability, non-termination, and parity asymmetric vector games with given initial credit are* 2EXP*-hard.*

Proof. We reduce from the halting problem of an *alternating Minsky machine* $\mathscr{M} = (\mathscr{L}, A, \mathscr{L}_{\text{Eve}}, \mathscr{L}_{\text{Adam}}, k)$ with counters bounded by $B \stackrel{\text{def}}{=} 2^{2^n}$ for $n \stackrel{\text{def}}{=} |\mathscr{M}|$. Such a machine is similar to an asymmetric vector system with increments $\ell \xrightarrow{\vec{e}_i} \ell'$, decrements $\ell \xrightarrow{-\vec{e}_i} \ell'$, and zero test actions $\ell \xrightarrow{i=0} \ell'$, all restricted to locations $\ell \in \mathscr{L}_{\text{Eve}}$; the only actions available to Adam are actions $\ell \xrightarrow{\vec{0}} \ell'$. The set of locations contains a distinguished 'halt' location $\ell_{\texttt{halt}} \in \mathscr{L}$ with no outgoing action. The machine comes with

the promise that, along any play, the norm of all the visited configurations $\ell(\vec{v})$ satisfies $\|\vec{v}\| < B$. The halting problem asks, given an initial location $\ell_0 \in \mathscr{L}$, whether Eve has a winning strategy to visit $\ell_{\mathtt{halt}}(\vec{v})$ for some $\vec{v} \in \mathbb{N}^k$ from the initial configuration $\ell_0(\vec{0})$. This problem is 2EXP-complete if $k \geq 3$ by standard arguments [FMR68].

Let us start by a quick refresher on Lipton's construction [Lip76]; see also [Esp98] for a nice exposition. At the heart of the construction lies a collection of one-player gadgets implementing *level j meta-increments* $\ell \xrightarrow{2^{2^j} \cdot \vec{c}} \ell'$ and *level j meta-decrements* $\ell \xrightarrow{-2^{2^j} \cdot \vec{c}} \ell'$ for some unit vector $\vec{c}$ using $O(j)$ auxiliary counters and $\mathsf{poly}(j)$ actions, with precondition that the auxiliary counters are initially empty in ℓ and post relation that they are empty again in ℓ'. The construction is by induction over j; let us first see a naive implementation for meta-increments. For the base case $j = 0$, this is just a standard action $\ell \xrightarrow{2\vec{c}} \ell'$. For the induction step $j+1$, we use the gadget of Figure 11.14 below, where $\vec{x}_j, \bar{\vec{x}}_j, \vec{z}_j, \bar{\vec{z}}_j$ are distinct fresh unit vectors: the gadget performs two nested loops, each of 2^{2^j} iterations, thus iterates the unit increment of $\vec{c}$ a total of $\left(2^{2^j}\right)^2 = 2^{2^{j+1}}$ times. A meta-decrement is obtained similarly.

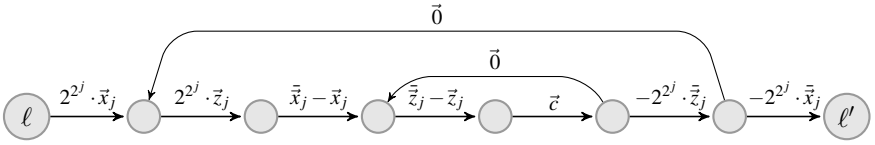


Figure 11.14: A naive implementation of the meta-increment $\ell \xrightarrow{2^{2^{j+1}} \cdot \vec{c}} \ell'$.

Note that this level $(j+1)$ gadget contains two copies of the level j meta-increment and two of the level j meta-decrement, hence this naive implementation has size $\exp(j)$. In order to obtain a polynomial size, we would like to use a single *shared* level j gadget for each j, instead of hard-wiring multiple copies. The idea is to use a 'dispatch mechanism,' using extra counters, to encode the choice of unit vector $\vec{c}$ and of return location ℓ'. Let us see how to do this in the case of the return location ℓ'; the mechanism for the vector $\vec{c}$ is similar. We enumerate the (finitely many) possible return locations $\ell_0, \ldots, \ell_{m-1}$ of the gadget implementing $\ell \xrightarrow{2^{2^{j+1}} \cdot \vec{c}} \ell'$. We use two auxiliary counters with unit vectors $\vec{r}_j$ and $\bar{\vec{r}}_j$ to encode the return location. Assume ℓ' is the ith possible return location, i.e., $\ell' = \ell_i$ in our enumeration: before entering the shared gadget implementation, we initialise $\vec{r}_j$ and $\bar{\vec{r}}_j$ by performing the action $\ell \xrightarrow{i \cdot \vec{r}_j + (m-i) \cdot \bar{\vec{r}}_j} \cdots$. Then, where we would simply go to ℓ' in Figure 11.14 at the end of the gadget, the shared gadget has a final action $\cdots \xrightarrow{\vec{0}} \ell_{\mathrm{return}_j}$ leading to a dispatch location for returns: for all $0 \leq i < m$, we have an action $\ell_{\mathrm{return}_j} \xrightarrow{-i \cdot \vec{r}_j - (m-i) \cdot \bar{\vec{r}}_j} \ell_i$ that leads to the desired return location.

Let us return to the proof. Consider an instance of the halting problem. We first exhibit a reduction to coverability; by Remark 22, this will also entail the 2EXP-hardness of parity asymmetric vector games. We build an asymmetric vector system $\mathscr{V} = (\mathscr{L}', A', \mathscr{L}'_{\mathrm{Eve}}, \mathscr{L}_{\mathrm{Adam}}, k')$ with $k' = 2k + O(n)$. Each of the counters c_i of $\mathscr{M}$ is paired with a *complementary* counter $\bar{\mathsf{c}}_i$ such that their sum is B throughout the sim-

ulation of $\mathscr{M}$. We denote by $\vec{c}_i$ and $\bar{\vec{c}}_i$ the corresponding unit vectors for $1 \leq i \leq k$. The vector system $\mathscr{V}$ starts by initialising the counters $\bar{\mathsf{c}}_i$ to B by a sequence of meta-increments $\ell'_{i-1} \xrightarrow{2^{2^n} \cdot \bar{\vec{c}}_i} \ell'_i$ for $1 \leq i \leq k$, before starting the simulation by an action $\ell'_k \xrightarrow{\vec{0}} \ell_0$. The simulation of $\mathscr{M}$ uses the actions depicted in Figure 11.15. Those maintain the invariant on the complement counters. Regarding zero tests, Eve yields the control to Adam, who has a choice between performing a meta-decrement that will fail if $\bar{\mathsf{c}}_i < 2^{2^n}$, which by the invariant is if and only if $\mathsf{c}_i > 0$, or going to ℓ'.

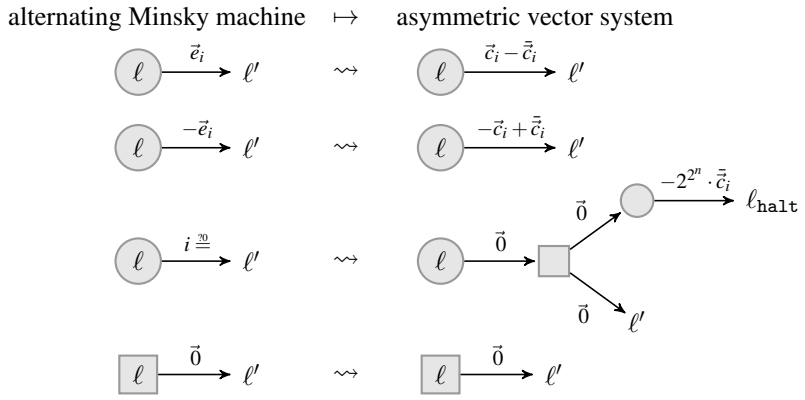


Figure 11.15: Schema of the reduction to coverability in the proof of Theorem 125.

It is hopefully clear that Eve wins the coverability game played on $\mathscr{V}$ starting from $\ell'_0(\vec{0})$ and with target configuration $\ell_{\texttt{halt}}(\vec{0})$ if and only if the alternating Minsky machine halts.

Regarding non-termination games, we use essentially the same reduction. First observe that, if Eve can ensure reaching $\ell_{\texttt{halt}}$ in the alternating Minsky machine, then she can do so after at most $|\mathscr{L}|B^k$ steps. We therefore use a 'time budget': this is an additional component in $\mathscr{V}$ with associated unit vector $\vec{t}$. This component is initialised to $|\mathscr{L}|B^k = |\mathscr{L}|2^{k2^n}$ before the simulation, and decreases by one at every step; see Figure 11.16. We also add a self loop $\ell_{\texttt{halt}} \xrightarrow{\vec{0}} \ell_{\texttt{halt}}$. Then the only way to avoid the sink and thus to win the non-termination game is to reach $\ell_{\texttt{halt}}$.

We still need to extend our initialisation phase. It suffices for this to implement a gadget for k-meta-increments $\ell \xrightarrow{2^{k2^j} \cdot \vec{c}} \ell'$ and k-meta-decrements $\ell \xrightarrow{-2^{k2^j} \cdot \vec{c}} \ell'$; this is the same argument as in Lipton's construction, with a base case $\ell \xrightarrow{2^k} \ell'$ for $j = 0$. Then we initialise our time budget through $|\mathscr{L}|$ successive k-meta-increments $\ell \xrightarrow{2^{k2^n} \cdot \vec{t}} \ell'$. □

The proof of Theorem 125 relies crucially on the fact that the dimension is not fixed: although $k \geq 3$ suffices in the alternating Minsky machine, we need $O(|\mathscr{M}|)$ additional counters to carry out the reduction. A separate argument is thus needed in order to match the EXP upper bound of Theorem 123 in fixed dimension.

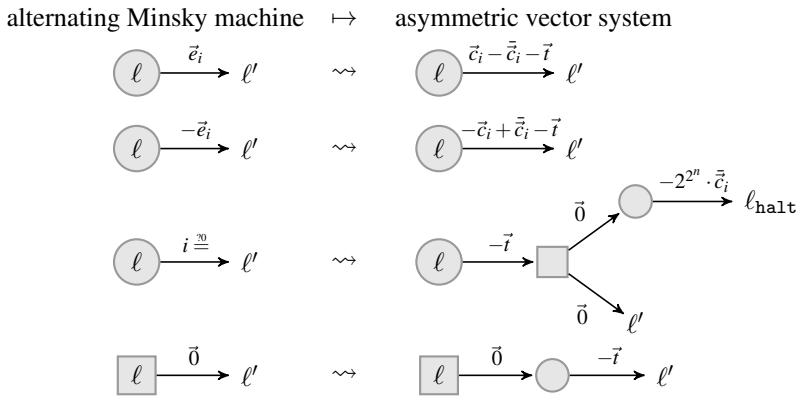


Figure 11.16: Schema of the reduction to non-termination in the proof of Theorem 125.

Theorem 126 (Fixed-dimensional coverability and non-termination asymmetric vector games are EXP-hard)**.** *Coverability, non-termination, and parity asymmetric vector games with given initial credit are* EXP*-hard in dimension $k \geq 2$.*

Proof. We exhibit a reduction from countdown games with given initial credit, which are EXP-complete by Theorem 113. Consider an instance of a configuration reachability countdown game: a countdown system $\mathscr{V} = (\mathscr{L}, A, \mathscr{L}_{\text{Eve}}, \mathscr{L}_{\text{Adam}}, 1)$ with initial configuration $\ell_0(n_0)$ and target configuration $\text{☺}(0)$—as seen in the proof of Theorem 113, we can indeed assume that the target credit is zero; we will also assume that Eve controls ☺ and that the only action available in ☺ is $\text{☺} \xrightarrow{-1} \text{☺}$. We construct an asymmetric vector system $\mathscr{V}'$ of dimension 2 such that Eve can ensure reaching $\text{☺}(0, n_0)$ from $\ell_0(n_0, 0)$ in $\mathscr{V}'$ if and only if she could ensure reaching $\text{☺}(0)$ from $\ell_0(n_0)$ in $\mathscr{V}$. The translation is depicted in Figure 11.17.

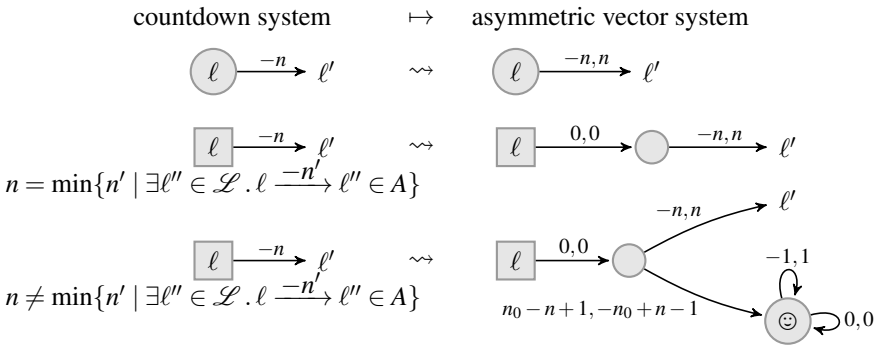


Figure 11.17: Schema of the reduction in the proof of Theorem 126.

The idea behind this translation is that a configuration $\ell(c)$ of $\mathscr{V}$ is simulated by a configuration $\ell(c, n_0 - c)$ in $\mathscr{V}'$. The crucial point is how to handle Adamś moves. In a configuration $\ell(c, n_0 - c)$ with $\ell \in \mathscr{L}_{\text{Adam}}$, according to the natural semantics of

$\mathscr{V}$, Adam should be able to simulate an action $\ell \xrightarrow{-n} \ell'$ if and only if $c \geq n$. Observe that otherwise if $c < n$ and thus $n_0 - c > n_0 - n$, Eve can play to reach ☺ and win immediately. An exception to the above is if n is minimal among the decrements in ℓ, because according to the natural semantics of $\mathscr{V}$, if $c < n$ there should be an edge to the sink, and this is handled in the second line of Figure 11.17.

Then Eve can reach $☺(0,n_0)$ if and only if she can cover $☺(0,n_0)$, if and only if she can avoid the sink thanks to the self loop $☺ \xrightarrow{0,0} ☺$. This shows the EXP-hardness of coverability and non-termination asymmetric vector games in dimension two; the hardness of parity follows from Remarks 21 and 22. □

11.5.3 Dimension One

Bibliographic references

Vector Addition Systems with States In their one-player version, i.e. in vector addition systems with states, all the games presented in Section 11.1 are decidable. With given initial credit, configuration reachability is simply called 'reachability' and was first shown decidable by Mayr [May81] (with simpler proofs in [Kos82, Lam92, Ler11]) and recently shown to be of non-elementary complexity [CLL$^+$19]. Coverability and non-termination are considerably easier, as they are EXPSPACE-complete [Lip76, Rac78] and so is parity [Hab97]. With existential initial credit, the problems are markedly simpler: configuration reachability becomes EXPSPACE-complete, while coverability is in NL and non-termination and parity can be solved in polynomial time by Theorem 120 using linear programming techniques [KS88].

Undecidability of Vector Games The undecidability results of Section 11.1.2 are folklore. One can find undecidability proofs in [ABd03, RSB05]; non-termination was called 'deadlock-freedom' by Raskin et al. [RSB05]. Configuration reachability is undecidable even in very restricted cases, like the *robot games* of [NPR16].

Succinct One-Counter Games One-dimensional vector systems are often called *one-counter nets* in the literature, by contrast with *one-counter automata* where zero tests are allowed. The EXPSPACE-completeness of succinct one-counter games was shown by Hunter [Hun15]. Countdown games were originally defined with given initial credit and a zero reachability objective, and shown EXP-complete in [JLS08]; see also [Kie13] for a variant called *hit-or-run games*. The hardness proofs for Theorems 113 and 114 are adapted from [JOS18], where countdown games with existential initial credit were first introduced.

Asymmetric Vector Games The asymmetric vector games of Section 11.3 appear under many guises in the literature: as 'and-branching' vector addition systems with states in [LMSS92], as 'vector games' in [Kan95], as 'B-games' in [RSB05], as 'single sided' vector addition games in [AMSS13], and as 'alternating' vector addition systems with states in [CS14].

The undecidability of configuration reachability shown in Section 11.3.1 was already proven by Lincoln et al. [LMSS92] and used to show the undecidability of propositional linear logic; Kanovich [Kan95, Kan16] refines this result to show the undecidability of the $(!, \oplus)$-Horn fragment of linear logic. Similar proof ideas are used for Boolean BI and separation logic in [LWG13, BK14].

Asymmetric Monotone Vector Games The notion of asymmetric infinite games over a well-quasi-ordered arena constitutes a natural extension of the notion of *well-structured systems* of [AČJT00] and [FS01], and was undertaken in [ABd03, RSB05]. The decidability of coverability and non-termination through wqo arguments like those of Fact 20 was shown by Raskin et al. [RSB05]. More advanced wqo techniques were needed for the first decidability proof of parity in [AMSS13]. See also [SS12] for more on the algorithmic uses of wqos.

By analysing the attractor computation of Section 11.3.2, one can show that Algorithm 11.1 works in 2EXP, thus matching the optimal upper bound from Theorem 123: this can be done using the Rackoff-style argument of [CS14] and the analysis of [BG11], or by a direct analysis of the attractor computation algorithm [LS19].

Energy Games An alternative take on energy games is to see a vector system $\mathscr{V} = (\mathscr{L}, A, \mathscr{L}_{\text{Eve}}, \mathscr{L}_{\text{Adam}}, k)$ as a finite arena with edges $\ell \xrightarrow{\vec{u}} \ell'$ coloured by $\vec{u}$, thus with set of colours $C \stackrel{\text{def}}{=} \mathbb{Z}^k$. For an initial credit $\vec{v}_0 \in \mathbb{N}^k$ and $1 \leq i \leq k$, the associated *energy objective* is then defined as

$$\mathsf{Energy}_{\vec{v}_0}(i) \stackrel{\text{def}}{=} \left\{ \pi \in E^\omega \;\middle|\; \forall n \in \mathbb{N} \,.\, \left(\vec{v}_0(i) + \sum_{0 \leq j \leq n} c(\pi)(i) \right) \geq 0 \right\} ,$$

that is, π is winning if the successive sums of weights on coordinate i are always non-negative. The *multi-energy objective* then asks for the play π to belong simultaneously to $\mathsf{Energy}_{\vec{v}_0}(i)$ for all $1 \leq i \leq k$. This is a multiobjective in the sense of the forthcoming Chapter 12. Multi-energy games are equivalent to non-termination games played on the arena $\mathscr{A}_{\mathbb{E}}(\mathscr{V})$ defined by the energy semantics.

The relationship with energy games was first observed in [AMSS13]. The equivalence with mean payoff games in dimension one was first noticed by Bouyer et al. [BFL$^+$08]. A similar connection in the multi-dimensional case was established in [CDHR10, VCD$^+$15] and will be discussed in Chapter 12.

Complexity Table 11.1 summarises the complexity results for asymmetric vector games. For the upper bounds with existential initial credit of Section 11.5.1, the existence of counterless winning strategies for Adam was originally shown by Brázdil et al. [BJK10] in the case of non-termination games; the proof of Lemma 136 is a straightforward adaptation using ideas from [CD12] to handle parities. An alternative proof through bounding games is presented in [CJLS17].

The coNP upper of Theorem 122 was shown soon after Brázdil et al.'s work by Chatterjee et al. [CDHR10] in the case of non-termination games. The extension of

Theorem 122 to parity was shown by [CRR14] by a reduction from parity to non-termination games somewhat reminiscent of ?. The proof of Theorem 122 takes a slightly different approach using Lemma 138 for finding non-negative cycles, which is a trivial adaptation of a result by Kosaraju and Sullivan [KS88]. The pseudo-polynomial bound of Corollary 21 is taken from [CJLS17].

For the upper bounds with given initial credit of Section 11.5.1, regarding coverability, the 2EXP upper bound of Theorem 123 was first shown by Courtois and Schmitz [CS14] by adapting Rackoff's technique for vector addition systems with states [Rac78]. Regarding non-termination, the first complexity upper bounds were shown by Brázdil et al. [BJK10] and were in kEXP, thus non-elementary in the size of the input. Very roughly, their argument went as follows: one can extract a pseudo-polynomial existential Pareto bound B in the one-player case from the proof of Theorem 120, from which the proof of Lemma 136 yields a $2^{|A|}(B+|\mathscr{L}|)$ existential Pareto bound in the two-player case, and finally by arguments similar to ? a tower of k exponentials on the given initial credit problem. The two-dimensional case with a unary encoding was shown a bit later to be in P by Chaloupka [Cha13]. Finally, a matching 2EXP upper bound (and pseudo-polynomial in any fixed dimension) was obtained by Jurdziński et al. [JLS15]. Regarding parity, Jančar [Jan15] showed how to obtain non-elementary upper bounds by reducing to the case of [BJK10], before a tight 2EXP upper bound (and pseudo-polynomial in fixed dimension with a fixed number of priorities) was shown in [CJLS17].

The coNP hardness with existential initial credit in Theorem 124 originates from [CDHR10]. The 2EXP-hardness of both coverability and non-termination games with given initial credit from Theorem 125 was shown in [CS14] by adapting Lipton's construction for vector addition systems with states [Lip76]; similar proofs can be found for instance in [DJLL12, BHSS12]. The hardness for EXP-hardness in dimension two was first shown by [FJLS11].

The NP $\cap$ coNP upper bounds in dimension one from Section 11.5.3 are due to Bouyer et al. [BFL$^+$08] for given initial credit and Chatterjee and Doyen [CD12] for existential initial credit.

Some Applications Besides their many algorithmic applications for solving various types of games, vector games have been employed in several fields to prove decidability and complexity results, for instance for linear, relevance, or separation logics [LMSS92, Kan95, Urq99, LWG13, BK14, Kan16], simulation and bisimulation problems [Kie13, AMSS13, CS14, JOS18], resource-bounded logics [ABDL18], orchestration synthesis [GVF$^+$18], as well as model-checking probabilistic timed automata [JLS08].

Table 11.1: The complexity of asymmetric vector games.

Game	Initial credit	Dimension		
		Fixed $k = 1$	Fixed $k \geq 2$	Arbitrary
configuration reachability	-	EXPSPACE-complete Theorem 118	undecidable Theorem 117 [LMSS92]	
coverability	existential		P-complete Theorem 121	
	given	in NP ∩ coNP	EXP-complete Theorems 123 and 126 [FJLS11, CS14]	2EXP-complete Theorems 123 and 125 [CS14]
non-termination	existential	in NP ∩ coNP [CD12]	in coNP	coNP-complete Theorems 122 and 124 [CDHR10]
	given	in NP ∩ coNP [BFL+08]	EXP-complete Theorems 123 and 126 [FJLS11, JLS15]	2EXP-complete Theorems 123 and 125 [CS14, JLS15]
parity	existential	in NP ∩ coNP [CD12]	in coNP	coNP-complete Theorems 122 and 124 [CDHR10, CRR14]
	given		EXP-complete Theorems 123 and 126 [FJLS11, CJLS17]	2EXP-complete Theorems 123 and 125 [CS14, CJLS17]

Part V

Multi

GAMES WITH MULTIPLE OBJECTIVES

Chapter 12

Games with multiple objectives

MICKAEL RANDOUR

Up to this chapter, we have mostly been interested in finding strategies that achieve a *single* objective or optimise a *single* payoff function. Our goal here is to discuss what happens when one goes further and wants to build strategies that (i) ensure *several objectives*, or (ii) provide *richer guarantees* than the simple worst-case or expectation ones used respectively in zero-sum games and Markov decision processes (MDPs).

Consider case (i). Such requirements arise naturally in applications: for instance, one may want to define a trade-off between the performance of a system and its energy consumption. A model of choice for this is the natural *multidimensional* extension of the games of Chapter 4, where we consider weight vectors on edges and combinations of objectives.

In case (ii), we base our study on stochastic models such as MDPs (Chapter 5). We will notably present how to devise controllers that provide strong guarantees in a worst-case scenario while behaving efficiently on average (based on a stochastic model of its environment built through statistical observations); effectively reconciling the rational antagonistic behaviour of Adam, used in games, with the stochastic interpretation of uncontrollable interaction at the core of MDPs.

Stepping into the *multi-objective* world is like entering a jungle: the sights are amazing but the wildlife is overwhelming. Providing an exhaustive account of existing multi-objective models and the latest developments in their study is a task doomed to fail: simply consider the combinatorial explosion of all the possible combinations based on the already non-exhaustive set of games studied in the previous chapters. Hence, our goal here is to guide the reader through their first steps in the jungle, highlighting the specific dangers and challenges of the multi-objective landscape, and displaying some techniques to deal with them. To that end, we focus on models studied in Chapter 2, Chapter 4, Chapter 5 and Chapter 11, and multi-objective settings that extend them. We favour simple, natural classes of problems, that already suffice to grasp the cornerstones of multi-objective reasoning.

Chapter outline In Section 12.1, we illustrate the additional complexity of multi-objective games and how relations between different classes of games that hold in the single-objective case often break as soon as we consider combinations of objectives.

The next two sections are devoted to the *simplest form* of multi-objective games: games with *conjunctions* of classical objectives. In Section 12.2, we present the classical case of multidimensional mean payoff and energy games, which preserve relatively nice properties with regard to their single-objective counterparts. In Section 12.3, we discuss the opposite situation of total payoff and shortest path games: their nice single-objective behaviour vanishes here.

In the last two sections, we explore a different meaning of *multi-objective* through so-called *rich behavioural models*. Our quest here is to find strategies that provide several types of guarantees, of different nature, for the same quantitative objective. In Section 12.4, we address the problem of *beyond worst-case synthesis*, which combines the rational antagonistic interpretation of two-player zero-sum games with the stochastic nature of MDPs. We will study the mean payoff setting and see how to construct strategies that ensure a strict worst-case constraint while providing the highest expected value possible. In Section 12.5, we briefly present *percentile queries*, which extend *probability threshold problems* in MDPs to their multidimensional counterparts. Interestingly, *randomised strategies* become needed in this context, whereas up to Section 12.5, we only consider deterministic strategies as they suffice.

We close the chapter with the usual bibliographic discussion and pointers towards some of the many recent advances in multi-objective reasoning.

12.1 From one to multiple dimensions

For the first part of this chapter, we consider *multidimensional quantitative games*. With regard to the formalism of Chapter 4, the only change to the arena is the set of colours associated with edges: we now have vectors in R^k where $k \in \mathbb{N}_{>0}$ is the *dimension* of the game. As before, for computational purposes, it makes sense to restrict our colouring to rational numbers, and for the sake of simplicity, we even consider *integers only* without loss of generality. Hence, $\mathfrak{c}\colon E \to \mathbb{Z}^k$.

For the weighted games of Chapter 4, where a single quantitative objective f is considered, we know that the value of the game exists. In most cases, optimal strategies do too, which makes the problems of computing the value and solving the game for a given threshold morally equivalent. In our simple multidimensional setting, we focus on *conjunctions* of objectives. Similarly to what we did in the one-dimension case, we will write $f_{\geq \vec{x}}$ with $\vec{x} \in \mathbb{Q}^k$ to define the (qualitative) winning condition

$$f_{\geq \vec{x}} = \bigcap_{i=1}^{k} \{\pi \in \mathrm{Paths}_\omega(G) \mid f_i(\pi) \geq \vec{x}_i\}$$

where $f_i(\pi)$ represents the evaluation of f on the sequence of colours in the i-th dimension and $\vec{x}_i$ represents the i-th component of vector $\vec{x}$. Hence we consider the natural semantics where we want to satisfy the original objective f component-wise.

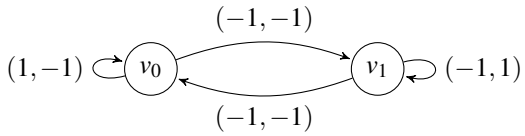


Figure 12.1: A simple multidimensional mean payoff game where Eve needs infinite memory to play (Pareto-)optimally.

Example 16. *Consider the simple one-player game in Figure 12.1 fitted with the mean payoff objective* $\texttt{MeanPayoff}^{-}$ *(recall that two variants exist depending on the use of lim-sup or lim-inf). Let us first recall that in the single-objective case, memoryless strategies suffice to play optimally (Corollary 6). In this game, such strategies permit to achieve payoffs* $(1,-1)$*,* $(-1,-1)$ *and* $(-1,1)$*. Intuitively,* $(-1,-1)$ *is not interesting since we can do better with* $(1,-1)$ *or* $(-1,1)$*. On the other hand, these two other payoffs are incomparable and thus should not be discriminated a priori. In the multi-objective world, there is usually no total order between the outcomes of a game — fixing a total order would actually boil down to transforming the game into a one-dimension game — which is why there is in general no optimal strategy but rather* Pareto-optimal *ones. Intuitively, a strategy is Pareto-optimal if there exists no other strategy yielding a payoff which is as good in all dimensions and strictly better in at least one dimension.*

Definition 30 (*Pareto-optimal strategy*)**.** *Given a k-dimension game* $\mathcal{G}$ *based on the conjunction of k maximising (w.l.o.g.) quantitative objectives* $(f_i)_{i=1}^{k}$*, a strategy* σ *for Eve is said to be* Pareto-optimal *if it guarantees a payoff* $\vec{x} \in \mathsf{R}^k$ *such that for all other strategy* σ' *of Eve ensuring payoff* $\vec{x}' \neq \vec{x}$*, it holds that* $\vec{x}_i > \vec{x}'_i$ *for some dimension* $i \in \{1,\dots,k\}$*.*

The concept of Pareto-optimality has an important consequence on multi-objective problems: the correspondence between solving a value problem and computing an optimal strategy that holds in the single-objective case does not carry over. Indeed, one may now be interested in computing the *Pareto frontier* consisting of all Pareto vectors achievable by Eve. This comes at great cost complexity-wise as this frontier may include many points, and in some settings, even an *infinite number of Pareto vectors* (see Section 12.5 for examples), sometimes forcing us to resort to approximation. This requires specific techniques that go beyond the focus of this chapter, hence in the following we mostly discuss the *value problem*, also referred to as 'solving the game' for a given threshold vector.

Example 17. *Let us go back to Example 16 and fix objective* $\texttt{MeanPayoff}^{-}_{\geq \vec{x}}$ *where* $\vec{x} = (0,0)$*. As discussed before, this threshold cannot be achieved by a memoryless strategy. Actually, this is also the case for any* finite-memory *strategy. Indeed, any finite-memory strategy induces an ultimately periodic play, where either (a) the periodic part only visits* v_0 *(resp.* v_1*), yielding payoff* $(1,-1)$ *(resp.* $(-1,1)$*) thanks to prefix independence of the mean payoff (Chapter 4), or (b) it visits both, in which case the mean payoff is of*

the form

$$\vec{y} = \texttt{MeanPayoff}^{-}(\pi) = \frac{a \cdot (1,-1) + 2 \cdot b \cdot (-1,-1) + c \cdot (-1,1)}{a + 2 \cdot b + c}$$

where $a, c \in \mathbb{N}$ and $b \in \mathbb{N}_{>0}$. Observe that $\vec{y}_1 + \vec{y}_2 = -4 \cdot b/(a + 2 \cdot b + c)$, which is strictly less than zero for any value of the parameters. Hence $\vec{x} = (0,0)$ is not achievable. Now consider what happens with infinite memory: let σ be the strategy of Eve that visits ℓ times v_0, then ℓ times v_1, and then repeats forever with increasing values of ℓ. The mean payoff of the resulting play is the limit of the previous equation when $a = c = \ell$ tends to infinity, with $b = 1$: intuitively, the switch between v_0 and v_1 becomes negligible in the long run and the mean payoff is $\frac{1}{2} \cdot (1,-1) + \frac{1}{2} \cdot (-1,1) = (0,0)$.

Remark 23. *While Eve cannot achieve $(0,0)$ with finite memory, she can achieve (i.e., ensure at least) any payoff $(-\varepsilon, -\varepsilon)$ for $0 < \varepsilon < 1$, using sufficient memory: for instance, by taking $b = 1$ and $a = c = \lceil \frac{1}{\varepsilon} - 1 \rceil$. In that sense, the payoff $\vec{x} = (0,0)$ achievable by an infinite-memory strategy can be seen as the supremum of payoffs achievable by finite-memory strategies. Actually, this is exactly how we defined strategy σ: Eve plays according to an infinite sequence of finite-memory strategies parametrised by ℓ, such that each strategy of the sequence ensures mean payoff $(-\varepsilon, -\varepsilon)$, with $\varepsilon \to 0$ when $\ell \to \infty$.*

Example 18. *The reasoning above holds similarly for $\texttt{MeanPayoff}^{+}$. With finite-memory, the lim-sup variant coincides with the lim-inf one: because the play is ultimately periodic, the limit exists. With infinite-memory, Eve can actually achieve the payoff $\vec{x}' = (1,1)$, witnessing a gap with the lim-inf variant. To do so, she has to play a strategy that alternates between v_0 and v_1 while staying in each vertex for a sufficiently long period such that the current mean over the corresponding dimension gets close to 1. Getting these means closer and closer to 1 and using the lim-sup component-wise then suffices to achieve payoff $\vec{x}'$. This is in stark contrast to the lim-inf variant, which cannot achieve any payoff $(\varepsilon, \varepsilon)$ for $\varepsilon > 0$ (the Pareto vectors correspond to linear combinations of simple cycles, as hinted before).*

Theorem 127. *Multidimensional mean payoff games require infinite-memory strategies for Eve. Furthermore, the lim-inf and lim-sup variants are not equivalent, i.e., their winning regions are in general not identical.*

This theorem already shows the first signs of our single-objective assumptions crumbling in the multi-objective world: we jump from memoryless determinacy to needing infinite memory, and objectives that were equivalent both in games and MDPs turn out to be different here. Buckle up, as this was only our first step.

12.2 Mean payoff and energy

Another well-known equivalence in one-dimension is the one between mean payoff and energy games (in the existential initial credit form), mentioned in Chapter 4. The reduction is trivial: Eve has a winning strategy (and an initial credit) in the energy game

if and only if she has a strategy to ensure mean payoff at least equal to zero in the mean payoff game played over the same arena. Intuitively, the mean payoff strategy of Eve has to reach a subgame where she can ensure that all cycles formed are non-negative (see cycle games in Chapter 4). The initial credit (which can be as high as Eve wants) offsets the cost of reaching such a subgame as well as the low point of cycles in it (which can be negative but is bounded).

How does it fare in multiple dimensions? The study of vector games with energy semantics in Chapter 11 gives the following result.

Theorem 128. *Solving multidimensional energy games is* coNP*-complete. Exponential-memory strategies suffice and are required for Eve, and memoryless ones suffice for Adam.*

Based on Theorem 127 and Example 17, it is clear that the aforementioned equivalence holds no more, as mean payoff games benefit from infinite memory while energy games do not. In Example 17, the strategy that achieves $\vec{x} = (0,0)$ for the mean payoff does so by switching infinitely often but with decreasing frequency between v_0 and v_1: the switch becomes negligible in the limit which is fine for the mean payoff. Still, this would lead the energy to drop below zero eventually, whatever the initial credit chosen by Eve, hence showing why the reduction does not carry over.

12.2.1 Finite memory

Game-theoretic models are generally used in applications, such as controller synthesis, where one actually wants to *implement* a winning strategy when it exists. This is why finite-memory strategies have a particular appeal. Hence it is interesting to study what happens when we restrict Eve to finite-memory strategies in multidimensional mean payoff games.

We first observe that when both players use finite-memory strategies, the resulting play is ultimately periodic, hence the lim-inf and lim-sup variants coincide (the limit exists) and take the value of the mean over the periodic part.

Proposition 11. *The lim-sup and lim-inf variants of multidimensional mean payoff games coincide under finite memory, i.e., their winning regions are identical in all games.*

We now go back to the relationship with energy games. In the following, we write $\vec{0}$ for the k-dimension vector $(0,\dots,0)$. When restricting both players to finite memory, we regain the equivalence between mean payoff and energy games by a natural extension of the argument sketched above for one-dimension games.

Theorem 129. *For all arena and initial vertex, Eve has a winning strategy for the existential initial credit multidimensional energy game if and only if she has a finite-memory winning strategy for the multidimensional (lim-inf or lim-sup) mean payoff game with threshold* $\vec{0}$*.*

Proof. Let $\mathscr{A}$ be an arena coloured by integer vectors of dimension k and v_0 be the initial vertex. We first consider the left-to-right implication. Assume that Eve has a

strategy σ and some initial credit $\vec{c}_0 \in \mathbb{N}^k$ such that she wins the energy objective over $\mathscr{A}$. By Theorem 128, we may assume σ to be finite-memory and $\mathscr{M} = (M, m_0, \delta)$ to be its memory structure. Let $\mathscr{A}_\sigma$ be the classical product of the arena with this memory structure ($\mathscr{A} \times \mathscr{M}$) restricted to the choices made by σ. We claim that any cycle in $\mathscr{A}_\sigma$ is non-negative in all dimensions (we simply project paths of $\mathscr{A}_\sigma$ to C^ω to interpret them as we do for paths in $\mathscr{A}$). By contradiction, assume that there exists a cycle whose sum of weights is strictly negative in some dimension. Then the play reaching this cycle and looping in it forever is a play consistent with σ that is losing for the energy objective, contradicting the hypothesis. Hence, it is indeed the case that all reachable cycles in $\mathscr{A}_\sigma$ are non-negative in all dimensions. Thus, σ ensures mean payoff at least equal to zero in all dimensions (for lim-inf and lim-sup variants).

In the opposite direction, assume that σ is a finite-memory winning strategy for $\texttt{MeanPayoff}^-_{\geq\vec{0}}$ (or equivalently $\texttt{MeanPayoff}^+_{\geq\vec{0}}$). Using the same argument as before, we have that all cycles in $\mathscr{A}_\sigma$ are non-negative. Therefore there exists some initial credit $\vec{c}_0 \in \mathbb{N}^k$ such that σ satisfies the energy objective. As a trivial bound, one may take initial credit $|V| \cdot |M| \cdot W$ in all dimensions, where $|V|$ is the number of vertices of $\mathscr{A}$, $|M|$ the number of memory states of $\mathscr{M}$, and W is the largest absolute weight appearing in the arena: this quantity bounds the lowest sum of weights achievable under an acyclic path. □

Observe that the finite-memory assumption is crucial to lift mean payoff winning strategies to the energy game. Intuitively, the reasoning would break for a strategy like the one used in Example 17 because the memory structure would need to be infinite and $\mathscr{A}_\sigma$ would actually not contain any cycle but an infinite path of ever-decreasing energy such that no bound on the initial credit could be established.

Also, note that Theorem 129 makes no mention of the specific variant of mean payoff used. This is because both players play using finite-memory: Eve by hypothesis and Adam thanks to the equivalence and Theorem 128. Hence, Proposition 11 applies. To sum up, we obtain the following.

Corollary 22. *Solving multidimensional mean payoff games under finite-memory is* coNP*-complete. Exponential-memory strategies suffice and are required for Eve, and memoryless ones suffice for Adam.*

12.2.2 Infinite memory

We now turn to the general case, where Eve is allowed to use infinite memory. By Example 18, we already know that lim-sup and lim-inf variants are not equivalent. We will cover the lim-sup case in details and end with a brief overview of the lim-inf one.

Lim-sup variant

Without loss of generality, we fix the objective $\texttt{MeanPayoff}^+_{\geq\vec{0}}$ (one can always modify the weights in the arena and consider the shifted-game with threshold zero). We have seen in Example 18 that Eve could focus on each dimension independently and alternatively in such a way that in the limit, she obtains the supremum in each dimension. This is the core idea that we will exploit.

Lemma 139. *Let $\mathscr{A}$ be an arena such that from all vertex $v \in V$ and for all dimension i, $1 \leq i \leq k$, Eve has a winning strategy for $\{\pi \in \mathrm{Paths}_\omega(G) \mid \mathtt{MeanPayoff}_i^+(\pi) \geq 0\}$. Then, from all vertex $v \in V$, she has a winning strategy for $\mathtt{MeanPayoff}^+_{\geq \vec{0}}$.*

Hence, being able to win in each dimension *separately* suffices to guarantee winning in all dimensions *simultaneously*. Note that the converse is obvious.

Proof. For each vertex $v \in V$ and dimension i, $1 \leq i \leq k$, let σ_i^v be a winning strategy for Eve from v for $\{\pi \in \mathrm{Paths}_\omega(G) \mid \mathtt{MeanPayoff}_i^+(\pi) \geq 0\}$.

Let $T_{\sigma_i^v}$ be the infinite tree obtained by *unfolding* σ_i^v: it represents all plays consistent with this strategy. Formally, such a tree is obtained inductively as follows:

- The root of the tree represents v.
- Given a node[1] η representing the branch (i.e., prefix of play) ρ starting in vertex v and ending in vertex v_η, we add children as follows:
  - if $v_\eta \in V_{\mathrm{Eve}}$, η has a unique child representing the vertex $\mathrm{Out}(e)$ reached through edge $e = \sigma_i^v(\rho)$;
  - otherwise η has one child for each possible successor of v_η, i.e., for each $\mathrm{Out}(e)$ such that $e \in E$ and $\mathrm{In}(e) = v_\eta$.

For $\varepsilon > 0$, we declare a node η of $T_{\sigma_i^v}$ to be *ε-good* if the mean over dimension i on the path from the root to η is at least $-\varepsilon$ (as usual, we project this path to C^ω to evaluate it). For $\ell \in \mathbb{N}$, let $\widehat{T}^{i,\ell}_{v,\varepsilon}$ be the tree obtained from $T_{\sigma_i^v}$ by removing all descendants of ε-good nodes that are at depth at least ℓ: hence, all branches of $\widehat{T}^{i,\ell}_{v,\varepsilon}$ have length at least ℓ and their leaves are ε-good.

We first show that $\widehat{T}^{i,\ell}_{v,\varepsilon}$ is a finite tree. By König's Lemma [Kön36], we only need to show that every branch is finite. By contradiction, assume it is not the case and there exists some infinite branch. By construction, it implies that this branch contains no ε-good node after depth ℓ. Thus, the corresponding play π, which is consistent with σ_i^v, necessarily has $\mathtt{MeanPayoff}_i^+(\pi) \leq -\varepsilon$. This contradicts the hypothesis that σ_i^v is winning for dimension i. Hence the tree is indeed finite.

Based on these finite trees, we now build an infinite-memory strategy for Eve that will be winning for the conjunct objective $\mathtt{MeanPayoff}^+_{\geq \vec{0}}$: it is presented as Algorithm 12.1.

Recall that W is the largest absolute weight in the game. Consider the situation whenever an iteration of the for-loop ends. Let M be the number of steps the play followed σ_i^v during this loop execution. Then, the mean payoff in dimension i is at least $\frac{-L\cdot W - M\cdot\varepsilon}{L+M} \geq \frac{-L\cdot W - M\cdot\varepsilon}{M}$. Since $M \geq \frac{L\cdot W}{\varepsilon}$ by definition, we obtain that the mean payoff in dimension i is at least $-2\cdot\varepsilon$.

Observe that since all trees are finite, we always exit the for-loop eventually, hence ε tends to zero. Therefore, the supremum mean payoff is at least zero in all dimensions, which makes this strategy winning for $\mathtt{MeanPayoff}^+_{\geq \vec{0}}$. □

[1] Nodes refer to the tree, vertices to the arena.

Algorithm 12.1: Winning strategy σ for $\texttt{MeanPayoff}^{+}_{\geq\vec{0}}$

$\varepsilon \leftarrow 1$
loop
  for $i = 1$ *to* k **do**
    Let v be the current vertex, L the length of the play so far
    $\ell \leftarrow \left\lceil \frac{L \cdot W}{\varepsilon} \right\rceil$
    Play according to σ_i^v until a leaf of $\widehat{T}_{v,\varepsilon}^{i,\ell}$ is reached
  $\varepsilon \leftarrow \frac{\varepsilon}{2}$

This construction is tight in the sense that infinite memory is needed for Eve, as previously proved. For Adam, we show a better situation. The proof scheme will also be the base of the upcoming algorithm.

Lemma 140. *Memoryless strategies suffice for Adam in multidimensional lim-sup mean payoff games.*

Proof. The proof works by induction on the number of vertices of the arena. The base case $|V| = 1$ is trivial. Assume the only vertex belongs to Adam. If there exists a self loop (recall we allow several edges per pair of vertices) which has a negative weight on some dimension, Adam wins by looping on it forever. In the opposite case, he cannot win.

Now assume $|V| \geq 2$. For $i \in \{1,\dots,k\}$, let W_{Adam}^i be the winning region of Adam for the complement of $\{\pi \in \text{Paths}_\omega(G) \mid \texttt{MeanPayoff}_i^+(\pi) \geq 0\}$, i.e., the region where Adam has a strategy to force a strictly negative mean payoff in dimension i (as studied in Chapter 4). Let $W_{\text{Adam}}^{\text{disj}} = \bigcup_{i=1}^k W_{\text{Adam}}^i$. We have two cases.

First, $W_{\text{Adam}}^{\text{disj}} = \emptyset$. Then, Eve can win all one-dimension games from everywhere and by Lemma 139, she can also win for $\texttt{MeanPayoff}^{+}_{\geq\vec{0}}$. Thus, Adam has no winning strategy.

Second, $W_{\text{Adam}}^{\text{disj}} \neq \emptyset$. Then, there exists $i \in \{1,\dots,k\}$ such that $W_{\text{Adam}}^i \neq \emptyset$. In this set, Adam has a memoryless winning strategy τ_i to falsify the winning condition $\{\pi \in \text{Paths}_\omega(G) \mid \texttt{MeanPayoff}_i^+(\pi) \geq 0\}$ (because one-dimension mean payoff games are memoryless determined, as proved in Corollary 6). This strategy also falsifies $\texttt{MeanPayoff}^{+}_{\geq\vec{0}}$, hence W_{Adam}^i is part of the winning region for Adam — we denote it W_{Adam}, as usual. By prefix independence of the mean payoff, the attractor $W_{\text{Adam}}^{i,\text{Pre}} = \text{Attr}_{\text{Adam}}(W_{\text{Adam}}^i)$ is also part of W_{Adam}. We denote by τ_{Pre} the corresponding attractor strategy of Adam. Moreover, the graph restricted to $V \setminus W_{\text{Adam}}^{i,\text{Pre}}$ constitutes a proper arena $\mathscr{A}'$.

Let W'_{Adam} be the winning region of Adam in $\mathscr{A}'$ for the original winning condition $\texttt{MeanPayoff}^{+}_{\geq\vec{0}}$. The arena $\mathscr{A}'$ has strictly less vertices than $\mathscr{A}$ since we removed the non-empty region W_{Adam}^i. Hence we can apply the induction hypothesis: Adam has a memoryless winning strategy τ' in W'_{Adam}. The region $V \setminus (W_{\text{Adam}}^{i,\text{Pre}} \cup W'_{\text{Adam}})$ is winning for Eve in $\mathscr{A}'$ by determinacy. But it is also winning in $\mathscr{A}$, i.e., the original game, since

Adam cannot force the play to go in $W_{\text{Adam}}^{i,\text{Pre}}$ from there (otherwise it would be part of the attractor too).

We define the following memoryless strategy for Adam, which we claim is winning from $W_{\text{Adam}} = W_{\text{Adam}}^{i,\text{Pre}} \cup W'_{\text{Adam}}$:

$$\tau(v) = \begin{cases} \tau_{\text{Pre}}(v) & \text{if } v \in W_{\text{Adam}}^{i,\text{Pre}} \setminus W_{\text{Adam}}^{i}, \\ \tau_i(v) & \text{if } v \in W_{\text{Adam}}^{i}, \\ \tau'(v) & \text{if } v \in W'_{\text{Adam}}. \end{cases}$$

Since we already know that Eve wins from $V \setminus W_{\text{Adam}}$, it remains to prove that τ is winning from W_{Adam} to conclude. Consider any play π consistent with τ and starting in W_{Adam}. Two cases are possible. First, the play eventually reaches W_{Adam}^{i} and Adam switches to τ_i: then prefix independence of the mean payoff guarantees that Adam wins. Second, the play never reaches W_{Adam}^{i}: then π necessarily stays in $\mathscr{A}'$, and τ' is winning from W'_{Adam} in $\mathscr{A}'$. Therefore, τ does win from everywhere in W_{Adam}, while being memoryless, which ends the proof. □

We use the core reasoning of this proof to build an algorithm solving multidimensional lim-sup mean payoff games (Algorithm 12.2). It uses as a black box a routine that computes (in pseudo-polynomial time) the winning vertices for Eve in one-dimension mean payoff games. This subalgorithm, presented in Section 4.3, is here dubbed `SolveOneDimMeanPayoff`, and takes as parameters the arena and the considered dimension.

Algorithm 12.2: Solver for multidimensional lim-sup mean payoff games

Data: Arena $\mathscr{A}$ with vertices V
Result: W_{Eve}, the winning region of Eve for $\texttt{MeanPayoff}^{+}_{\geq \vec{0}}$
$\mathscr{A}' \leftarrow \mathscr{A}$; $V' \leftarrow V$
repeat
  `LosingVertices` $\leftarrow$ `false`
  for $i = 1$ *to* k **do**
    $W_{\text{Adam}}^{i} \leftarrow V' \setminus \texttt{SolveOneDimMeanPayoff}(\mathscr{A}', i)$
    if $W_{\text{Adam}}^{i} \neq \emptyset$ **then**
      $V' \leftarrow V' \setminus W_{\text{Adam}}^{i}$
      $\mathscr{A}' \leftarrow \mathscr{A}'[V']$ `/* Restriction of` $\mathscr{A}'$ `to` V' `*/`
      `LosingVertices` $\leftarrow$ `true`
until `LosingVertices` $=$ `false`
return V'

Intuitively, we iteratively remove vertices that are declared losing for Eve because Adam can win on some dimension from them. Since removing vertices based on some dimension i may decrease the power of Eve and her ability to win for another dimension i', we need the outer loop: in the end, we ensure that V' contains exactly all the vertices from which Eve has a winning strategy for each dimension. By Lemma 139 and the proof of Lemma 140, we know that this is equal to W_{Eve}.

We recall (Section 1.7) that, given an arena $\mathscr{A}$ and a set of vertices X, $\mathscr{A}[X]$ denotes the subarena induced by X.

Remark 24. *The restriction $\mathscr{A}'[V']$ induces a proper subarena. Indeed, we have that $W^i_{\text{Adam}} = Attr_{\text{Adam}}(W^i_{\text{Adam}})$ since any vertex v from which Adam can force to reach W^i_{Adam} also belongs to W^i_{Adam} by prefix independence of the mean payoff.*

We wrap up with the following theorem.

Theorem 130. *Solving multidimensional lim-sup mean payoff game is in* $\mathsf{NP} \cap \mathsf{coNP}$. *Infinite-memory strategies are required for Eve and memoryless ones suffice for Adam. Furthermore, the winning regions can be computed in pseudo-polynomial time, through at most $|V| \cdot k$ calls to an algorithm solving one-dimension mean payoff games.*

Proof. The correctness of Algorithm 12.2 follows from Lemma 139 and Lemma 140, and its complexity is trivial to assess, using `SolveOneDimMeanPayoff` as a pseudo-polynomial black-box. The memory bounds follow from Lemma 139, Lemma 140 and Theorem 127. Hence, only the $\mathsf{NP} \cap \mathsf{coNP}$ membership remains. Recall that the decision problem under study is: given an arena $\mathscr{A}$ and an initial vertex v_0, does v_0 belong to W_{Eve} or not?

We first prove that the problem is in NP. A non-deterministic algorithm guesses the winning region W_{Eve} containing v_0 and witness memoryless strategies σ_i for all dimensions (we know that memoryless strategies suffice by Corollary 6). Then, it checks for every dimension i, for every vertex $v \in W_{\text{Eve}}$, that σ_i is winning. This boils down to solving a polynomial number of one-player one-dimension mean payoff games for Adam over the arenas $\mathscr{A}_{\sigma_i}$ obtained by fixing σ_i.[2] As noted in Section 4.3, it can be done in polynomial time using Karp's algorithm for finding the minimum cycle mean in a weighted digraph [Kar78]. By Lemma 139, we know that if the verification checks out, Eve has a winning strategy in W_{Eve} for objective $\texttt{MeanPayoff}^+_{\geq\vec{0}}$.

Finally, we prove coNP membership. The algorithm guesses a memoryless winning strategy τ for Adam (from v_0). The verification then consists in checking that Eve has no winning strategy in the arena $\mathscr{A}_\tau$. This can be done using Algorithm 12.2, through $|V| \cdot k$ calls to `SolveOneDimMeanPayoff`. In this case however, such calls only need to solve *one-player* one-dimension mean payoff games for Eve, which again can be done in polynomial time, resorting to Karp's algorithm. Thus, the verification takes polynomial time in total, and coNP membership follows. □

Lim-inf variant

For the sake of conciseness, we give only a brief sketch. Without loss of generality, we fix the objective $\texttt{MeanPayoff}^-_{\geq\vec{0}}$. We know that infinite-memory strategies are needed for Eve by Theorem 127. Again, things look better for Adam.

Lemma 141. *Memoryless strategies suffice for Adam in multidimensional lim-inf mean payoff games.*

[2] The arena $\mathscr{A}_\sigma$ induced by applying a strategy σ on arena $\mathscr{A}$ can be obtained through the product $\mathscr{A} \times \mathscr{M}$, with $\mathscr{M}$ the memory structure of σ, as presented in Section 1.6.

Sketch. We mention the sketch as it is interesting in its own right. Recall that Chapter 4 presented a general recipe, due to Gimbert and Zielonka [GZ04, GZ05], to prove memoryless determinacy. Clearly, such a recipe cannot work here due to Theorem 127. Still, a similar result by Kopczyński deals with *half-positional determinacy* [Kop06]. This new recipe states that if the objective of Eve is both *prefix independent* and *convex*, then memoryless strategies suffice for Adam. We already know that $\texttt{MeanPayoff}^{-}_{\geq\vec{0}}$ is prefix independent. An objective is said to be convex if it is closed under combinations (shuffling): if two infinite sequences of colours $\pi = \rho_1\rho_2\dots$ and $\pi' = \rho'_1\rho'_2\dots$, with all ρ_i, ρ'_i being finite prefixes, belong to the objective, then $\pi'' = \rho_1\rho'_1\rho_2\rho'_2\dots$ does too. Conjunctions of lim-inf mean payoff are convex, hence the result applies here. Note that this approach corresponds to the one presented in Theorem 16: submixing objectives are also called concave, and Eve's objective being convex implies that Adam's objective is concave; hence the reasoning above. □

Remark 25. *Lim-sup mean payoff objectives are not convex, hence the ad-hoc proof in Lemma 140. Consider the integer sequence* $\pi = (2)^{5^0}(-4)^{5^1}(2)^{5^2}(-4)^{5^3}\dots$ *where the length of the i-th sequence of numbers is* 5^{i-1}*. One can prove that at the end of each sequence of* 2*'s (resp.* -4*'s), the mean is above (and tends to)* 1 *(resp. is exactly* -3*). Let* π' *be the sequence obtained by swapping all* 2*'s and* -4*'s. We have* $\texttt{MeanPayoff}^{+}(\pi) = \texttt{MeanPayoff}^{+}(\pi') \geq 1$*, hence* $\pi, \pi' \in \texttt{MeanPayoff}^{+}_{\geq 0}$*.*

Still, by shuffling π *and* π' *in one-one alternation, we build* $\pi'' = 2, -4, 2, -4\dots$*, which is such that* $\texttt{MeanPayoff}^{+}(\pi'') = -1$*, hence* $\pi'' \notin \texttt{MeanPayoff}^{+}_{\geq 0}$*. Hence lim-sup mean payoff is not convex.*

Complexity-wise, multidimensional lim-inf mean payoff games look a lot like multidimensional energy games, even though we proved they are not equivalent without memory restrictions.

Theorem 131. *Solving multidimensional lim-inf mean payoff games is* coNP*-complete. Infinite-memory strategies are required for Eve and memoryless ones suffice for Adam.*

We discussed memory through Example 17 and Lemma 141. The coNP-hardness can be shown through a reduction from 3UNSAT similar to the one used for existential initial credit multidimensional energy games in Section 11.5. The matching upper bound relies on memoryless strategies being sufficient for Adam, and the capacity to solve one-player instances of multidimensional lim-inf mean payoff games in polynomial time. The latter problem is addressed by reduction to detecting non-negative multi-cycles in graphs (which can be done in polynomial time based on [KS88]).

Wrap-up

We have seen that multidimensional mean payoff games and multidimensional energy games behave relatively well. Sure, infinite memory is needed for Eve in general for the former, but complexity-wise, the gap with one-dimension games is small and even non-existent for the lim-sup variant. Furthermore, if we are interested in finite-memory strategies, the equivalence with energy games is preserved. Hence, we may say that both mean payoff and energy games hold up nicely in the multidimensional world.

12.3 Total payoff and shortest path

In this section, we turn to two other objectives deeply studied in Chapter 4: we study total payoff and shortest path games. We will see that the multidimensional setting has dire consequences for both.

12.3.1 Total payoff vs. mean payoff

We start with total payoff games. As for the mean payoff, we explicitly consider the two variants, $\texttt{TotalPayoff}^+$ and $\texttt{TotalPayoff}^-$, for the lim-sup and lim-inf definitions respectively. While Chapter 4 was written using the lim-sup variant, all results are identical for the lim-inf one in one-dimension games [GS09a].

Recall that one-dimension total payoff games are memoryless determined and solving them is in $\mathsf{NP} \cap \mathsf{coNP}$ (even in $\mathsf{UP} \cap \mathsf{coUP}$ [GS09a]). Furthermore, Chapter 4 taught us that total payoff can be seen as a *refinement* of mean payoff, as it permits to reason about low (using the lim-inf variant) and high (using the lim-sup one) points of partial sums along a play when the mean payoff is zero. We formalize this relationship in the next lemma, and study what happens in multiple dimensions.

Lemma 142. *Fix an arena $\mathscr{A}$ and an initial vertex $v_0 \in V$. Let A, B, C and D denote the following assertions.*

A. Eve has a winning strategy for $\texttt{MeanPayoff}^+_{\geq \vec{0}}$.

B. Eve has a winning strategy for $\texttt{MeanPayoff}^-_{\geq \vec{0}}$.

C. There exists $\vec{x} \in \mathbb{Q}^k$ such that Eve has a winning strategy for $\texttt{TotalPayoff}^-_{\geq \vec{x}}$.

D. There exists $\vec{x} \in \mathbb{Q}^k$ such that Eve has a winning strategy for $\texttt{TotalPayoff}^+_{\geq \vec{x}}$.

In one-dimension games ($k = 1$), all four assertions are equivalent. In multidimensional ones ($k > 1$), the only implications that hold are: $C \implies D \implies A$ and $C \implies B \implies A$. All other implications are false in general.

Lemma 142 is depicted in Figure 12.2: the only implications that carry over to multiple dimensions are depicted by solid arrows.

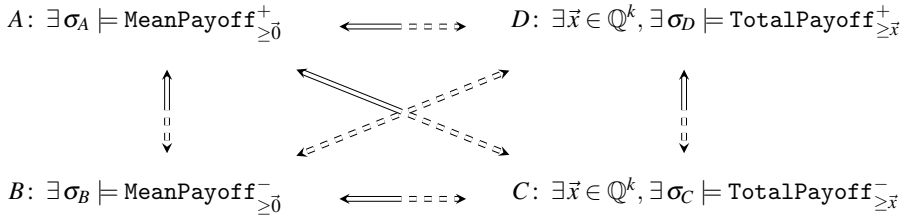


Figure 12.2: Equivalence between mean payoff and total payoff games. Dashed implications are only valid in one-dimension games. We use $\sigma \models \Omega$ as a shortcut for 'σ is winning from v_0 for Ω'.

Proof. The implications that remain true in multiple dimensions are the trivial ones. First, satisfaction of the lim-inf version of a given objective clearly implies satisfaction of its lim-sup version by definition. Hence, $B \implies A$ and $C \implies D$. Second, consider a play $\pi \in \texttt{TotalPayoff}^{-}_{\geq \vec{x}}$ (resp. $\texttt{TotalPayoff}^{+}_{\geq \vec{x}}$) for some $\vec{x} \in \mathbb{Q}^k$. For all dimension $i \in \{1, \ldots, k\}$, the corresponding sequence of mean payoff infima (resp. suprema) over prefixes can be *lower-bounded* by a sequence of elements of the form $\frac{\vec{x}_i}{\ell}$ with ℓ the length of the prefix. We can do this because the sequence of total payoffs over prefixes is a sequence of integers: it always achieves the value of its limit $\vec{x}_i$ instead of only tending to it asymptotically as could a sequence of rationals (such as the mean payoffs). Since $\frac{\vec{x}_i}{\ell}$ tends to zero over an infinite play, we do have that $\pi \in \texttt{MeanPayoff}^{-}_{\geq \vec{0}}$ (resp. $\texttt{MeanPayoff}^{+}_{\geq \vec{0}}$). Thus, $C \implies B$ and $D \implies A$. Along with the transitive closure $C \implies A$, these are all the implications preserved in multidimensional games.

In one-dimension games, all assertions are equivalent. As seen before, lim-inf and lim-sup mean payoff games coincide as memoryless strategies suffice for both players. Thus, we add $A \implies B$ and $D \implies B$ by transitivity. Second, consider a memoryless (w.l.o.g.) strategy σ_B for Eve for $\texttt{MeanPayoff}^{-}_{\geq \vec{0}}$. Let π be any consistent play. Then all cycles in π are non-negative, otherwise Eve cannot ensure winning with σ_B (because Adam could pump the negative cycle). Thus, the sum of weights along π is at all times bounded from below by $-(|V|-1) \cdot W$ (for the longest acyclic prefix), with W the largest absolute weight, as usual. Therefore, we have $B \implies C$, and we obtain all other implications by transitive closure.

For multidimensional games, all dashed implications are false.

1. To show that implication $D \implies B$ does not hold, consider the Eve-owned one-player game where $V = \{v\}$ and the only edges are two self loops of weights $(1,-2)$ and $(-2,1)$. Clearly, any finite vector $\vec{x} \in \mathbb{Q}^2$ for $\texttt{TotalPayoff}^{+}_{\geq \vec{x}}$ can be achieved by an infinite-memory strategy consisting in playing both loops successively for longer and longer periods, each time switching after getting back above threshold $\vec{x}$ in the considered dimension. However, it is impossible to build any strategy, even with infinite memory, that satisfies $\texttt{MeanPayoff}^{-}_{\geq \vec{0}}$ as the lim-inf mean payoff would be at best a linear combination of the two cycle values, i.e., strictly less than zero in at least one dimension in any case.

2. Lastly, consider the game in Figure 12.1 where we modify the weights to add a third dimension with value 0 on the self loops and -1 on the other edges. As already proved, the strategy that plays for ℓ steps in the left cycle, then goes for ℓ steps in the right one, then repeats for $\ell' > \ell$ and so on, is a winning strategy for $\texttt{MeanPayoff}^{-}_{\geq \vec{0}}$. Nevertheless, for any strategy of Eve, the play is such that either (i) it only switches between v_0 and v_1 a finite number of times, in which case the sum in dimension 1 or 2 decreases to infinity from some point on; or (ii) it switches infinitely often and the sum in dimension 3 decreases to infinity. In both cases, objective $\texttt{TotalPayoff}^{+}_{\geq \vec{x}}$ is not satisfied for any vector $\vec{x} \in \mathbb{Q}^3$. Hence, $B \implies D$ is falsified.

We only need to consider these two cases: all other dashed implications are false as they would otherwise contradict the last two cases by transitivity. □

We see that the relationship between mean payoff and total payoff games breaks in multiple dimensions. Nonetheless, one may still hope for good properties for the latter, as one-dimension total payoff games are in $\mathsf{NP} \cap \mathsf{coNP}$ (Section 4.6). This hope, however, will not last long.

12.3.2 Undecidability

In contrast to mean payoff games, total payoff ones become undecidable in multiple dimensions.

Theorem 132. *Total payoff games are undecidable in any dimension $k \geq 5$.*

Proof. We use a reduction from two-dimensional robot games [NPR16], which were mentioned in Chapter 11. They are a restricted case of configuration reachability vector games, recently proved to be already undecidable. Using the formalism of Chapter 11, they are expressible as follows: $\mathscr{V} = (\mathscr{L} = \{\ell_0, \ell_1\}, A, \mathscr{L}_{\text{Eve}} = \{\ell_0\}, \mathscr{L}_{\text{Adam}} = \{\ell_1\})$ and $A \subseteq \mathscr{L} \times [-M, M]^2 \times \mathscr{L}$ for some $M \in \mathbb{N}$. The game starts in configuration $\ell_0(x_0, y_0)$ for some $x_0, y_0 \in \mathbb{Z}$ and the goal of Eve is to reach configuration $\ell_0(0,0)$: solving such a game is undecidable.

The reduction is as follows. Given a robot game $\mathscr{V}$, we build a five-dimension total payoff game $\mathscr{G}$ such that Eve wins in $\mathscr{G}$ if and only if she wins in $\mathscr{V}$. Let $\mathscr{G} = (\mathscr{A}, \mathtt{TotalPayoff}^{+}_{\geq \vec{0}})$ (we will discuss the lim-inf case later), where arena $\mathscr{A}$ has vertices $V = V_{\text{Eve}} \uplus V_{\text{Adam}}$ with $V_{\text{Eve}} = \{v_{\mathsf{init}}, v_0, v_{\mathsf{stop}}\}$ and $V_{\text{Adam}} = \{v_1\}$, and E is built as follows:

- if $(\ell_i, (a,b), \ell_j) \in A$, then $(v_i, (a, -a, b, -b, 0), v_j) \in E$,
- $(v_0, (0,0,0,0,1), v_{\mathsf{stop}}) \in E$ and $(v_{\mathsf{stop}}, (0,0,0,0,0), v_{\text{stop}}) \in E$,
- $(v_{\mathsf{init}}, (x_0, -x_0, y_0, -y_0, -1), v_0) \in E$ (where (x_0, y_0) is the initial credit in $\mathscr{V}$).

The initial vertex is v_{init}. Intuitively, dimensions 1 and 2 (resp. 3 and 4) encode the value of the first counter (resp. second counter) and its opposite at all times. The initial credit is encoded thanks to the initial edge, then the game is played as in the vector game, with the exception that Eve may branch from v_0 to the absorbing vertex v_{stop}, which has a zero self loop. The role of the last dimension is to force Eve to branch eventually (if she aims to win).

We proceed to prove the correctness of the reduction. First, let $\sigma_{\mathscr{G}}$ be a winning strategy of Eve in $\mathscr{G}$. We claim that Eve can also win in $\mathscr{V}$. Any play π consistent with $\sigma_{\mathscr{G}}$ necessarily ends in v_{stop}: otherwise its lim-sup total payoff on the last dimension would be -1 (as the sum always stays at -1). Due to the branching edge and the self loop having weight zero in all first four dimensions, we also have that the current sum on these dimensions must be non-negative when branching, otherwise the total payoff objective would be falsified. By construction of $\mathscr{A}$, the only way to achieve this is to have a sum exactly equal to zero in all first four dimensions (as dimensions 1 and 2 are opposite at all times and so are 3 and 4). Finally, observe that obtaining a partial sum of $(0,0,0,0,-1)$ in v_0 is equivalent to reaching configuration $\ell_0(0,0)$ in $\mathscr{V}$. Hence, we can easily build a strategy $\sigma_{\mathscr{V}}$ in $\mathscr{V}$ that mimics $\sigma_{\mathscr{G}}$ in order to win

the robot game. This strategy $\sigma_{\mathscr{V}}$ could in general use arbitrary memory (since we start with an arbitrary winning strategy $\sigma_{\mathscr{G}}$) while formally robot games as defined in [NPR16] only allow strategies to look at the current configuration. Still, from $\sigma_{\mathscr{V}}$, one can easily build a corresponding strategy that meets this restriction ($\mathscr{V}$ being a configuration reachability game, there is no reason to choose different actions in two visits of the same configuration). Hence, if Eve wins in $\mathscr{G}$, she also wins in $\mathscr{V}$.

For the other direction, from a winning strategy $\sigma_{\mathscr{V}}$ in $\mathscr{V}$, we can similarly define a strategy $\sigma_{\mathscr{G}}$ that mimics it in $\mathscr{G}$ to reach v_0 with partial sum $(0,0,0,0,-1)$, and at that point, branches to v_{stop}. Such a strategy ensures reaching the absorbing vertex with a total payoff of zero in all dimensions, hence is winning in $\mathscr{G}$.

Thus, the reduction holds for lim-sup total payoff. Observe that the exact same reasoning holds for the lim-inf variant. Indeed, the last dimension is always -1 outside of v_{stop}, hence any play not entering v_{stop} also has its lim-inf below zero in this dimension. Furthermore, once v_{stop} is entered, the sum in all dimensions stays constant, hence the limit exists and both variants coincide. □

An almost identical reduction can be used for *shortest path* games.

Theorem 133. *Shortest path games are undecidable in any dimension $k \geq 4$.*

Remark 26. *For the sake of consistency, we use the convention established in Section 4.5: the shortest path payoff takes the opposite of the sum of weights along a path and Eve aims to maximise it (which is equivalent to minimising this sum in the natural interpretation of the shortest path objective). Hence, paths not reaching the target are assigned payoff $-\infty$.*

Proof. The proof is almost identical to the last one. We use objective $\texttt{ShortestPath}_{\geq\vec{0}}$ with target edge $(v_{\mathsf{stop}},(0,0,0,0),v_{\mathsf{stop}})$ and drop the last dimension in arena $\mathscr{A}$: it is now unnecessary as the shortest path objective by definition will force Eve to branch to v_{stop}, as otherwise the value of the play would be $-\infty$ in all dimensions. The rest of the reasoning is the same as before. □

Remark 27. *The decidability of total payoff games with $k \in \{2,3,4\}$ dimensions and shortest path games with $k \in \{2,3\}$ dimensions remains an open question. Furthermore, our undecidability results crucially rely on weights being in $\mathbb{Z}$: they do not hold when we restrict weights to $\mathbb{N}$.*

Memory

Let us go back to the game used in Item 1 in the proof of Lemma 142: we have seen that for any threshold $\vec{x} \in \mathbb{Q}^2$, Eve has an infinite-memory strategy that is winning for $\texttt{TotalPayoff}^{+}_{\geq\vec{x}}$. In other words, she can ensure an *arbitrarily high* total payoff with infinite memory. Yet, it is easy to check that there exists no finite-memory strategy of Eve that can achieve a finite threshold vector in the very same game: alternating would still be needed, but the negative amount to compensate grows boundlessly with each alternation, thus no amount of finite memory can ensure to go above the threshold infinitely often. This simple game highlights a huge gap between finite and infinite memory: with finite memory, the total payoff on at least one dimension is $-\infty$; with

infinite memory, the total payoff in both dimensions may be as high as Eve wants. This further highlights the untameable behaviour of multidimensional total payoff games.

Wrap-up

Multiple dimensions are a curse for total payoff and shortest path games as both become undecidable. This is in stark contrast to mean payoff and energy games, which remain tractable, as seen in Section 12.2. The bottom line is that most of the equivalences, relationships, and well-known behaviours of one-dimension games simply fall apart when lifting them to multiple dimensions.

12.4 Beyond worst-case synthesis

We now turn to a completely different meaning of *multi-objective*. Let us take a few steps back. Throughout this book, we have studied two types of interaction between players: rational, antagonistic interaction between Eve and Adam; and stochastic interaction with a random player. Consider the quantitative settings of Chapter 4 and Chapter 5. In the zero-sum two-player games of the former, Adam is seen as a *purely antagonistic adversary*, so the goal of Eve is to ensure strict *worst-case guarantees*, i.e., a minimal performance level against all possible strategies of Adam. In the MDPs of the latter, Eve interacts with randomness (through actions or random vertices) and she wants to ensure a good *expected value* for the considered payoff.

For most objectives, these two paradigms yield elegant and simple solutions: e.g., memoryless strategies suffice for both games and MDPs with a mean payoff objective. Nevertheless, the corresponding strategies have clear weaknesses: strategies that are good for the worst-case may exhibit suboptimal behaviours in probable situations while strategies that are good for the expected value may be terrible in some unlikely but possible situations. A natural question, of theoretical and practical interest, is to build — *synthesize* — strategies that combine both paradigms: strategies that both ensure (a) some worst-case threshold no matter how the adversary behaves (i.e., against any arbitrary strategy) and (b) a good expectation against the expected behaviour of the adversary (given as a stochastic model). We call this task *beyond worst-case synthesis*.

The goal of this section is to illustrate the complexity of beyond worst-case synthesis and how it requires fine-tuned interaction between the worst-case and average-case aspects. To that end, we focus on a specific case: the synthesis of *finite-memory strategies* for beyond worst-case *mean payoff* objectives. Due to the highly technical nature of this approach, we will not present all its details, but rather paint in broad strokes its cornerstones. We hope to give the reader sufficient intuition and understanding to develop a clear view of the challenges arising from rich behavioural models, and some of the techniques that come to the rescue.

12.4.1 The decision problem

Our goal is to mix the games of Chapter 4 and the MDPs of Chapter 5, so we need to go back and forth between these models.

Two-player game. As before, we start with an arena $\mathscr{A} = (G = (V,E), V_{\text{Eve}}, V_{\text{Adam}})$, where the vertices are split between Eve's and Adam's. This arena represents the antagonistic interaction between Eve and Adam, so we consider a worst-case constraint on the corresponding game. We study a single mean payoff function, so our colouring is $\mathfrak{c}\colon E \to \mathbb{Z}$. Let $\alpha \in \mathbb{Q}$ be the worst-case threshold: we are looking for a strategy of Eve that is winning for objective $\texttt{MeanPayoff}^{-}_{>\alpha}$. Two things to note: first, we consider the lim-inf variant w.l.o.g. as we focus on *finite-memory* strategies (recall Proposition 11); second, we use a strict inequality as it will ease the formulation of the upcoming results.

Markov decision process. To make the connection with MDPs, we fix a finite-memory randomised strategy for Adam in the arena $\mathscr{A}$, τ^{st}. Recall that a randomised strategy is a function $\text{Paths}(G) \to \mathscr{D}(E)$, where $\mathscr{D}(E)$ denotes the set of all probability distributions over E. As usual, we may build $\mathscr{A}_{\tau^{\mathsf{st}}}$, the product of the arena $\mathscr{A}$ with the memory structure of τ^{st}, restricted to the choices made by τ^{st}. Since τ^{st} is assumed to be stochastic, what we obtain is not a one-player game for Eve, but an MDP.

To understand this relationship, it is easier to consider the alternative — and equivalent — formalism of MDPs, based on random vertices (as used for stochastic games in Chapter 6). Assume for instance that τ^{st} is a randomised memoryless strategy, i.e., a function $V_{\text{Adam}} \to \mathscr{D}(E)$. Then, the MDP $\mathscr{A}_{\tau^{\mathsf{st}}}$ is immediately obtained by replacing each Adam's vertex v by a random vertex such that $\delta(v) = \tau^{\mathsf{st}}(v)$, i.e., the probabilistic transition function uses the same probability distributions as Adam's strategy. Formally, we build the MDP $\mathscr{P} = \mathscr{A}_{\tau^{\mathsf{st}}} = (G, V_{\text{Eve}}, V_{\text{Rand}} = V_{\text{Adam}}, \delta = \tau^{\mathsf{st}})$.

In contrast to Chapter 6, we explicitly allow the transition function to assign probability zero to some edges of the underlying graph G, i.e., the support of $\delta(v)$ in some vertex $v \in V_{\text{Rand}}$ might not include all edges $e \in E$ such that $\text{In}(e) = v$. This is important as far as modelling is concerned, as in our context, transition functions will be defined according to a stochastic model for Adam, and we cannot reasonably assume that such a model always involves all the possible actions of Adam. Consequently, given the MDP $\mathscr{P}$, we define the subset of edges $E_\delta = \{e \in E \mid \text{In}(e) \in V_{\text{Rand}} \implies \delta(\text{In}(e))(e) > 0\}$, representing all edges that either start in a vertex of Eve, or are chosen with non-zero probability by the transition function δ. Edges in $E \setminus E_\delta$ will only matter in the two-player game interpretation, whereas all MDP-related concepts, such as end-components, are defined with regard to edges in E_δ exclusively.

Beyond worst-case problem. Let us sum up the situation: we have a two-player arena $\mathscr{A}$ with a mean payoff objective $\texttt{MeanPayoff}^{-}_{>\alpha}$ and a finite-memory stochastic model for Adam yielding the MDP $\mathscr{A}_{\tau^{\mathsf{st}}}$. Now, let $\beta \in \mathbb{Q}$ be the expected value threshold we want to ensure in the MDP (i.e., on average against the stochastic model of Adam).

Problem 19 (Beyond worst-case mean payoff problem)**.**

INPUT: *An arena $\mathscr{A}$, a finite-memory stochastic model τ^{st}, an initial vertex v_0, two thresholds $\alpha, \beta \in \mathbb{Q}$*

OUTPUT: *Does Eve have a finite-memory strategy σ such that σ is winning for objective $\texttt{MeanPayoff}^{-}_{>\alpha}$ from v_0 in $\mathscr{A}$ and $\mathbb{E}^{\sigma}_{\mathscr{A}_{\tau^{\mathsf{st}}}, v_0}[\texttt{MeanPayoff}^{-}] > \beta$?*

We assume $\beta > \alpha$, otherwise the problem trivially reduces to the classical worst-case analysis: if all plays consistent with σ have mean payoff greater than $\alpha \geq \beta$ then the expected value is also greater than α — and thus greater than β — regardless of the stochastic model.

12.4.2 The approach in a nutshell

We present our solution to the beyond worst-case (BWC) problem in Algorithm 12.3. We give an intuitive sketch of its functioning in the following, and illustrate it on a toy example.

Inputs and outputs

The algorithm takes as input: an arena $\mathscr{A}^{\mathsf{in}}$ and its (integer) colouring $\mathfrak{c}^{\mathsf{in}}$, a finite-memory stochastic model of Adam τ^{in}, a worst-case threshold α^{in}, an expected value threshold β^{in}, and an initial vertex v_0^{in}. Its output is YES if and only if there exists a finite-memory strategy of Eve satisfying the BWC problem.

The output as described in Algorithm 12.3 is Boolean: the algorithm answers whether a satisfying strategy exists or not, but does not explicitly construct it (to avoid tedious formalization within the pseudocode). Nevertheless, we sketch the synthesis process in the following and we highlight the role of each step of the algorithm in the construction of a winning strategy, as producing a witness winning strategy is a straightforward by-product of the process we apply to decide satisfaction of the BWC problem.

Preprocessing

The first part of the algorithm is dedicated to the preprocessing of the arena $\mathscr{A}^{\mathsf{in}}$ and the stochastic model τ^{in} given as inputs in order to apply the second part of the algorithm on a modified arena $\mathscr{A}$ and stochastic model τ^{st}, simpler to manipulate. We show in the following that the answer to the BWC problem on the modified arena is YES if and only if it is also YES on the input arena, and we present how a winning strategy of Eve in $\mathscr{A}$ can be transferred to a winning strategy in $\mathscr{A}^{\mathsf{in}}$.

The preprocessing is composed of four main steps. First, we modify the colouring function $\mathfrak{c}^{\mathsf{in}}$ in order to consider the equivalent BWC problem with thresholds $(0, \beta)$ instead of $(\alpha^{\mathsf{in}}, \beta^{\mathsf{in}})$. This classical trick is used to get rid of explicitly considering the worst-case threshold in the following, as it is equal to zero.

Second, observe that any strategy that is winning for the BWC problem must also be winning for the classical *worst-case problem*, as solved in the two-player games of Chapter 4. Such a strategy cannot allow visits of any vertex from which Eve cannot ensure winning against an antagonistic adversary because mean payoff is a prefix independent objective (hence it is not possible to 'win' it over the finite prefix up to such a vertex). Thus, we reduce our study to $\mathscr{A}^{\mathsf{w}}$, the subarena induced by Eve's worst-case winning vertices — which we compute in pseudo-polynomial time thanks to `SolveWorstCaseMeanPayoff`$(\mathscr{A}^{\mathsf{in}}, \mathfrak{c}^{\mathsf{p}})$ (implementing the algorithm of Section 4.3). Note that we use the modified colouring and that $\mathscr{A}^{\mathsf{w}}$ is a proper arena (same argument

Algorithm 12.3: Solver for the beyond worst-case mean payoff problem

Data: Arena $\mathscr{A}^{\mathsf{in}} = (G^{\mathsf{in}} = (V^{\mathsf{in}}, E^{\mathsf{in}}), V^{\mathsf{in}}_{\text{Eve}}, V^{\mathsf{in}}_{\text{Adam}})$, colouring $\mathfrak{c}^{\mathsf{in}} \colon E^{\mathsf{in}} \to \mathbb{Z}$, finite-memory stochastic model τ^{in} for Adam with memory structure $\mathscr{M}$ and initial memory state m_0, worst-case and expected value thresholds $\alpha^{\mathsf{in}} = a/b, \beta^{\mathsf{in}} \in \mathbb{Q}$, $\alpha^{\mathsf{in}} < \beta^{\mathsf{in}}$, initial vertex $v_0^{\mathsf{in}} \in V^{\mathsf{in}}$

Result: YES if and only if Eve has a finite-memory strategy σ for the BWC problem from v_0^{in} for thresholds pair $(\alpha^{\mathsf{in}}, \beta^{\mathsf{in}})$

`/* Preprocessing */`

if $\alpha^{\mathsf{in}} \neq 0$ **then**

  Modify the colouring: $\forall e \in E^{\mathsf{in}}$, $\mathfrak{c}^{\mathsf{p}}(e) \leftarrow b \cdot \mathfrak{c}^{\mathsf{in}}(e) - a$

  Consider the new thresholds pair $(0, \beta \leftarrow b \cdot \beta^{\mathsf{in}} - a)$

else

  $\mathfrak{c}^{\mathsf{p}} \leftarrow \mathfrak{c}^{\mathsf{in}}$

$V_{\mathsf{wc}} \leftarrow$ `SolveWorstCaseMeanPayoff`$(\mathscr{A}^{\mathsf{in}}, \mathfrak{c}^{\mathsf{p}})$

if $v_0^{\mathsf{in}} \notin V_{\mathsf{wc}}$ **then**

  return NO

else

  $\mathscr{A}^{\mathsf{w}} \leftarrow \mathscr{A}^{\mathsf{in}}[V_{\mathsf{wc}}]$ `/* Restriction of` $\mathscr{A}^{\mathsf{in}}$ `to` V_{wc} `*/`

  Let $\mathscr{A} \leftarrow \mathscr{A}^{\mathsf{w}} \times \mathscr{M} = (G = (V, E), V_{\text{Eve}}, V_{\text{Adam}})$ be the arena obtained by product with the memory structure of Adam's stochastic model τ^{in}

  Let $v_0 \leftarrow (v_0^{\mathsf{in}}, m_0)$ be the corresponding initial vertex in $\mathscr{A}$

  Let $\mathfrak{c}$ be the transcription of $\mathfrak{c}^{\mathsf{p}}$ in $\mathscr{A}$ such that $e = (v,m) \to (v',m')$ has colour $\mathfrak{c}(e) = c$ iff $v \xrightarrow{c} v'$ in $\mathscr{A}^{\mathsf{w}}$ according to $\mathfrak{c}^{\mathsf{p}}$

  Let τ^{st} be the memoryless transcription of τ^{in} on $\mathscr{A}$

  Let $\mathscr{P} \leftarrow \mathscr{A}_{\tau^{\mathsf{st}}} = (G, V_{\text{Eve}}, V_{\text{Rand}} = V_{\text{Adam}}, \delta = \tau^{\mathsf{st}})$ be the corresponding MDP `/* Random vertices formalism */`

`/* Main algorithm */`

Compute $\mathscr{U}_{\mathsf{w}}$ the set of *maximal winning end components* of $\mathscr{P}$

Modify the colouring:

$$\forall e \in E, \mathfrak{c}'(e) \leftarrow \begin{cases} \mathfrak{c}(e) \text{ if } \exists U \in \mathscr{U}_{\mathsf{w}} \text{ s.t. } \{\text{In}(e), \text{Out}(e)\} \subseteq U \\ 0 \text{ otherwise} \end{cases}$$

Compute the maximal expected value β^* from v_0 in $\mathscr{P}$ using $\mathfrak{c}'$

if $\beta^* > \beta$ **then**

  return YES

else

  return NO

as Remark 24). Obviously, if from the initial vertex v_0^{in}, Eve cannot win the worst-case problem, then the answer to the BWC problem is NO.

Third, we build arena $\mathscr{A}$, the product of $\mathscr{A}^{\mathsf{w}}$ and the memory structure of Adam's stochastic model τ^{in}. Intuitively, we expand the initial arena by integrating the memory

elements in the graph. Note that this does not modify the power of Adam in the two-player interpretation of the arena.

Fourth, the finite-memory stochastic model τ^{in} on $\mathscr{A}^{\mathsf{in}}$ clearly translates to a memoryless stochastic model τ^{st} on $\mathscr{A}$. This will help us obtain elegant proofs for the second part of the algorithm.

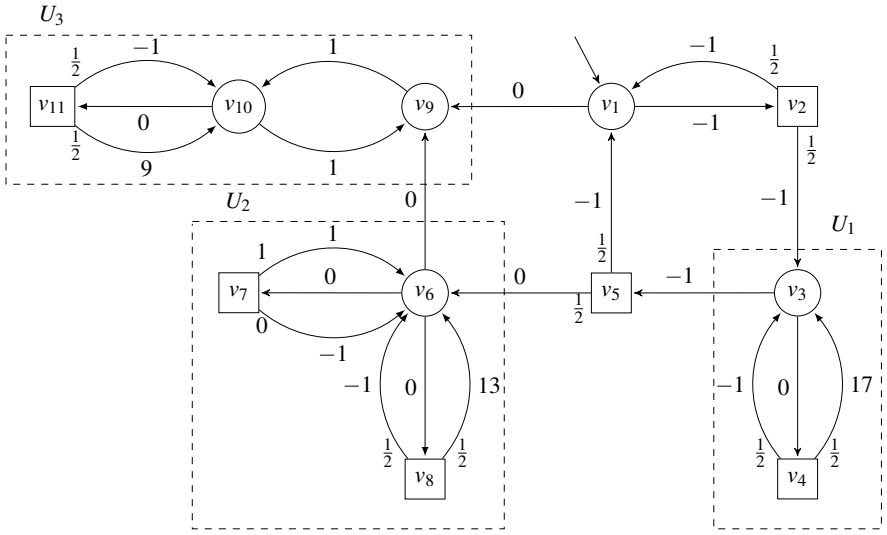


Figure 12.3: Beyond worst-case mean payoff problem: U_2 and U_3 are maximal winning end components, U_1 is losing.

Example 19. *In order to illustrate several notions and strategies, we will consider the arena depicted in Figure 12.3 throughout our presentation. The stochastic model of Adam is memoryless and is described by the probabilities written close to the start of outgoing edges. The colouring (weights) is written besides them.*

We consider the BWC problem with the worst-case threshold $\alpha = 0$. Observe that this arena satisfies the assumptions guaranteed at the end of the preprocessing part of the algorithm. That is, the worst-case threshold is zero, a worst-case winning strategy of Eve exists in all vertices (e.g., the memoryless strategy choosing edges (v_1, v_9), (v_3, v_5), (v_6, v_9), (v_9, v_{10}) and (v_{10}, v_9) in their respective starting vertices), and the stochastic model is memoryless, as explained above.

Analysis of end components

The second part hence operates on an arena $\mathscr{A}$ such that from all vertices, Eve has a strategy to achieve a strictly positive mean payoff value (recall that $\alpha = 0$). We consider the MDP $\mathscr{P} = \mathscr{A}_{\tau^{\mathsf{st}}}$ and notice that the underlying graphs of $\mathscr{A}$ and $\mathscr{P}$ are the same thanks to τ^{st} being memoryless. The following steps rely on the analysis of *end components* (ECs) in the MDP, i.e., strongly connected subgraphs in which Eve can ensure to stay when playing against Adam's stochastic model (Definition 14).

The motivation to the analysis of ECs is the following. It is well-known that under any arbitrary strategy σ of Eve in $\mathscr{P}$, the probability that vertices visited infinitely often along a play constitute an EC is one (Lemma 69). Recall that the mean payoff is prefix independent, therefore the value of any play only depends on those colours that are seen infinitely often. Hence, the expected mean payoff $\mathbb{E}^{\sigma}_{\mathscr{P},v_0}[\texttt{MeanPayoff}^{-}]$ depends *uniquely* on the value obtained in the ECs. Inside an EC, we can compute the maximal expected value that can be achieved by Eve, and this value is the same in all vertices of the EC, as established in Theorem 63.

Consequently, in order to satisfy the expected value requirement, an acceptable strategy for the BWC problem has to favor reaching ECs with a sufficient expectation, but under the constraint that it should also ensure satisfaction of the worst-case requirement. As we show in the following, this constraint implies that some ECs with high expected values may still need to be avoided because they do not permit to guarantee the worst-case requirement. This is the cornerstone of the classification of ECs that follows.

Classification of end components

Let $\mathscr{E} \subseteq 2^V$ denote the set of all ECs in $\mathscr{P}$. Notice that by definition, only edges in E_δ, as defined earlier, are involved to determine which sets of vertices form an EC in $\mathscr{P}$. As such, for any EC $U \in \mathscr{E}$, there may exist edges from $E \setminus E_\delta$ starting in U, such that Adam can force leaving U when using an arbitrary strategy in $\mathscr{A}$. Still these edges will never be used by the stochastic model τ^{st}. This remark will be important to the definition of strategies of Eve that guarantee the worst-case requirement, as Eve needs to be able to react to the hypothetical use of such an edge. We will see that it is also the case *inside* an EC.

Now, we want to consider the ECs in which Eve can ensure that the worst-case requirement will be fulfilled (i.e., without having to leave the EC): we call them *winning* ECs (WECs). The others will need to be eventually avoided, hence will have zero impact on the expectation of a finite-memory strategy satisfying the BWC problem. So we call the latter *losing* ECs (LECs). The subtlety of this classification is that it involves considering the ECs both in the MDP $\mathscr{P}$, and in the arena $\mathscr{A}$.

Formally, let $U \in \mathscr{E}$ be an EC. It is *winning* if, in the subarena induced by U, from all vertices, Eve has a strategy to ensure a *strictly* positive mean payoff against any strategy of Adam *that only chooses edges which are assigned non-zero probability by* τ^{st}, or equivalently, edges in E_δ. This can be interpreted as looking at arena $\mathscr{A}_\delta$, which is the restriction of $\mathscr{A}$ to edges in E_δ.

We denote $\mathscr{W} \subseteq \mathscr{E}$ the set of such ECs. Non-winning ECs are *losing*: in those, whatever the strategy of Eve played against the stochastic model τ^{st} (or any strategy with the same support), there exists at least one play for which the mean payoff is not strictly positive (even if its probability is zero, its mere existence is not acceptable for the worst-case requirement).

Example 20. *Note that an EC is winning if Eve has a worst-case winning strategy from all vertices. This point is important as it may well be the case that winning strategies exist in a strict subset of vertices of the EC. This does not contradict the definition of*

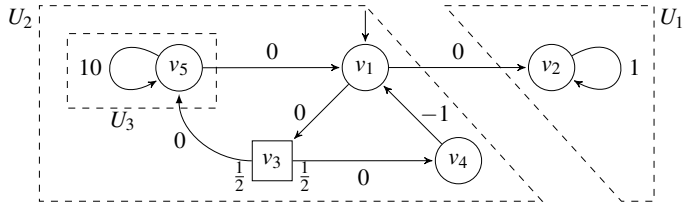


Figure 12.4: End component U_2 is losing. The set of maximal winning end components is $\mathscr{U}_{\mathsf{w}} = \mathscr{W} = \{U_1, U_3\}$.

ECs as strongly connected subgraphs, as the latter only guarantees that every vertex can be reached with probability one, and not necessarily surely. Hence one cannot call upon the prefix independence of the mean payoff to extend the existence of a winning strategy to all vertices.

Such a situation can be observed on the arena of Figure 12.4, where the EC U_2 is losing (because from v_1, the play $(v_1v_3v_4)^\omega$ can be forced by Adam, yielding mean payoff $-1/3 \leq 0$), while its sub-EC U_3 is winning. From v_1, Eve can ensure to reach U_3 almost-surely, but not surely, which is critical in this case.

Maximal winning end components

Based on these definitions, observe that Algorithm 12.3 does not actually compute the set $\mathscr{W}$ containing all WECs, but rather the set $\mathscr{U}_{\mathsf{w}} \subseteq \mathscr{W}$, defined as $\mathscr{U}_{\mathsf{w}} = \{U \in \mathscr{W} \mid \forall U' \in \mathscr{W}, U \subseteq U' \implies U = U'\}$, i.e., the set of *maximal* WECs (MWECs).

The intuition on *why we can* restrict our study to this subset is as follows. If an EC $U_1 \in \mathscr{W}$ is included in another EC $U_2 \in \mathscr{W}$, i.e., $U_1 \subseteq U_2$, we have that the maximal expected value achievable in U_2 is at least equal to the one achievable in U_1. Indeed, Eve can reach U_1 with probability one (by virtue of U_2 being an EC and $U_1 \subseteq U_2$) and stay in it forever with probability one (by virtue of U_1 being an EC): hence the expectation of such a strategy would be equal to what can be obtained in U_1 thanks to the prefix independence of the mean payoff. This property implies that it is sufficient to consider MWECs in our computations.

As for *why we do it*, observe that the complexity gain is critical. The number of WECs can be as large as $|\mathscr{W}| \leq |\mathscr{E}| \leq 2^{|V|}$, that is, exponential in the size of the input. Yet, the number of MWECs is bounded by $|\mathscr{U}_{\mathsf{w}}| \leq |V|$ as they are disjoint by definition: for any two WECs with a non-empty intersection, their union also constitutes an EC, and is still winning because Eve can essentially stick to the EC of her choice.

The computation of the set $\mathscr{U}_{\mathsf{w}}$ is executed by a recursive subalgorithm calling polynomially-many times an oracle solving the worst-case problem (e.g., following the pseudo-polynomial-time algorithm of Section 4.3). Roughly sketched, this algorithm computes the maximal EC decomposition of an MDP (in polynomial time by Theorem 66), then checks for each EC U in the decomposition (their number is polynomial) if U is winning or not, which requires a call to an oracle solving the worst-case threshold problem on the corresponding subgame. If U is losing, it may still be the case that a sub-EC $U' \subsetneq U$ is winning. Therefore we recurse on the MDP reduced to U, where

vertices from which Adam can win in U have been removed (they are a no-go for Eve). Hence the stack of calls is also at most polynomial.

Lemma 143. *The set $\mathscr{U}_{\mathsf{w}}$ of MWECs can be computed in pseudo-polynomial time, and deciding if a set of vertices $U \subseteq V$ belongs to $\mathscr{U}_{\mathsf{w}}$ is in $\mathsf{NP} \cap \mathsf{coNP}$.*

The complexity follows from Theorem 39 and $\mathsf{P}^{\mathsf{NP} \cap \mathsf{coNP}} = \mathsf{NP} \cap \mathsf{coNP}$ [Bra79].

Example 21. *Consider the running example in Figure 12.3. Note that vertices v_1, v_2 and v_5 do not belong to any EC: given any strategy of Eve in $\mathscr{P}$, with probability one, any consistent play will only visit these vertices a finite number of times (Lemma 69). The set of MWECs is $\mathscr{U}_{\mathsf{w}} = \{U_2, U_3\}$. Obviously, these ECs are disjoint. The set of WECs is larger, $\mathscr{W} = \mathscr{U}_{\mathsf{w}} \cup \{\{v_9, v_{10}\}, \{v_6, v_7\}\}$.*

End component U_1 is losing: in the subarena $\mathscr{A}_\delta[U_1]$, Adam's strategy consisting in always picking the -1 edge guarantees a negative mean payoff. Note that this edge is present in E_δ as it is assigned probability $1/2$ by the stochastic model τ^{st}. Here, we witness why it is important to base our definition of WECs on $\mathscr{A}_\delta$ rather than $\mathscr{A}$. Indeed, in $\mathscr{A}[U_2]$, it is also possible for Adam to guarantee a negative mean payoff by always choosing edges with weight -1. However, to achieve this, Adam has to pick edges that are not in E_δ: this will never happen against the stochastic model and as such, this can be watched by Eve to see if Adam uses an arbitrary antagonistic strategy, and dealt with. If Adam conforms to E_δ, i.e., if he plays in $\mathscr{A}_\delta$, he has to pick the edge of weight 1 in v_7 and Eve has a worst-case winning strategy consisting in always choosing to go in v_7. This EC is thus classified as winning. Note that for U_3, in both subarenas $\mathscr{A}[U_3]$ and $\mathscr{A}_\delta[U_3]$, Eve can guarantee a strictly positive mean payoff by playing $(v_9\, v_{10})^\omega$: even arbitrary strategies of Adam cannot endanger Eve in this case.

Lastly, consider the arena depicted in Figure 12.4. While U_2 is a strict superset of U_3, the former is losing whereas the latter is winning, as explained above. Hence, the set $\mathscr{U}_{\mathsf{w}}$ is equal to $\{U_1, U_3\}$.

Ensure reaching winning end components

As discussed, under any arbitrary strategy of Eve, vertices visited infinitely often form an EC with probability one (Lemma 69). Now, if we take a *finite-memory* strategy that *satisfies* the BWC problem, we can refine this result and state that they form a *winning* EC with probability one. Equivalently, let $\mathtt{Inf}(\pi)$ denote the set of vertices visited infinitely often along a play π: we have that the probability that a play π is such that $\mathtt{Inf}(\pi) = U$ for some $U \in \mathscr{E} \setminus \mathscr{W}$ is zero. The equality is crucial. It may be the case, with non-zero probability, that $\mathtt{Inf}(\pi) = U' \subsetneq U$, for some $U' \in \mathscr{W}$, and $U \in \mathscr{E} \setminus \mathscr{W}$ (hence the recursive algorithm to compute $\mathscr{U}_{\mathsf{w}}$). It is clear that Eve should not visit all the vertices of a LEC forever, as then she would not be able to guarantee the worst-case threshold inside the corresponding subarena.[3]

Lemma 144. *For any initial vertex v_0 and finite-memory strategy σ that satisfies the BWC problem, it holds that $\mathbb{P}^{\sigma}_{\mathscr{P}, v_0}[\{\pi \mid \mathtt{Inf}(\pi) \in \mathscr{W}\}] = 1$.*

[3] This is no longer true if Eve may use infinite memory: there may still be some incentive to stay in a LEC. But this goes beyond the scope of our overview.

We denote $V_{\mathsf{neg}} = V \setminus \bigcup_{U \in \mathscr{U}_{\mathsf{w}}} U$ the set of vertices that, with probability one, are only seen a finite number of times when a (finite-memory) BWC satisfying strategy is played, and call them *negligible* vertices.

Our ultimate goal here is to modify the colouring of $\mathscr{P}$ from $\mathfrak{c}$ to $\mathfrak{c}'$, such that a classical optimal strategy for the expected value problem (Theorem 70) using this new colouring $\mathfrak{c}'$ will naturally avoid LECs and prescribe which WECs are the most interesting to reach for a BWC strategy on the initial arena $\mathscr{A}$ and MDP $\mathscr{P}$ with colouring $\mathfrak{c}$. For the sake of readability, let us simply use $\mathscr{P}$ and $\mathscr{P}'$ to refer to MDP $\mathscr{P}$ with respective colourings $\mathfrak{c}$ and $\mathfrak{c}'$.

Observe that the expected value obtained in $\mathscr{P}$ by any BWC satisfying strategy of Eve only depends on the weights of edges involved in WECs, or equivalently, in MWECs (as the set of plays that are not eventually trapped in them has measure zero). Consequently, we define colouring $\mathfrak{c}'$ as follows: we keep the weights unchanged in edges that belong to some $U \in \mathscr{U}_{\mathsf{w}}$, and we put them to zero everywhere else, i.e., on any edge involving a negligible vertex. Weight zero is taken because it is lower than the expectation granted by WECs, which is *strictly* greater than zero by definition (as $\alpha = 0$).

Example 22. *Consider U_1 in Figure 12.3. This EC is losing as argued before. The optimal expectation achievable in $\mathscr{P}[U_1]$ by Eve is 4: this is higher than what is achievable in both U_2 and U_3. Note that there exists no WEC included in U_1. By Chapter 5, we know that, from v_1, any strategy of Eve will see its expectation bounded by the maximum between the optimal expectations of the ECs U_1, U_2 and U_3. Our previous arguments further refine this bound by restricting it to the maximum between the expectations of U_2 and U_3. Indeed, Eve cannot benefit from the expected value of U_1 while using finite memory, as being trapped in U_1 induces the existence of plays losing for the worst-case constraint. Hence there is no point in playing inside U_1 and Eve may as well cross it directly and try to maximise its expectation using the WECs, U_2 and U_3. The set of negligible vertices in $\mathscr{P}$ is $V_{\mathsf{neg}} = V \setminus (U_2 \cup U_3) = \{v_1, v_2, v_3, v_4, v_5\}$. We depict $\mathscr{P}'$ in Figure 12.5.*

In the arena depicted in Figure 12.4, we already observed that $\mathscr{E} = \{U_1, U_2, U_3\}$ and $\mathscr{W} = \mathscr{U}_{\mathsf{w}} = \{U_1, U_3\}$. Consider the negligible vertex $v_1 \in V_{\mathsf{neg}} = U_2 \setminus U_3$. A finite-memory strategy of Eve may only take the edge (v_1, v_3) finitely often in order to ensure the worst-case requirement. If Eve were to play this edge repeatedly, the losing play $(v_1 v_3 v_4)^\omega$ would exist (while of probability zero). Therefore, Eve can only ensure that U_3 is reached with a probability arbitrarily close to one, and not equal to one, because at some point, she has to switch to edge (v_1, v_2) (after a bounded time since Eve uses a finite-memory strategy).

Reach the highest valued winning end components

We compute the maximal expected mean payoff β^* that can be achieved by Eve in $\mathscr{P}'$, from v_0. This computation takes polynomial time and memoryless strategies suffice to achieve the maximal value, as established in Theorem 70. As seen before, such a strategy reaches an EC of $\mathscr{P}'$ with probability one. Basically, we build a strategy that favours reaching ECs with high associated expectations in $\mathscr{P}'$.

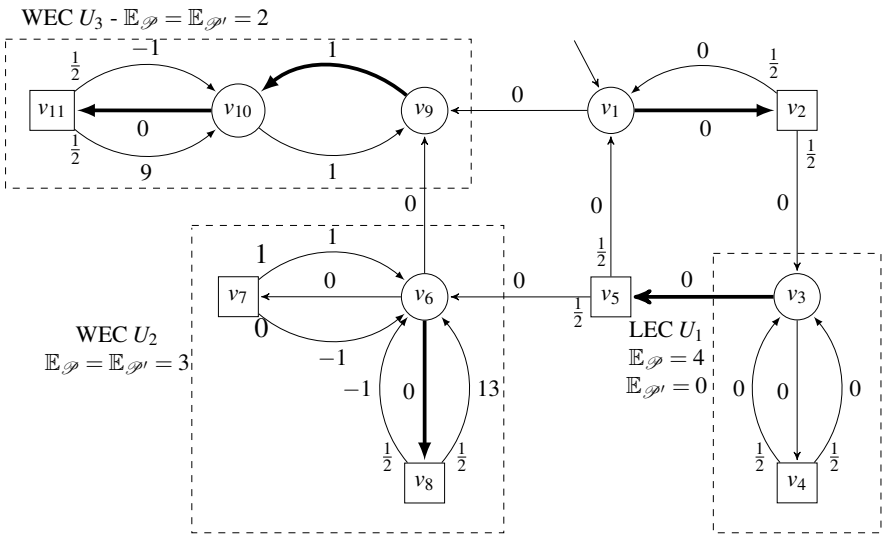


Figure 12.5: Putting all weights outside MWECs to zero naturally drives the optimal expectation strategy in $\mathscr{P}'$, depicted by the thick edges, toward the highest valued MWECs. ECs are annotated with their corresponding optimal expectations in the original MDP $\mathscr{P}$ and the modified MDP $\mathscr{P}'$.

We argue that the ECs reached with probability one by this strategy are necessarily WECs in $\mathscr{P}$. Clearly, if a WEC is reachable instead of a losing one, it will be favoured because of the weights definition in $\mathscr{P}'$ (expectation is strictly higher in WECs). Thus it remains to check if the set of WECs is reachable with probability one from any vertex in V. That is the case because of the preprocessing: we know that all vertices are winning for the worst-case requirement. Clearly, from any vertex in $A = V \setminus \bigcup_{U \in \mathscr{E}} U$, Eve cannot ensure to stay in A (otherwise it would form an EC) and thus must be able to win the worst-case requirement from reached ECs. Now for any vertex in $B = \bigcup_{U \in \mathscr{E}} U \setminus \bigcup_{U \in \mathscr{U}_{\mathsf{w}}} U$, i.e., vertices in LECs and not in any winning sub-EC, Eve cannot win the worst-case by staying in B, by definition of LEC. Since we know Eve can ensure the worst-case by hypothesis, it is clear that she must be able to reach $C = \bigcup_{U \in \mathscr{U}_{\mathsf{w}}} U$ from any vertex in B, as claimed.

Inside winning end components

Based on that, we know that WECs of $\mathscr{P}$ will be reached with probability one when maximizing the expected value in $\mathscr{P}'$. Let us first consider what we can say about such ECs if we assume that $E_\delta = E$, i.e., if the stochastic model τ^{st} maps all possible edges to non-zero probabilities. We establish a finite-memory *combined strategy* σ^{cmb} of Eve that ensures (i) worst-case satisfaction while yielding (ii) an expected value ε-close to the maximal expectation inside the EC.

For two well-chosen parameters $K, L \in \mathbb{N}$, it is informally defined as follows: in

phase (a), play a memoryless expected value optimal strategy σ^{e} for K steps and memorise $\mathtt{Sum} \in \mathbb{Z}$, the sum of weights along these steps; in phase (b), if $\mathtt{Sum} > 0$, go to (a), otherwise play a memoryless worst-case optimal strategy σ^{wc} for L steps, then go to (a). In (a), Eve tries to increase her expectation and approach the optimal one, while in (b), she compensates, if needed, losses that occurred in (a).

The two memoryless strategies exist on the subarena induced by the EC: by definition of ECs, based on E_δ, the stochastic model of Adam will never be able to force leaving the EC against the combined strategy.

A key result to our approach is the existence of values for K and L such that (i) and (ii) are verified. We see plays as sequences of periods, each starting with phase (a).

First, for any K, it is possible to define $L(K)$ such that any period composed of phases $(a)+(b)$ ensures a mean payoff at least $1/(K+L) > 0$. Periods containing only phase (a) trivially induce a mean payoff at least $1/K$ as they are not followed by phase (b). Both rely on the weights being integers. As the length of any period is bounded by $(K+L)$, the inequality remains strict for the mean payoff of any play, granting (i).

Now, consider parameter K. Clearly, when $K \to \infty$, the expectation over phase (a) tends to the optimal one. Nevertheless, phases (b) also contribute to the overall expectation of the combined strategy, and (in general) lower it so that it is strictly less than the optimal for any $K, L \in \mathbb{N}$. Hence to prove (ii), we not only need that the probability of playing phase (b) decreases when K increases, but also that it decreases faster than the increase of L, needed to ensure (i), so that overall, the contribution of phases (b) tends to zero when $K \to \infty$. This is indeed the case and is proved using (rather technical) results bounding the probability of observing a mean payoff significantly (more than some ε) different than the optimal expectation along a phase (a) of length $K \in \mathbb{N}$: this probability decreases exponentially when K increases, while L only needs to be polynomial in K.

Theorem 134. *Let $U \in \mathscr{W}$ be a WEC, τ^{st} be such that $E_\delta = E$, $v_0 \in U$ be the initial vertex, and let $\beta^* \in \mathbb{Q}$ be the maximal expected value achievable by Eve in EC U. Then, for all $\varepsilon > 0$, there exists a finite-memory strategy of Eve that satisfies the BWC problem for the thresholds pair $(0, \beta^* - \varepsilon)$.*

Example 23. *Consider the subarena $\mathscr{A}_\delta[U_3] = \mathscr{A}[U_3]$ from Figure 12.3 and the initial vertex v_{10}. Clearly, the worst-case requirement can be satisfied, that is why the EC is classified as winning. Always choosing to go to v_9 when in v_{10} is an optimal memoryless worst-case strategy σ^{wc} that guarantees a mean payoff $\alpha^* = 1$. Its expectation is $\mathbb{E}^{\sigma^{\mathsf{wc}}}_{(\mathscr{A}[U_3])_{\tau^{\mathsf{st}}}, v_{10}}[\mathtt{MeanPayoff}^-] = 1$. On the other hand, the strategy σ^{e} that always selects the edge going to v_{11} is optimal regarding the expected value criterion: it induces expectation $\beta^* = \big(0 + \big(1/2 \cdot 9 + 1/2 \cdot (-1)\big)\big)/2 = 2$ against the stochastic model τ^{st}. However, it can only guarantee a mean payoff of value $-1/2$ in the worst-case.*

By the reasoning above, we know that it is possible to find finite-memory strategies satisfying the BWC problem for any thresholds pair $(0, 2-\varepsilon)$, $\varepsilon > 0$. In particular, consider the thresholds pair $(0, 3/2)$. We build a combined strategy σ^{cmb} as sketched before. Let $K = L = 2$: the strategy plays the edge (v_{10}, v_{11}) once, then if the edge of value 9 has been chosen by Adam, it chooses (v_{10}, v_{11}) again; otherwise it chooses the edge (v_{10}, v_9) once and then resumes choosing (v_{10}, v_{11}). This

strategy satisfies the BWC problem. In the worst-case, Adam always chooses the -1 *edge, but each time he does so, the* -1 *is followed by two* $+1$ *thanks to the cycle* $v_{10}v_9v_{10}$*. Strategy* σ^{cmb} *hence guarantees a mean payoff equal to* $(0-1+1+1)/4 = 1/4 > 0$ *in the worst-case. For the expected value requirement, we can build the induced Markov chain* $(\mathscr{A}[U_3])_{\sigma^{\mathsf{cmb}},\tau^{\mathsf{st}}}$ *(Figure 12.6) and check that its expectation is* $\mathbb{E}^{\sigma^{\mathsf{cmb}}}_{(\mathscr{A}[U_3])_{\tau^{\mathsf{st}}},v_{10}}[\texttt{MeanPayoff}^-] = 5/3 > 3/2$ *(Chapter 4).*

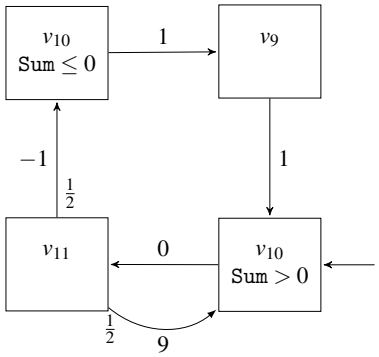


Figure 12.6: Markov chain induced by the combined strategy σ^{cmb} and the stochastic model τ^{st} over the WEC U_3 of $\mathscr{A}$.

Remark 28. *Memoryless strategies do not suffice for the BWC problem, even with randomisation. Indeed, the edge* (v_{10}, v_{11}) *cannot be assigned a non-zero probability as it would endanger the worst-case requirement (since the play* $(v_{10}v_{11})^\omega$ *cycling on the edge of weight* -1 *would exist and have a negative mean payoff). Hence, the only acceptable memoryless strategy is* σ^{wc}*, which has only an expectation of* 1.

Now, consider what happens if $E_\delta \subsetneq E$. Then, if Adam uses an arbitrary strategy, he can take edges of probability zero, i.e., in $E \setminus E_\delta$, either staying in the EC, or leaving it. In both cases, this must be taken into account in order to satisfy the worst-case constraint as it may involve dangerous weights (recall that zero-probability edges are not considered when an EC is classified as winning or not). Fortunately, if this were to occur, Eve could switch to a worst-case winning memoryless strategy σ^{sec}, which exists in all vertices thanks to the preprocessing, to preserve the worst-case requirement. Regarding the expected value, this has no impact as it occurs with probability zero against τ^{st}. The strategy to follow in WECs hence adds this reaction procedure to the combined strategy: we call it the *witness-and-secure strategy* σ^{wns}.

Theorem 135. *Let* $U \in \mathscr{W}$ *be a WEC,* $v_0 \in U$ *be the initial vertex, and* $\beta^* \in \mathbb{Q}$ *be the maximal expected value achievable by Eve in EC* U*. Then, for all* $\varepsilon > 0$*, there exists a finite-memory strategy of Eve that satisfies the BWC problem for the thresholds pair* $(0, \beta^* - \varepsilon)$.

Example 24. *Consider the WEC* U_2 *in Figure 12.3 and the initial vertex* $v_6 \in U_2$*. Eve can ensure a strictly positive mean payoff in the subarena* $\mathscr{A}_\delta[U_2]$*, but not in* $\mathscr{A}[U_2]$.

Indeed, it is easy to see that by always choosing the -1 *edges (which requires an edge* $(v_7, v_6) \in E \setminus E_\delta$*), Adam can ensure a negative mean payoff whatever the strategy of Eve. However, there exists a strategy that ensures the worst-case constraint, i.e., that yields a strictly positive mean payoff against any strategy of Adam, by leaving the EC. Let* σ^{sec} *be the memoryless strategy that takes the edge* (v_6, v_9) *and then cycle on* $(v_{10}v_9)^\omega$ *forever: it guarantees a mean payoff of* $1 > 0$*.*

For a moment, consider the EC U_2 *in* $\mathscr{A}_\delta$*. Graphically, it means that the* -1 *edge from* v_7 *to* v_6 *disappears. In the subarena* $\mathscr{A}_\delta[U_2]$*, there are two particular memoryless strategies. The optimal worst-case strategy* σ^{wc} *guarantees a mean payoff of* $1/2 > 0$ *by choosing to go to* v_7*. The optimal expectation strategy* σ^{e} *yields an expected mean payoff of* 3 *by choosing to go to* v_8 *(naturally this strategy yields the same expectation whether we consider edges in* E_δ *or in* E*). Based on them, we build the combined strategy* σ^{cmb} *of Eve as defined earlier and by Theorem 134, for any* $\varepsilon > 0$*, there are values of* K *and* L *such that it satisfies the BWC problem for thresholds* $(0, 3-\varepsilon)$ *in* $\mathscr{A}_\delta[U_2]$*. For instance, for* $K = L = 2$*, we have* $\mathbb{E}^{\sigma^{\mathsf{cmb}}}_{(\mathscr{A}[U_2])_{\tau^{\mathsf{st}}}, v_6}[\texttt{MeanPayoff}^-] = 13/6$*.*

We construct the witness-and-secure strategy σ^{wns} *based on* σ^{cmb} *and* σ^{sec} *as described above. In this case, that means playing as* σ^{cmb} *until the* -1 *edge from* v_7 *to* v_6 *is taken by Adam. This strategy ensures a worst-case mean payoff equal to* $1 > 0$ *thanks to* σ^{sec} *and yields expectation* $\mathbb{E}^{\sigma^{\mathsf{wns}}}_{(\mathscr{A}[U_2])_{\tau^{\mathsf{st}}}, v_6}[\texttt{MeanPayoff}^-] = 13/6$ *for* $K = L = 2$*.*

Finally, notice that securing the mean payoff by switching to σ^{sec} *is needed to satisfy the worst-case requirement if Adam plays in* $E \setminus E_\delta$*. Also, observe that it is still necessary to alternate according to* σ^{cmb} *in* $\mathscr{A}_\delta[U_2]$ *and that playing* σ^{e} *is not sufficient to ensure the worst-case (because Eve has to deal with the* -1 *edge from* v_8 *to* v_6 *that remains in* E_δ*).*

Global strategy synthesis

In summary, (a) LECs should be avoided and will be by a strategy that optimises the expectation on the MDP $\mathscr{P}'$; (b) in WECs, Eve can obtain (ε-closely) the expectation of the EC *and* ensure the worst-case threshold.

Hence, we finally compare the value β^* computed by Algorithm 12.3 with the expected value threshold β: (i) if it is strictly higher, we conclude that there exists a finite-memory strategy satisfying the BWC problem, and (ii) if it is not, we conclude that there does not exist such a strategy.

To prove (i), we establish a finite-memory strategy in $\mathscr{A}$, called *global strategy* σ^{glb}, of Eve that ensures a strictly positive mean payoff against an antagonistic adversary, and ensures an expected mean payoff ε-close to β^* (hence, strictly greater than β) against the stochastic adversary modeled by τ^{st} (i.e., in $\mathscr{P}$). The intuition is as follows. We play the memoryless optimal strategy of $\mathscr{P}'$ for a sufficiently long time, defined by a parameter $N \in \mathbb{N}$, in order to be with probability close to one in a WEC (the convergence is exponential by results on absorption times in Markov chains). Then, if we are inside a WEC, we switch to the corresponding witness-and-secure strategy (there is a different one for each MWEC) which, as sketched in the previous paragraph, ensures the worst-case and the expectation thresholds. If we are not yet in a WEC, then we

switch to a worst-case winning strategy, which always exists thanks to the preprocessing. Thus the mean payoff of plays that do not reach WECs is strictly positive. Since in WECs we are ε-close to the maximal expected value of the EC, we can conclude that it is possible to play the optimal expectation strategy of $\mathscr{P}'$ for sufficiently long to obtain an overall expected value which is arbitrarily close to β^*, and still guarantee the worst-case threshold in all consistent plays.

To prove (ii), it suffices to understand that only ECs have an impact on the expectation, and that LECs cannot be used forever without endangering the worst-case requirement.

Note that given a winning strategy on $\mathscr{A}$, it is possible to build a corresponding winning strategy on $\mathscr{A}^{\mathsf{in}}$ by reintegrating the memory states of τ^{in} in the memory structure of the winning strategy of Eve. Hence Algorithm 12.3 is correct and complete.

Theorem 136. *If Algorithm 12.3 answers* YES*, then there exist values of the parameters such that the pure finite-memory global strategy σ^{glb} satisfies the BWC mean payoff problem. In the opposite case, there exists no finite-memory strategy that satisfies the BWC mean payoff problem.*

Example 25. *Consider the arena in Figure 12.3 and the associated MDP $\mathscr{P}$. Following Chapter 5, analysis of the maximal ECs U_1, U_2 and U_3 reveals that the maximal expected mean payoff achievable in $\mathscr{P}$ is* 4*. It is for instance obtained by the memoryless strategy that chooses to go to v_2 from v_1 and to v_4 from v_3. Observe that playing in U_1 forever is needed to achieve this expectation. By Lemma 144, this should not be allowed as the worst-case cannot be ensured if it is. Indeed, Adam can produce worst-case losing plays by playing the -1 edge. Clearly, the maximal expected value that Eve can ensure while guaranteeing the worst-case requirement is thus bounded by the maximal expectation in $\mathscr{P}'$, i.e., by* 3*, as depicted in Figure 12.5. Let σ^{e} denote an optimal memoryless expectation strategy in $\mathscr{P}'$ that tries to enter U_2 by playing (v_1, v_2) and (v_3, v_5), and then plays edge (v_6, v_8) forever (thick edges in Figure 12.5).*

Observe that Algorithm 12.3 answers YES *for any thresholds pair $(0, \beta)$ such that $\beta < 3$. For the sake of illustration, we construct the global strategy σ^{glb} as presented earlier, with $N = 6$ and $K = L = 2$. For the first six steps, it behaves exactly as σ^{e}. Note that after the six steps, the probability of being in U_2 is $1/4 + 1/8 = 3/8$. Then, σ^{glb} switches to another strategy depending on the current vertex (σ^{wns} or σ^{wc}) and sticks to this strategy forever. In particular, if the current vertex belongs to U_2, it switches to σ^{wns} for $K = L = 2$, which guarantees the worst-case threshold and induces an expectation of $13/6$. By definition of σ^{glb}, if the current vertex after six steps is not in U_2, then σ^{glb} switches to σ^{wc} which guarantees a mean payoff of* 1 *by reaching vertex v_9 and then playing $(v_9 v_{10})^\omega$. Overall, the expected mean payoff of σ^{glb} against τ^{st} is*

$$\mathbb{E}^{\sigma^{\mathsf{glb}}}_{\mathscr{A}_{\tau^{\mathsf{st}}, v_1}}[\mathtt{MeanPayoff}^-] \geq \frac{3}{8} \cdot \frac{13}{6} + \frac{5}{8} \cdot 1 = \frac{23}{16}.$$

Notice that by taking N, K and L large enough, it is possible to satisfy the BWC problem for any $\beta < 3$ with the strategy σ^{glb}. Also, observe that the WEC U_2 is crucial to achieve expectations strictly greater than 2*, which is the upper bound when limited to EC U_3. For instance, $N = 25$ and $K = L = 2$ implies an expectation strictly greater than* 2 *for the global strategy.*

Lastly, note that in general, the maximal expectation achievable in $\mathscr{P}'$ (and thus in $\mathscr{P}$ when limited to strategies that respect the worst-case requirement) may depend on a combination of ECs instead of a unique one. This is transparent through the solving of the expected value problem in the MDP $\mathscr{P}'$.

Complexity bounds

The input size of the algorithm depends on the size of the arena, the size of the memory structure for the stochastic model, and the encodings of probabilities, weights and thresholds. We can prove that all computing steps require (deterministic) polynomial time except for calls to an algorithm solving the worst-case threshold problem, which is in $\mathsf{NP} \cap \mathsf{coNP}$ and not known to be in P (Theorem 39). Hence, the overall complexity of the BWC problem is in $\mathsf{NP} \cap \mathsf{coNP}$ (using $\mathsf{P}^{\mathsf{NP} \cap \mathsf{coNP}} = \mathsf{NP} \cap \mathsf{coNP}$ [Bra79]) and may collapse to P if the worst-case problem were to be proved in P.

The BWC problem is at least as difficult as the worst-case problem thanks to a trivial polynomial-time reduction from the latter to the former. Thus, membership to $\mathsf{NP} \cap \mathsf{coNP}$ can be seen as optimal regarding our current knowledge of mean payoff games.

Theorem 137. *The BWC mean payoff problem is in* $\mathsf{NP} \cap \mathsf{coNP}$ *and at least as hard as solving mean payoff games. Moreover, pseudo-polynomial-memory strategies may be necessary for Eve and are always sufficient.*

The memory bounds follow from the (involved) probability results used to determine the values of parameters K, L and N in the aforementioned strategies: such parameters need to be polynomial in the size of the arena but also in the probabilities, weights and thresholds.

Thanks to the pseudo-polynomial-time algorithm of Section 4.3 for mean payoff games, we obtain the following corollary.

Corollary 23 (Beyond worst-case for mean payoff games)**.** *Algorithm 12.3 solves the BWC mean payoff problem in pseudo-polynomial time.*

Wrap-up

As witnessed by our long overview, solving the beyond worst-case problem requires much more involved techniques than solving the two individual problems, worst-case and expected value, separately. Complexity-wise, it is fortunate that the problem stays in $\mathsf{NP} \cap \mathsf{coNP}$, and is no more complex that simple mean payoff games. The multi-objective nature of the problem still incurs a cost with regard to strategies: whereas memoryless strategies suffice both in mean payoff games and mean payoff MDPs, we here need pseudo-polynomial memory. Finally, note that Eve does not need to use randomness: pure strategies still suffice.

12.5 Percentile queries

We close this chapter by a quick detour to multidimensional MDPs. When considering single-dimension MDPs with payoffs, as in Chapter 5, there are two different (yet both natural) settings that arise depending on how one decides to aggregate play values through the probability measure. Let $\mathscr{P}$ be an MDP, v an initial vertex, and f the payoff function. In the first setting, Eve's goal is to optimise the expected value of the payoff function, that is, to find a strategy σ that maximises $\mathbb{E}^{\sigma}_{\mathscr{P},v}[f]$. In the second setting, we set a performance threshold to achieve for the payoff function, say $\alpha \in \mathbb{Q}$, essentially creating the qualitative objective $f_{\geq\alpha}$, and Eve aims to maximise the probability to achieve this objective, i.e., she is looking for a strategy σ that maximises $\mathbb{P}^{\sigma}_{\mathscr{P},v}[f_{\geq\alpha}]$. The concept of percentile query extends the latter problem to multidimensional payoffs.

From now on, assume we have an MDP $\mathscr{P}$ with a multidimensional colouring function $\mathfrak{c}\colon E \to \mathbb{Z}^k$. Whether $\mathscr{P}$ uses actions as in Chapter 5 or random vertices as in Chapter 6 does not matter for our discussion — both are equivalent modulo slight modifications of the MDP. Recall that we denote by f_i, $1 \leq i \leq k$, the projection of f to its i-th component.

Problem 20 (Percentile query problem)**.**

INPUT: *An MDP $\mathscr{P}$, an initial vertex v_0, a payoff function f, $q \geq 1$ the number of percentile constraints in the query, q dimensions $l_i \in \{1,\dots,k\}$, q value thresholds $\alpha_i \in \mathbb{Q}$, q probability thresholds $\mu_i \in \mathbb{Q} \cap [0,1]$.*

OUTPUT: *Does Eve have a strategy σ such that σ is winning for the conjunction of q constraints, called percentile query,*

$$\mathscr{Q} = \bigwedge_{i=1}^{q} \mathbb{P}^{\sigma}_{\mathscr{P},v_0}[f_{l_i} \geq \alpha_i] \geq \mu_i ?$$

As usual, we also want to synthesize such a strategy σ if one exists. Note that this percentile query framework permits to express rich properties, as each of the q constraint can use a different dimension, value threshold and probability threshold. It is also possible to have different constraints related to the same dimension, for example to enforce different value thresholds for different quantiles.

The percentile query problem has been studied for a variety of payoff functions. Our aim here is not to give an exhaustive account of the corresponding results and techniques, but to highlight some new phenomena that arise in this setting, in comparison to what we have seen up to now.

12.5.1 An additional leap in complexity

The expressiveness of percentile queries asks for richer classes of strategies, even in very simple MDPs.

Example 26. *Consider the single-player game depicted in Figure 12.7. Note that it is an MDP (using only Dirac distributions). Consider the payoff function* $\texttt{MeanPayoff}^{-}$.

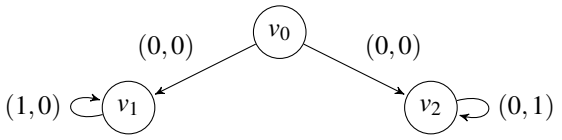


Figure 12.7: Randomised strategies are needed to achieve any percentile query of the form $\mathbb{P}^{\sigma}_{\mathscr{P},v_0}[\mathtt{MeanPayoff}_1^- \geq 1] \geq \mu_1 \wedge \mathbb{P}^{\sigma}_{\mathscr{P},v_0}[\mathtt{MeanPayoff}_2^- \geq 1] \geq \mu_2$ with $\mu_1, \mu_2 > 0$.

It is clear that due to its prefix independence, any play π ending in v_1 (resp. v_2) will yield $\mathtt{MeanPayoff}^-(\pi) = (1,0)$ *(resp.* $(0,1)$*). Hence to achieve a percentile query*

$$\mathscr{Q} = \bigwedge_{i=1}^{2} \mathbb{P}^{\sigma}_{\mathscr{P},v_0}[\mathtt{MeanPayoff}_i^- \geq 1] \geq \mu_i,$$

Eve must go toward v_i with probability at least μ_i. If both probability thresholds are non-zero, then this is only achievable by using randomness within Eve's strategy.

Example 26 uses the mean payoff for the sake of consistency with the previous sections, but observe that it can be emulated with virtually all objectives considered in this book. In particular, using reachability with two target sets (corresponding to the edges $(1,0)$ and $(0,1)$) is sufficient.

While *pure* strategies were used in most chapters[4] of this book, *randomised* strategies have already been considered in specific settings, such as in Chapter 7, usually to break some kind of symmetry and/or make one's strategy hard to predict. In our setting of percentile queries, we are still dealing with relatively simple models of games: we consider turn-based, perfect information games. Yet, the need for randomness arises from the expressiveness of our class of objectives, which in general require careful balance between different stochastic options.

Example 27. *Let us have another look at the MDP in Figure 12.7. Consider now that the probability thresholds μ_1 and μ_2 in query $\mathscr{Q}$ are not fixed a priori. Instead, we are interested in the set of vectors (μ_1, μ_2) that Eve can achieve. In particular, we want to determine the Pareto frontier*[5] *and the corresponding Pareto-optimal strategies. What is interesting here is that, in our simple example, there is already an infinite, non-countable, number of Pareto vectors. Indeed, Eve can ensure any vector (μ_1, μ_2) such that $\mu_1, \mu_2 \geq 0$ and $\mu_1 + \mu_2 = 1$ by simply taking the edge leading to v_i with probability μ_i.*

Although the Pareto frontier consists of an infinite number of points in Example 27, it can be represented in a finite way, as it is essentially obtained through linear combinations of two extreme vectors: $(1,0)$ and $(0,1)$. Interestingly, these two vectors correspond to what can be achieved with *pure* strategies, and their convex hull yields the Pareto frontier. So, the Pareto frontier can be represented as a convex polytope whose vertex representation is given by the vectors achievable by pure strategies. This

[4] Without loss of generality as they suffice in the respective contexts of these chapters.

[5] One can easily adapt Definition 30 to this context.

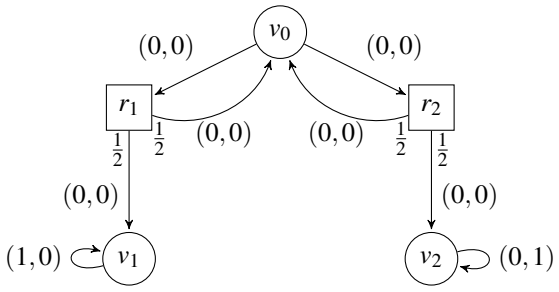


Figure 12.8: When restricted to pure strategies, there are still infinitely many Pareto vectors for the query $\mathbb{P}^{\sigma}_{\mathscr{D},v_0}[\texttt{MeanPayoff}_1^- \geq 1] \geq \mu_1 \wedge \mathbb{P}^{\sigma}_{\mathscr{D},v_0}[\texttt{MeanPayoff}_2^- \geq 1] \geq \mu_2$. For example, $(1,0)$, $(0,1)$, $(1/2,1/2)$, $(3/4,1/4)$, $(1/4,3/4)\ldots$

is not merely an artefact resulting from the simplicity of our example; similar phenomena occur in many settings mixing MDPs and multiple objectives.

While the continuous aspect of the Pareto frontier stems from the possibility to use randomness in strategies, complex Pareto frontiers are also to be expected when restricting Eve to pure strategies.

Example 28. *Consider the MDP in Figure 12.8. It uses the random vertices formalism as in Section 12.4. This MDP is a slight adaptation of the one in Figure 12.7, the crux being that when Eve tries to go to v_i now, she has to cross r_i, which has probability $1/2$ to send her back to v_0. In the long run, it does not really matter, as Eve will almost-surely end up in v_1 or v_2 (Chapter 5). And if Eve is allowed to use randomness, we obtain the same Pareto frontier as in the previous example. Yet, these random vertices serve a purpose.*

When restricted to pure strategies, Eve cannot use the inherent randomness of her strategy to achieve any given vector (μ_1,μ_2), as she could in Example 27. Nonetheless, by using memory, Eve is still able to achieve infinitely many Pareto vectors. For example, by first choosing to go to r_1, then r_2 (if the play comes back to v_0), then r_1 again (and then every time the play goes back to v_0), Eve will achieve vector $(3/4,1/4)$.

It is relatively easy to see that infinitely many Pareto vectors can be generated with memory and no randomness in Example 28; for example all vectors of the form $(1-p,p)$ where $p = 1/2^n$ for $n \in \mathbb{N}$. Still, all such vectors could already be generated via randomised memoryless strategies as sketched before.

In particular, all vectors achievable by using memory and no randomness are of the form (μ_1,μ_2), with $\mu_1,\mu_2 \geq 0$ and $\mu_1+\mu_2 = 1$ — but not all such vectors can be achieved that way! Hence, by restricting the use of randomness, we have effectively created 'gaps' in the Pareto frontier, and rendered its description much more difficult. In full generality, it is usually necessary to use both randomness and memory to satisfy percentile queries.

Proposition 12. *Pareto-optimal strategies for the percentile query problem may require randomness and memory (possibly infinite depending on the payoff function).*

12.5.2 Complexity overview

We close our discussion of percentile queries with an overview of their complexity for various payoffs studied in Chapter 2, Chapter 4, and Chapter 5. We sum up the situation in Table 12.1. Some of these results are quite technical to establish, so our goal here is only to highlight interesting elements with respect to everything that has been discussed in the previous chapters and in our own.

	Single-constraint	Single-dim. Multi-constraint	Multi-dim. Multi-constraint
`Reach`	P	$\mathrm{P}(\mathscr{P})\cdot\mathrm{E}(\mathscr{Q})$, PSPACE-h	—
$f \in \mathscr{F}$	P	P	$\mathrm{P}(\mathscr{P})\cdot\mathrm{E}(\mathscr{Q})$ PSPACE-h.
$\texttt{MeanPayoff}^{+}$	P	P	P
$\texttt{MeanPayoff}^{-}$	P	$\mathrm{P}(\mathscr{P})\cdot\mathrm{E}(\mathscr{Q})$	$\mathrm{P}(\mathscr{P})\cdot\mathrm{E}(\mathscr{Q})$
`ShortestPath`	$\mathrm{P}(\mathscr{P})\cdot\mathrm{P}_{\mathsf{ps}}(\mathscr{Q})$ PSPACE-h.	$\mathrm{P}(\mathscr{P})\cdot\mathrm{P}_{\mathsf{ps}}(\mathscr{Q})$ (one target) PSPACE-h.	$\mathrm{P}(\mathscr{P})\cdot\mathrm{E}(\mathscr{Q})$ PSPACE-h.
ε-gap `DiscountedPayoff`	$\mathrm{P}_{\mathsf{ps}}(\mathscr{P},\mathscr{Q},\varepsilon)$ NP-h.	$\mathrm{P}_{\mathsf{ps}}(\mathscr{P},\varepsilon)\cdot\mathrm{E}(\mathscr{Q})$ NP-h.	$\mathrm{P}_{\mathsf{ps}}(\mathscr{P},\varepsilon)\cdot\mathrm{E}(\mathscr{Q})$ PSPACE-h.

Table 12.1: Complexity of percentile query problems for various payoffs. Here $\mathscr{F} = \{\texttt{Inf},\texttt{Sup},\texttt{LimInf},\texttt{LimSup}\}$. Parameters $\mathscr{P}$ and $\mathscr{Q}$ respectively represent the size of the MDP, and the size of the query; $\mathrm{P}(x)$, $\mathrm{E}(x)$ and $\mathrm{P}_{\mathsf{ps}}(x)$ respectively denote polynomial, exponential and pseudo-polynomial time in parameter x. For the shortest path, only non-negative weights can be used, as otherwise the problem is undecidable.

Let us take a moment to appreciate Table 12.1. First, the payoffs present in the left column have all been discussed before; the only oddity being the notion of ε-gap attached to the discounted payoff. Its presence is merely technical: we do not know if percentile queries using the discounted payoff can be solved exactly (and it is linked to long-standing open questions), but a slight relaxation of the problem, called ε-gap problem, can be solved. Intuitively, this relaxation consists in allowing an ε-wide uncertainty area around the value thresholds (α_i) of the query. Second, some of the expressiveness of the queries is hidden in the table. For example, when using `Reach` or `ShortestPath`, one may consider different target sets in each constraint. Similarly, when using `DiscountedPayoff`, the discount factors may vary across constraints. Finally, when meaningful, the complexity is broken down into two parts, representing the relative dependency toward the size of the MDP, and the size of the query. The interest of this approach is that, in general and for practical applications, the model size is large whereas the query, encoding a specification, is comparatively much smaller. With that in mind, the polynomial dependency in the size of the MDP for most cases can be seen as good news.

Now, let us compare to what we know outside of percentile queries. Note that single-constraint queries correspond to the probability threshold problems in MDPs, studied in Chapter 5. We see that in most cases, the jump to multiple dimensions induces an exponential blow-up (in the number of constraints). If we compare to two-player (non-stochastic) games, as studied in Section 12.1, Section 12.2, and Sec-

tion 12.3, we see that the undecidability of shortest path objectives holds up if we replace the antagonistic player Adam by stochasticity. On the complexity side, the situation varies from payoff to payoff with, again, an interesting lack of symmetry between the two variants of mean payoff, in stark contrast to the single-dimension case.

Bibliographic references

As discussed in the introduction, the literature on multi-objective models is too vast to provide a full account here. We therefore limit ourselves to some directions particularly relevant to our focus.

Multidimensional games. Our presentation of mean payoff games is inspired by Velner et al. [VCD+15]. Brenguier and Raskin studied the Pareto frontiers of these games in [BR15]. While we considered *conjunctions* of mean payoff objectives, Velner proved that Boolean combinations lead to undecidability [Vel15].

Energy games were discussed — through the prism of vector games — in Chapter 11. The link between energy games and mean payoff games under finite memory was established in [VCD+15]. While triple-exponential bounds on memory for Eve's strategies could be derived from [BJK10], the first exponential upper bounds were proved by Chatterjee et al. [CRR14], also encompassing conjunctions with *parity* objectives. These bounds have since been refined [JLS15] but remain exponential; indeed, it is known that exponential memory is necessary for Eve [CRR14].

The undecidability of total payoff games was first established by Chatterjee et al. in [CDRR15] via reduction from the halting problem for two-counter machines: we provided here a new, simpler proof based on robot games [NPR16]. This undecidability result, along with the complexity barriers of mean payoff and total payoff games, motivated the introduction of (multidimensional) *window objectives*: conservative variants of mean payoff and total payoff objectives that benefit from increased tractability and permit to reason about time bounds [CDRR15]. They have since been studied in a variety of contexts: variants of parity objectives [BHR16a], Markov decision processes [BDOR20], timed games [MRS21], etc.

The undecidability of shortest path games is formally new, but the result was already established for MDPs by Randour et al. in [RRS17]. Here, we use the aforementioned new approach based on robot games.

Consumption games were studied by Brázdil et al. [BCKN12]: they have the flavor of energy games but are actually incomparable. In such games, only negative weights are used, and gaining energy can only be done through particular 'reload edges' that refill the energy up to a predefined capacity.

Parts of our presentation of multidimensional games are inspired by [Ran14].

Combinations of different objectives. We focused on multidimensional games obtained by conjunction of *identical* objectives. Conjunctions of *heterogeneous* objectives have been studied in numerous different contexts, including mean payoff parity games [CHJ05, DJL18], energy parity games [CD12, CRR14], average-energy games

with energy constraints [BMR$^+$18, BHM$^+$17], energy mean payoff games [BHRR19], or Boolean combinations of simple quantitative objectives [BHR16b]. While similarities often exist across these different settings, ad hoc techniques are still needed and complexity results may vary greatly. Developing a general framework encompassing large classes of multi-objective problems is still a vastly unexplored challenge. Some progress has been achieved recently with a focus on strategy complexity; we discuss it further below.

Beyond worst-case synthesis. Our presentation is mostly inspired by Bruyère et al. in [BFRR17], which introduced the BWC synthesis problem, and where all technical details can be found. As noted in [BFRR17], allowing large inequalities in the BWC problem may require infinite-memory strategies. The case of infinite-memory strategies was studied in [CR15] along with multidimensional BWC mean payoff problems.

BWC problems were studied for other objectives, such as shortest path [BFRR17] or parity [BRR17]; and on other related models (e.g., [BKN16, AKV16]). BWC principles have been implemented in the tool UPPAAL [DJL$^+$14]. Boolean combinations of objectives akin to the BWC problem have been considered in MDPs [BGR20].

The BWC framework aims to provide strategies that exhibit high performance in normal operating conditions while offering a high-level of resilience in extreme conditions. A kindred — but softer — approach is the study of strategies in MDPs that achieve a trade-off between the expectation and the variance over the outcomes, giving a statistical measure of the stability of the performance. Brázdil et al. have considered the mean payoff with this philosophy in [BCFK17].

Percentile queries. The framework of percentile queries was introduced by Randour et al. in [RRS17], where they studied a variety of payoffs: all results mentioned in this chapter are from [RRS17]. As mentioned in Section 12.5.2, the percentile query problem was established to be undecidable for the shortest path payoff when both positive and negative weights are allowed. The theory of percentile queries can be seen as a quantitative follow-up to the work of Etessami et al. on reachability objectives, in [EKVY08].

Several other problems have been considered on multidimensional MDPs. For example, in [BBC$^+$14], Brázdil et al. study two different problems based on the mean payoff. On the one hand, they consider the optimisation of the expected value vector. On the other hand, they show how to optimise the probability that the payoff vector is above a threshold vector. Observe that, in comparison to percentile queries, the latter problem asks for a bound on the joint probability of the thresholds, that is, the probability of satisfying all constraints simultaneously. In contrast, percentile queries bound the marginal probabilities separately, which may allow for more modeling flexibility. Another complementary approach was considered by Haase and Kiefer in [HK15]: whereas percentile queries allow for conjunctions between probability constraints on simple value inequalities, they consider only one probability constraint but allow for conjunctions of value constraints within this single probability constraint. Hence, both frameworks are incomparable.

Various frameworks mixing concepts from beyond worst-case synthesis and percentile queries have been developed in the recent years, both in exact, formal approaches (e.g., [BGMR18, CKK17]), and in reinforcement learning (e.g., [KPR18, CENR18, BCNV20]). Comparisons between percentile queries, beyond worst-case synthesis, and other rich behavioural models can be found in [RRS14, BCH$^+$16].

Pareto frontiers. As hinted throughout this chapter, the first step in understanding multi-objective settings is often to fix acceptable performance thresholds and to focus on the corresponding decision problem, asking if there exists a strategy to ensure these thresholds. Yet, to fully embrace the complexity of multi-objective frameworks, to be able to visualise the interplay between different (qualitative and quantitative) aspects and the corresponding trade-offs, we have to look at Pareto frontiers. This endeavour is generally difficult and one requires specific techniques to provide efficient approximations of Pareto frontiers — due to their complexity, exact computation is often out of reach.

We already mentioned [BR15], which deals with Pareto frontiers in multidimensional mean payoff games. A seminal approach in the case of MDPs was developed by Forejt et al. in [FKP12]. In a nutshell, it consists in obtaining successive approximations of the Pareto frontier by solving many one-dimension problems. Each of these is obtained by reinterpreting the original k-objective problem as a weighted sum of each individual objective. Having the weight assignment vary yields different one-dimension problems, but also permits to approximate the real Pareto frontier from a different angle, as each weight assignment morally corresponds to looking in a different direction within the k-dimension space. The crux is then to explore this space efficiently by a clever choice of weights at each iteration.

Complex strategies. The additional expressive power of multi-objective games and MDPs comes at a cost in terms of algorithmic complexity, but also with regard to the classes of strategies that are necessary to play (Pareto-)optimally. We have seen various examples in this chapter were Eve had to resort to strategies with (finite or infinite) memory and/or randomness. In a similar spirit to the characterisations of memoryless-determined objectives discussed in the early chapters of this book, recent research has striven to characterise the need for complex strategies in broad classes of multi-objective settings.

Recently, there has been a lot of progress on understanding the power of finite-memory strategies, both in games [BRO$^+$22, BRV23] and in MDPs [BORV21]. An overview by Bouyer et al. is given in [BRV22]. With a particular focus on multi-objective games, Le Roux et al. studied general conditions under which finite-memory determinacy can be maintained through combinations of finite-memory determined objectives [RPR18].

With regard to randomness — which we have seen to be necessary in the most general multi-objective settings, it is interesting to see that not all forms of randomness were created equal: when considering finite-memory (Section 1.5) randomised strategies, one can put randomness in different parts of the memory structure (initial state, update function, and/or 'next-action' function that exploits the memory to take a

decision), with differences in terms of resulting power [MR22]. This is in contrast to the celebrated Kuhn's theorem in classical game theory, which crucially relies on the availability of infinite memory.

Finally, as complex strategies are often prohibitive in practical applications, it is sometimes interesting to consider multi-objective problems where one looks for strategies of limited complexity. That is, even if the problem requires complex strategies for (Pareto-)optimality, one may be interested in how good simple strategies can be. For example, [DKQR20] develops techniques to explore the Pareto frontier of multi-objective MDPs under the constraint that only pure (i.e., without randomness) and limited-memory strategies should be considered. As seen in Example 28, such constraints often break the nice structure of (unrestricted) Pareto frontiers, which renders their exploration (and representation) more difficult.


4
6
PLAYER
MULTIPLAYER
GAMES
R

Chapter 13

Multiplayer Games

ROMAIN BRENGUIER, OCAN SANKUR

In two player games seen so far, players had objectives that are opposite to each other's, so we were able to define them giving only Eve's objective. Adam was seen as a purely adversarial player. Such games are called *zero-sum* games since, in a quantitative setting, the sum of the payoffs of the two players would sum up to zero in any outcome. However, the objectives of the players are not entirely conflicting in all games. In particular, in multiplayer games, that is, games with more than two players, the binary view of zero-sum games does not make sense; but there are also interesting examples of non-zero sum games with only two players (we will see one below). In this setting, winning strategies are no longer suitable to describe rational behaviours since the opponents should no longer be seen as purely adversarial. In fact, when the objectives of the players are not opposite, some cooperation becomes possible. Then, rather than assuming that opponents are purely adversarial, it is interesting to study the possible outcomes when they are simply *rational*, that is, follow the best strategy for their own objectives. The notion of equilibria we will study in this chapter aims at describing such rational behaviours.

If one is expecting for sure some specific strategies to be played by the opponents, then the most rational response is to choose the *best response*, that is, the strategy that is optimal for the player against the given strategies of the other players. Thus, if we assign strategies to players, and if the players are all aware of the strategies of the other players, then each player will be willing to change their strategy if theirs does not turn out to be a best response. Such a situation is seen as unstable and is undesirable in many applications of game theory. *Nash equilibrium* is defined simply as a stable situation in such a setting: a strategy profile in which the strategy of each player is a best response to the rest of the strategies. Thus, no player has any incentive to change their strategy.

We will see the formal definition of a Nash equilibrium in the next section. Let us first consider the following example.

The following Hawk-Dove game was first presented by the biologists Smith and

Table 13.1: The Hawk-Dove game. Each column corresponds to a strategy of P_1 and each line to a strategy of P_2.

	Hawk	Dove
Hawk	0 , 0	1 , 4
Dove	4 , 1	3 , 3

Price, and shown in Table 13.1. Here, two animals are fighting for resources and can choose to either act as a hawk or as a dove. If both player choose hawk they will have to fight for resources, and thus only get payoff 0. If only one chooses hawk, they get a high payoff of 4, because they get all the valuable resources for themselves, while the dove gets 1: they get plenty of resources but gets hunted. When they both choose dove, they both get a payoff of 3: they have to share resources but do not get hunted.

When a player chooses hawk then the best payoff for the opponent is obtained by choosing dove, so as to avoid fighting for resources. So, dove is the best response to hawk. Reciprocally, the best response to dove is to play hawk. There are two 'equilibria': (Hawk, Dove) and (Dove, Hawk), where no player has an interest in changing their strategy. Note that the highest payoff a player can ensure (against all adversary strategies) is only 1.

Nash showed the existence of such equilibria in normal-form games[1], which may require randomized strategies. This result revolutionized the field of economics, where it is used to analyze competitions between firms or government economic policies for example. Game theory and the concept of Nash equilibrium are now applied to diverse fields: in finance to analyse the evolution of market prices, in biology to understand the evolution of some species, in political sciences to explain public choices made by parties.

In this chapter, we will first study the computation of Nash equilibria in multiplayer concurrent games with ω-regular objectives. The algorithms we present here differ from those that were given for normal-form games since ours are infinite-duration with omega-regular objectives. We will then present extensions of this notion such as secure and robust equilibria. The second result we develop is the notion of admissibility: this is a different approach to the study of rational behaviours and consists in eliminating for each player irrational choices of strategies.

13.1 Nash Equilibria for games in normal form

The normal form games we consider differ from the matrix games of Chapter 7, in that each player has their own payoff. So for instance, when player 1 chooses column Hawk, and player 2 chooses row Dove, the payoff for player 1 is $f_{P_1}(\text{Hawk}, \text{Dove}) = 4$.

[1] normal-form games are also called matrix games, see Chapter 7.

Table 13.2: A game of medium access.

	Emit	Wait
Emit	-1, -1	2, 0
Wait	0 , 2	0, 0

Let us call a vector of strategies specifying a strategy for each player a *strategy profile*. In normal-form games, each cell of the table Δ corresponds to a strategy profile.

Definition 31. *A* Nash equilibrium *is a* stable *strategy profile in which strategy is a best response against the other strategies.*

Thus a Nash equilibrium is a stable situation in the sense that no player has an incentive in changing their strategy. Nash proved that when players are allowed to randomise among all their strategies, there always exists a Nash equilibrium.

Theorem 138 (Existence of Nash equilibria)**.** *In every normal-form game with a finite number of players, each having a finite number of pure strategies, there exists a randomised Nash equilibrium.*

Note that not all games contain pure Nash equilibria. For example, in the rock-paper-scissors game, the best response to rock is paper, to paper is scissors, and to scissors is rock, so none of these pure strategies can be an equilibrium.

For finding a pure Nash equilibrium in a normal-form game, there is a simple polynomial time algorithm. For each strategy profile, we look for each player whether they have a better response than their current strategy. If no player has a better response, the strategy profile is a Nash equilibrium, otherwise we move to the next one, and if none satisfies the condition then there is no equilibrium.

Example 29 (Medium Access Control)**.** *Consider a medium access control problem, where several users share access to a wireless channel. A communication over the channel is successful if there are no collisions, that is, if a single user is transmitting their message only. During each slot, each user chooses either to transmit or to idle. Intuitively, the number of packets transmitted without collision decreases with the number of users emitting in the same slot. Furthermore each attempt at transmitting has a cost. An example payoff for two players, is represented in Table Table 13.2.*

We encourage the reader to find the Nash equilibria of the above game.

The game described above corresponds to a single slot of this system. In a practical scenario, there would be a succession of slots and the payoff would be the sum of payoffs over all slots. Normal-form games are thus not sufficient to represent games with repetitions and to study the evolution of the behaviours as the game evolves.

One possibility to model repetition is to use *games in extensive form* which are games played on finite trees. However such games only model a fixed number of repetitions unlike infinite or arbitrary duration games as studied in this book. We thus study, in the rest of this chapter, algorithms for games played on graphs.

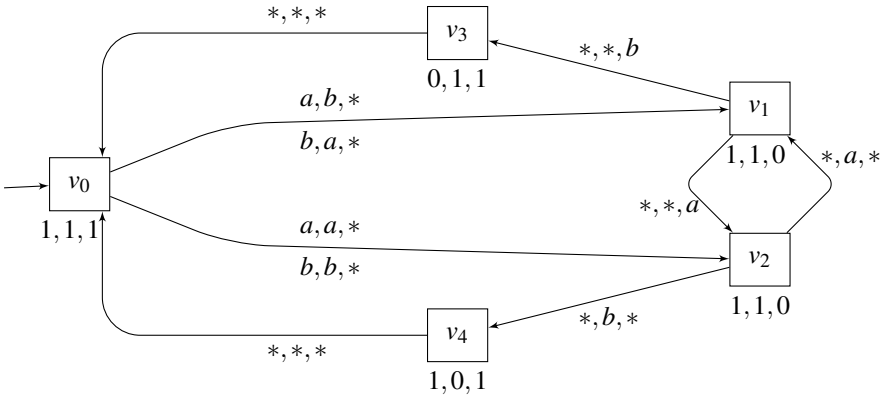


Figure 13.1: Example of a three-player concurrent arena. The symbol $*$ on edges can be replaced by either a or b.

13.1.1 Definitions

Definition 32. *A multiplayer* arena $\mathscr{A}$ *with k players is a tuple $\langle V, A, \Delta, (c_P)_{P\in\mathscr{P}}\rangle$, where:*

- *V is a finite set of vertices;*
- *$\mathscr{P} = \{1,2,\ldots,k\}$ is the set of players;*
- *A is a finite set of actions, a tuple $(a_P)_{P\in\mathscr{P}}$ containing one action a_P for each player A is called a* move*, thus A^k is the set of possible moves;*
- *$\Delta : V \times A^k \to V$ is the transition function which associates to a pair of vertices and moves the resulting state;*
- *$(c_P)_{P\in\mathscr{P}}$ is a tuple of colouring functions with $c_P : V \to C$ for each $P \in \mathscr{P}$.*

Example 30. *A simple three-player concurrent game is represented in Figure 13.1. Vertices are v_0, v_1, v_2, v_3 and v_4. Players are named P_1, P_2, P_3. The set of actions is $A = \{a,b\}$. The transition relation is given by the edges in the graph, for instance $\Delta(s_0,(a,b,a))$ is v_1. In our figures, $*$ represents an arbitrary action. The colouring function is represented below vertices as tuples ranging over players. For instance, a vertex labelled by $(1,1,0)$ assigns the first two players the colour 1, and the third player the colour 0, In particular, $c_{P_1}(v_2) = 1$, and $c_{P_3}(v_2) = 0$.*

A *history* of the multiplayer arena $\mathscr{A}$ is a finite sequence of states and moves ending with a state, i.e. a word in $(V \cdot A^{\mathscr{P}})^* \cdot V$. Note that unlike for two player games we include actions in the history, because knowing the source and target vertices does not mean you know which player chose which actions.

For a history π, we write π_i the i-th vertex of π, starting from 0, and $\mathsf{move}_i(\pi)$ its i-th move, thus $\pi = \pi_0 \cdot \mathsf{move}_0(\pi \cdot \pi_1 \cdots \mathsf{move}_{n-1}(\pi) \cdot \pi_n$, and with this notation

$\mathsf{move}_i(\pi)_P$ is the i-th action of player P in h. The length $|\pi|$ of such a history is $n+1$. We write last(π) the last vertex of h, i.e. $\pi_{|\pi|-1}$. A play ρ is an infinite sequence of vertices and moves, i.e. an element of $(V \cdot A^{\mathscr{P}})^\omega$.

Definition 33 (Strategy and coalition)**.** *A strategy is a function which associates an action to each history. We often write σ_A for a strategy of player P. A coalition $\mathscr{C}$ is a set of players in $\mathscr{P}$, and we write $-\mathscr{C}$ for the remaining players, that is $-\mathscr{C} = Agt \setminus \mathscr{C}$. Let $\mathscr{C}$ be a coalition, a strategy $\sigma_{\mathscr{C}}$ for $\mathscr{C}$ is a function which associates a strategy σ_P to each player $P \in \mathscr{C}$. Given a strategy $\sigma_{\mathscr{C}}$, when it is clear from the context, we simply write σ_P for $\sigma_{\mathscr{C}}(P)$.*

Definition 34 (Outcomes)**.** *A history π is compatible with the strategy $\sigma_{\mathscr{C}}$ for coalition $\mathscr{C}$ if, for all $k < |\pi| - 1$ and all $P \in \mathscr{C}$, we have $(\mathsf{move}_k(\pi))_P = \sigma_P(\pi \leq k)$, and $\Delta(\pi_k, \mathsf{move}_k(\pi)) = \pi_{k+1}$. A play ρ is compatible with the strategy $\sigma_{\mathscr{C}}$ if all its prefixes are. We write $\mathsf{Out}_{\mathscr{A}}(v_0, \sigma_{\mathscr{C}})$ for the set of plays in $\mathscr{A}$ that are compatible with strategy $\sigma_{\mathscr{C}}$ and have initial vertex v_0. Let $\mathsf{Out}_{\mathscr{A}}(\sigma_{\mathscr{C}})$ denote the union of $\mathsf{Out}_{\mathscr{A}}(v_0, \sigma_{\mathscr{C}})$ for all v_0, and $\mathsf{Out}_{\mathscr{A}}(v_0)$ the union of all $\mathsf{Out}_{\mathscr{A}}(v_0, \sigma_{\mathscr{C}})$. The subscript $\mathscr{A}$ can be omitted if it is clear from the context. These paths are called* outcomes *of $\sigma_{\mathscr{C}}$ from v_0.*

Note that when the coalition $\mathscr{C}$ is composed of all the players the outcome from a given state is unique.

Example 31. *Consider in the example of Figure 13.1, the following strategies:*

- *P_1 always plays a, i.e. $\sigma_{P_1}(\pi) = a$ for all histories π;*
- *P_2 plays a in v_0 if it is the first state and then always plays b, i.e. $\sigma_{P_2}(v_0) = a$ and $\sigma_{P_2}(\pi) = b$ for all $\pi \neq v_0$;*
- *P_3 always plays b, i.e. $\sigma_{P_3}(\pi) = b$.*

The outcome from v_0 in that case is

$$\begin{aligned} \mathsf{Out}(v_0, \sigma_{\{P_1,P_2,P_3\}}) &= v_0 \cdot (a,a,b) \cdot v_2 \cdot (a,b,b) \cdot v_4 \cdot (a,b,b) \cdot \\ &\quad (v_0 \cdot (a,b,b) \cdot v_1 \cdot (a,b,b) \cdot v_3 \cdot (a,b,b))^\omega \end{aligned}$$

Definition 35 (Multiplayer game)**.** *A* payoff *function associates a real number to each outcome. We will be mostly be interested in solving games with qualitative objectives, that is payoffs that take values* 0 *and* 1*. A* multiplayer game *$(\mathscr{A}, (f_P)_{P \in \mathscr{P}})$ is given by a multiplayer arena $\mathscr{A}$, an initial vertex v_0 and payoff function f_P for each player P. When f_P is qualitative we simply write Ω_P for the corresponding objective.*

13.1.2 The Nash equilibrium problem

In this section we will present an algorithm to compute Nash equilibria in multiplayer games. The problem we are interested in is to decide the existence of a Nash equilibrium in which the objectives of a given set of players are satisfied.

Problem 21 (Existence of a constrained Nash equilibrium)**.**

INPUT: *A multiplayer game* $(\mathscr{A}, (f_P)_{P\in\mathscr{P}})$*, payoff bounds* $(b_P)_{P\in\mathscr{P}}$*, and an initial vertex* v_0

OUTPUT: *Does there exist a Nash equilibrium* $\sigma_{\mathscr{P}}$ *such that for all* $P \in \mathscr{P}$*, we have* $f_P(\mathsf{Out}(v_0, \sigma_{\mathscr{P}})) \geq b_P$*?*

The algorithm is based on a reduction to zero-sum two-players games, which allows us to use algorithms presented in the previous chapters of this book. More precisely, we present the *deviator game*, which is a transformation of a concurrent multiplayer game into a turn-based zero-sum game, such that there are strong links between equilibria in the former and winning strategies in the latter. The proofs of this section are independent of the type of objectives we consider.

13.1.3 Deviators

A central notion we use is that of *deviators*. These are the players who have played different moves from those prescribed in a given profile, thus causing a deviation from the expected outcome. Formally, a deviator from move $a_{\mathscr{P}}$ to $a'_{\mathscr{P}}$ is a player $D \in \mathscr{P}$ such that $a_D \neq a'_D$. We denote the set of deviators by

$$\mathsf{Dev}(a_{\mathscr{P}}, a'_{\mathscr{P}}) = \{D \in \mathscr{P} \mid a_D \neq a'_D\}.$$

We extend the definition to pairs of histories and strategies by taking the union of deviator sets of each step along the history. Formally,

$$\mathsf{Dev}(\pi, \sigma_{\mathscr{P}}) = \bigcup_{0\leq i<|h|} \mathsf{Dev}(\mathsf{move}_i(\pi), \sigma_{\mathscr{P}}(\pi_{\leq i})).$$

For an infinite play ρ, we define $\mathsf{Dev}(\rho, \sigma_{\mathscr{P}}) = \bigcup_{i\in\mathbb{N}} \mathsf{Dev}(\mathsf{move}_i(\rho), \sigma_{\mathscr{P}}(\rho_{\leq i}))$. Intuitively, having chosen a strategy profile $\sigma_{\mathscr{P}}$ and observed a play ρ, deviators represent the players that must have changed their strategies from $\sigma_{\mathscr{P}}$ in order to generate ρ.

Lemma 145. *Given a play* ρ*, strategy profile* $\sigma_{\mathscr{P}}$*, a coalition* $\mathscr{C}$ *contains* $\mathsf{Dev}(\rho)$*, if and only if, there exists a strategy* $\sigma'_{\mathscr{C}}$ *such that* $\mathsf{Out}(\rho_1, \sigma_{-\mathscr{C}}, \sigma'_{\mathscr{C}}) = \rho$*.*

Proof. Assume that coalition $\mathscr{C}$ contains $\mathsf{Dev}(\rho, \sigma_{\mathscr{P}})$. We define the strategy $\sigma_{\mathscr{C}}$ to be such that for all $i \in \mathbb{N}$, $\sigma_{\mathscr{C}}(\rho_{\leq i}) = (\mathsf{move}_i(\rho))_{\mathscr{C}}$. By hypothesis, we have, for all indices i, $\mathsf{Dev}(\mathsf{move}_i(\rho), \sigma_{\mathscr{P}}(\rho_{\leq i})) \subseteq \mathscr{C}$, so for all players $A \notin \mathscr{C}$, $\sigma_A(\rho_{\leq i}) = (\mathsf{move}_i(\rho))_A$. Then $\Delta(\rho_i, \sigma'_{\mathscr{C}}(\rho_{\leq i}), \sigma_{-\mathscr{C}}(\rho_{\leq i})) = \rho_{i+1}$. Hence ρ is the outcome of the profile $(\sigma_{-\mathscr{C}}, \sigma'_{\mathscr{C}})$.

For the other direction, let $\sigma_{\mathscr{P}}$ be a strategy profile, $\sigma'_{\mathscr{C}}$ a strategy for coalition $\mathscr{C}$, and $\rho \in \mathit{Out}_G(\rho_0, \sigma_{-\mathscr{C}}, \sigma'_{\mathscr{C}})$. For all indices i, $\mathsf{move}_i(\rho) = (\sigma_{-\mathscr{C}}(\rho_{\leq i}), \sigma'_{\mathscr{C}}(\rho_{\leq i}))$. Therefore for all players $A \notin \mathscr{C}$, $(\mathsf{move}_i(\rho))_A = \sigma_A(\rho_{\leq i})$. Consequently, this implies that $\mathsf{Dev}(\mathsf{move}_i(\rho), \sigma_{\mathscr{P}}(\rho_{\leq i})) \subseteq \mathscr{C}$. Hence $\mathsf{Dev}(\rho, \sigma_{\mathscr{P}}) \subseteq \mathscr{C}$. □

Example 32. *In the example of Figure 13.1, we consider again the strategies, such that for all histories* π*,* $\sigma_{P_1}(\pi) = a$*,* $\sigma_{P_2}(v_0) = a$ *and if* $\pi \neq v_0$*,* $\sigma_{P_2}(\pi) = b$*, and* $\sigma_{P_3}(\pi) = b$*. Then* $\mathsf{Dev}(v_0 \cdot (a,a,b) \cdot v_2 \cdot (a,a,b) \cdot v_1 \cdot (a,b,a) \cdot v_2, \sigma_{\mathscr{P}})$ *is the union of:*

- $\mathsf{Dev}(\sigma_{\mathscr{P}}(v_0),(a,a,b)) = \mathsf{Dev}((a,a,b),(a,a,b)) = \varnothing$
- $\mathsf{Dev}(\sigma_{\mathscr{P}}(v_0\cdot(a,a,b)\cdot v_2),(a,a,b)) = \mathsf{Dev}((a,b,b),(a,a,b)) = \{P_2\}$
- $\mathsf{Dev}(\sigma_{\mathscr{P}}(v_0\cdot(a,a,b)\cdot v_2\cdot(a,a,b)\cdot v_1),(a,b,a)) = \mathsf{Dev}((a,b,b),(a,b,a)) = \{P_3\}$.

We obtain $\mathsf{Dev}(v_0\cdot(a,a,b)\cdot v_2\cdot(a,a,b)\cdot v_1\cdot(a,b,a)\cdot v_2,\sigma_{\mathscr{P}}) = \{P_2,P_3\}$. *This means that both* P_2 *and* P_3 *need to change their strategies from* $\sigma_{\mathscr{P}}$ *to obtain the given history.*

Note that Nash equilibria are defined only with respect to deviations by single players, that is, we require all players to achieve worse or equal payoffs than the prescribed profile when they single-handedly change strategies. Thus, only the outcomes with singleton deviator sets will be of interest for us in the next section where we present the algorithm.

13.1.4 Deviator Game

We now present an algorithm to reduce multiplayer games to two-player games using the notion of deviators we just defined. Given a *game* $\mathscr{G} = (\mathscr{A},(f_A)_{A\in\mathscr{P}})$, we define the deviator game, denoted $\mathsf{DevGame}(\mathscr{G})$. Intuitively, in this game, Eve needs to play according to an equilibrium, while Adam tries to find a profitable deviation for any player. The vertices are $V' = V\times 2^{\mathscr{P}}$, where the second component, a subset of $\mathscr{P}$, records the deviators of the current history.

At each step, Eve chooses an action profile, and Adam chooses the move that will apply. Adam can either respect Eve's choice, or pick a different action profile in which case the deviators will be added to the second component of the vertex. The game begins in $(v_0,\varnothing)$ and then proceeds as follows: from a vertex (v,D), Eve chooses an action profile $a_{\mathscr{P}}$, and Adam chooses a possibly different one $a'_{\mathscr{P}}$. The next vertex is $(\Delta(v,a'_{\mathscr{P}}),D\cup\mathsf{Dev}(a_{\mathscr{P}},a'_{\mathscr{P}}))$.

Example 33. *An example of a partial construction of the deviator game for the example of Figure 13.1, is given in Figure 13.2. We cannot represent the full construction here, as there are 40 vertices.*

We define projections proj_V and $\mathsf{proj}_{\mathsf{Dev}}$ from V' to V and from V' to $2^{\mathscr{P}}$ respectively, as well as proj_A from $Act^{\mathscr{P}}\times Act^{\mathscr{P}}$ to $Act^{\mathscr{P}}$ which maps to the second component of the product, that is, Adam's action.

For a history or play ρ, define $\pi_{\mathsf{Out}}(\rho)$ as the play ρ' for which, $\rho'_i = \mathsf{proj}_V(\rho_i)$ and $\mathsf{move}_i(\rho') = \mathsf{proj}_A(\mathsf{move}_i(\rho))$ for all i. This is thus the play induced by Adam's actions. Let us also denote $\mathsf{Dev}(\rho) = \mathsf{proj}_{\mathsf{Dev}}(\mathrm{last}(\rho))$.

We can associate a strategy of Eve to each strategy profile $\sigma_{\mathscr{P}}$ such that she chooses the moves prescribed by $\sigma_{\mathscr{P}}$ at each history of $\mathsf{DevGame}(\mathscr{G})$. Formally, we write $\kappa(\sigma_{\mathscr{P}})$ for the strategy defined by $\kappa(\sigma_{\mathscr{P}})(\pi) = \sigma_{\mathscr{P}}(\mathsf{proj}_{\mathsf{Out}}(\pi))$ for all histories π.

The following lemma states the correctness of the construction of the deviator game $\mathsf{DevGame}(\mathscr{G})$, in the sense that it records the set of deviators in the strategy profile suggested by Adam with respect to the strategy profile suggested by Eve.

Lemma 146. *Let* $\mathscr{G}$ *be a multiplayer game,* v *a vertex,* $\sigma_{\mathscr{P}}$ *a strategy profile, and* $\sigma_\exists = \kappa(\sigma_{\mathscr{P}})$ *the associated strategy in the deviator game.*

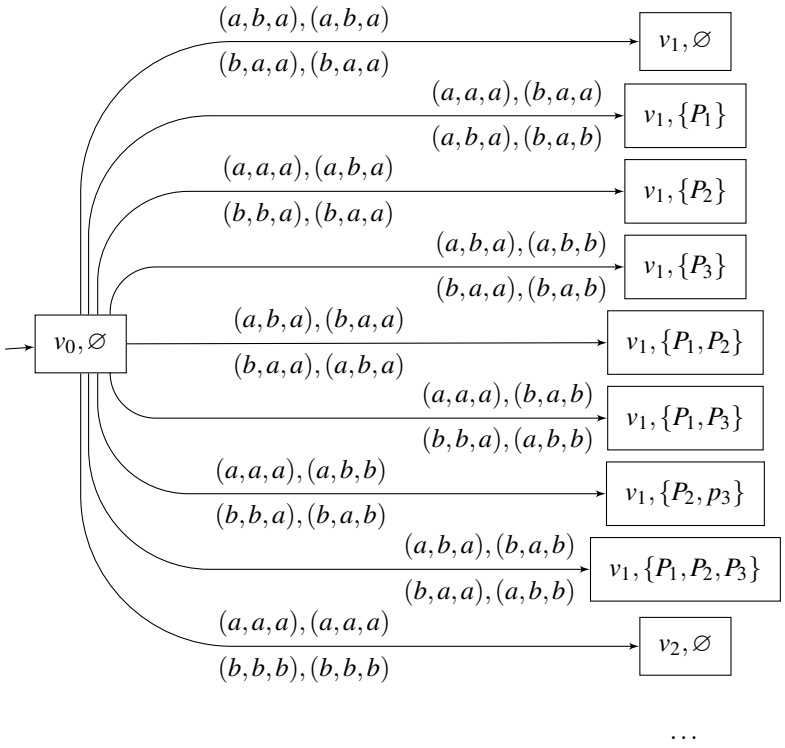


Figure 13.2: Example a deviator game construction.

1. *If* $\rho \in Out_{DevGame(\mathscr{G})}((v,\emptyset),\sigma_\exists)$*, then* $Dev(proj_{Out}(\rho),\sigma_{\mathscr{P}}) = Dev(\rho)$.
2. *If* $\rho \in Out_G(v)$ *and for all index i,* $\rho'_i = (\rho_i, Dev(\rho_{\leq i},\sigma_{\mathscr{P}}))$ *and* $move_i(\rho') = (\sigma_{\mathscr{P}}(\rho_{\leq i}), move_i(\rho))$*, then* $\rho' \in Out_{DevGame(\mathscr{G})}((v,\emptyset),\sigma_\exists)$.

Proof. We prove that for all i, $\mathsf{Dev}(\mathsf{proj}_{\mathsf{Out}}(\rho_{\leq i},\sigma_{\mathscr{P}}) = \mathsf{proj}_{\mathsf{Dev}}(\rho_{\leq i})$, which implies the property. The property holds for i = 0, since initially both sets are empty. Assume now that it holds for $i \geq 0$. Then:

$$
\begin{aligned}
&\mathsf{Dev}(\mathsf{proj}_{\mathsf{Out}}(\rho_{\leq i+1}),\sigma_{\mathscr{P}})\\
&= \mathsf{Dev}(\mathsf{proj}_{\mathsf{Out}}(\rho_{\leq i}),\sigma_{\mathscr{P}}) \cup \mathsf{Dev}(\sigma_{\mathscr{P}}(\mathsf{proj}_{\mathsf{Out}}(\rho_{\leq i})), \mathsf{proj}_A(\mathsf{move}_{i+1}(\rho)))\\
&\quad \text{(by definition of deviators)}\\
&= \mathsf{Dev}(\rho_{\leq i}) \cup \mathsf{Dev}(\sigma_{\mathscr{P}}(\mathsf{proj}_{\mathsf{Out}}(\rho_{\leq i}), \mathsf{proj}_A(\mathsf{move}_{i+1}(\rho)))\\
&\quad \text{(by induction hypothesis)}\\
&= \mathsf{Dev}(\rho_{\leq i}) \cup \mathsf{Dev}(\sigma_\exists(\rho_{\leq i}), \mathsf{proj}_A(\mathsf{move}_{i+1}(\rho)))\\
&\quad \text{(by definition of } \sigma_\exists \text{)}\\
&= \mathsf{Dev}(\rho_{\leq i}) \cup \mathsf{Dev}(\mathsf{move}_{i+1}(\rho))\\
&\quad \text{(by assumption } \rho \in \mathsf{Out}_{\mathsf{DevGame}(\mathscr{G})}((v,\emptyset),\sigma_\exists))\\
&= \mathsf{Dev}(\rho_{\leq i+1})\\
&\quad \text{(by construction of } \mathsf{DevGame}(\mathscr{G}))
\end{aligned}
$$

Which concludes the induction.

We now prove the second part. The property is shown by induction. It holds for v_0. Assume it is true up to index $i > 0$, then

$$
\begin{aligned}
&\Delta'(\rho'_i, \sigma_\exists(\rho'_{\leq i}), \mathsf{move}_i(\rho))\\
&= \Delta'((\rho_i, \mathsf{Dev}(\rho_{\leq i},\sigma_{\mathscr{P}})), \sigma_\exists(\rho'_{\leq i}), \mathsf{move}_i(\rho))\\
&\quad \text{(by definition of } \rho' \text{)}\\
&= \Delta(\rho_i, \mathsf{move}_i(\rho)), \mathsf{Dev}(\rho_{\leq i},\sigma_{\mathscr{P}}) \cup \mathsf{Dev}(\sigma_\exists(\rho'_{\leq i}), \rho_{i+1}))\\
&\quad \text{(by construction of } \Delta' \text{)}\\
&= (\rho_{i+1}, \mathsf{Dev}(\rho_{\leq i},\sigma_{\mathscr{P}}) \cup \mathsf{Dev}(\sigma_\exists(\rho'_{\leq i}), \rho_{i+1}))\\
&\quad \text{(since } \rho \text{ is an outcome of the game)}\\
&= (\rho_{i+1}, \mathsf{Dev}(\rho_{\leq i},\sigma_{\mathscr{P}}) \cup \mathsf{Dev}(\sigma_{\mathscr{P}}(\rho_{\leq i}), \rho_{i+1}))\\
&\quad \text{(by construction of } \sigma_\exists\text{)}\\
&= (\rho_{i+1}, \mathsf{Dev}(\rho_{\leq i+1},\sigma_{\mathscr{P}}))\\
&\quad \text{(by definition of deviators)}\\
&= \rho'_{i+1}.
\end{aligned}
$$

□

The objective of Eve in the deviator game is defined so that winning strategies correspond to equilibria of the original game. First, as an intermediary step, given coalition $\mathscr{C}$, player P and bound b, we will construct an objective stating that we can ensure that the payoff of P will not exceed b even if players in $\mathscr{C}$ change their strategies. Consider the following objective in $\mathsf{DevGame}(\mathscr{G})$:

$$\Omega(\mathscr{C},P,b) = \{\rho \in \mathsf{Out}_{\mathsf{DevGame}(\mathscr{G})} \mid \mathsf{Dev}(\rho) \subseteq \mathscr{C} \Rightarrow f_P(\mathsf{proj}_{\mathsf{Out}}(\rho)) \leq b\}.$$

Intuitively, this says that if only players in $\mathscr{C}$ deviated from the strategy suggested by Eve, then the payoff of P is smaller than b. We now show that a strategy ensuring bound b for the payoff of P against coalition $\mathscr{C}$ corresponds to a winning strategy for $\Omega(\mathscr{C},P,b)$ in the deviator game.

Lemma 147. *Let $\mathscr{C} \subseteq \mathscr{P}$ be a coalition, $\sigma_{\mathscr{P}}$ be a strategy profile, $b \in \mathbb{R}$ a bound, and P a player. For all strategies $\sigma'_{\mathscr{C}}$, vertex v_0, and coalition $\mathscr{C}$, the following equivalence holds: $f_P(\mathsf{Out}_{\mathsf{DevGame}(\mathscr{G})}(v_0, \sigma_{-\mathscr{C}}, \sigma'_{\mathscr{C}})) \leq b$ if, and only if, $\kappa(\sigma_{\mathscr{P}})$ is winning in $\mathsf{DevGame}(\mathscr{G})$ for objective $\Omega(\mathscr{C},P,b)$.*

Proof. Let ρ be an outcome of $\sigma_\exists = \kappa(\sigma_{\mathscr{P}}) \in \mathsf{DevGame}(\mathscr{G})$. By Lemma 146, we have that $\mathsf{Dev}(\rho) = \mathsf{Dev}(\mathsf{proj}_V(\rho), \sigma_{\mathscr{P}})$. By Lemma 145, $\mathsf{proj}_V(\rho)$ is the outcome of $(\sigma_{-\mathsf{Dev}(\rho)}, \sigma'_{\mathsf{Dev}(\rho)})$ for some $\sigma'_{\mathsf{Dev}(\rho)}$. If $\mathsf{Dev}(\rho) \subseteq \mathscr{C}$, then

$$f_P(\mathsf{proj}_V(\rho)) = f_P(\sigma_{-\mathscr{C}}, \sigma_{\mathscr{C}\setminus\mathsf{Dev}(\rho)}, \sigma'_{\mathsf{Dev}(\rho)}) = f_P(\sigma_{-\mathscr{C}}, \sigma''_{\mathscr{C}})$$

where $\sigma''_P = \sigma'_P$ if $P \in \mathsf{Dev}(\rho)$ and σ_P otherwise. By hypothesis, this payoff is smaller than or equal to b. This holds for all outcomes ρ of $\sigma_\exists$, thus $\sigma_\exists$ is a winning strategy for $\Omega(\mathscr{C},P,b)$.

For the other direction, assume $\sigma_\exists = \kappa(\sigma_{\mathscr{P}})$ is a winning strategy in $\mathsf{DevGame}(\mathscr{G})$ for $\Omega(\mathscr{C},P,b)$. Let $\sigma'_{\mathscr{C}}$ be a strategy for $\mathscr{C}$ and ρ the outcome of $(\sigma'_{\mathscr{C}}, \sigma_{-\mathscr{C}})$. By Lemma 145, $\mathsf{Dev}(\rho, \sigma_{\mathscr{P}}) \subseteq \mathscr{C}$. By Lemma 146, $\rho' = (\rho_j, \mathsf{Dev}(\rho_{\leq j}, \sigma_{\mathscr{P}}))_{j\in\mathbb{N}}$ is an outcome of $\sigma_\exists$. We have that $\mathsf{Dev}(\rho') = \mathsf{Dev}(\rho, \sigma_{\mathscr{P}}) \subseteq \mathscr{C}$. Since $\sigma_\exists$ is winning, ρ is such that $f_P(\mathsf{proj}_V(\rho)) \leq b$. Since $f_P(\mathsf{proj}_V(\rho')) = f_P(\rho)$, this shows that for all strategies $\sigma'_{\mathscr{C}}$, $f_P(\sigma_{-\mathscr{C}}, \sigma'_{\mathscr{C}}) \leq b$. □

Now, Eve can show that there is a Nash equilibrium in a given game by proving that whenever there is a single deviator, the deviating player does not gain more than without the deviation, while she does not have to prove anything on plays involving several deviators.

Theorem 139. *Let $\mathscr{G} = (\mathscr{A}, (f_P)_{P\in\mathscr{P}})$ be a game, $\sigma_{\mathscr{P}}$ a strategy profile in $\mathscr{G}$, vertex v_0, and $F = (f_P(\mathsf{Out}_{\mathscr{A}}(v_0, \sigma_{\mathscr{P}})))_{P\in\mathscr{P}}$ the payoff profile of $\sigma_{\mathscr{P}}$ from v_0. The strategy profile $\sigma_{\mathscr{P}}$ is a Nash equilibrium if, and only if, strategy $\kappa(\sigma_{\mathscr{P}})$ is winning in $\mathsf{DevGame}(\mathscr{A})$ for the objective $N(F)$ defined by:*

$$N(F) = \{\rho \mid |\mathsf{Dev}(\rho)| \neq 1\} \cup \bigcup_{P\in\mathscr{P}} \{\rho \mid \mathsf{Dev}(\rho) = \{P\} \wedge f_P(\mathsf{proj}_{\mathsf{Out}}(\rho)) \leq F_P\}.$$

Proof. By Lemma 147, $\sigma_{\mathscr{P}}$ is a Nash equilibrium if, and only if, for each player P, $\kappa(\sigma_{\mathscr{P}})$ is winning for $\Omega(\{P\}, P, F_P)$. So it is enough to show that for each player P, $\kappa(\sigma_{\mathscr{P}})$ is winning for $\Omega(\{P\}, P, F_P)$ if, and only if, $\kappa(\sigma_{\mathscr{P}})$ is winning for $N(F)$.

Implication Let ρ be an outcome of $\kappa(\sigma_{\mathscr{P}})$.

- If $|\mathsf{Dev}(\rho)| \neq 1$, then ρ is in $N(F)$ by definition.
- If $|\mathsf{Dev}(\rho)| = 1$, then for $\{P\} = \mathsf{Dev}(\rho)$, $f_P(\mathsf{proj}_{\mathsf{Out}}(\rho)) \leq F_P$ because $\kappa(\sigma_{\mathscr{P}})$ is winning for $\Omega(\mathsf{Dev}(\rho), P, F_P)$. Therefore ρ is in $N(F)$.

This holds for all outcomes ρ of $\kappa(\sigma_{\mathscr{P}})$ and shows that $\kappa(\sigma_{\mathscr{P}})$ is winning for $N(F)$.

Reverse implication Assume that $\kappa(\sigma_{\mathscr{P}})$ is winning for $N(F)$. We now show that $\kappa(\sigma_{\mathscr{P}})$ is winning for $\Omega(\{P\}, P, F_P)$ for each player P. Let ρ be an outcome of $\kappa(\sigma_{\mathscr{P}})$, we have $\rho \in N(F)$. We show that ρ belongs to $\Omega(\{P\}, P, F_P)$:

- If $\mathsf{Dev}(\rho) = \varnothing$ then $\rho = \mathsf{Out}(v_0, \sigma_{\mathscr{P}})$ and $f_P(\rho) = F_P$, so ρ is in $\Omega(\{P\}, P, F_P)$
- If $\mathsf{Dev}(\rho) \not\subseteq \{P\}$, then $\rho \in \Omega(\mathscr{C}, P, F_P)$ by definition.
- Otherwise $\mathsf{Dev}(\rho) = \{P\}$. Since $\rho \in N(F)$, $f_P(\rho) \leq F_P$. Hence $\rho \in \Omega(\mathscr{C}, P, F_P)$.

This holds for all outcomes ρ of $\kappa(\sigma_{\mathscr{P}})$ and shows it is winning for $\Omega(\{P\}, P, F_P)$ for each player $P \in \mathscr{P}$, which shows that $\sigma_{\mathscr{P}}$ is a Nash equilibrium. □

Algorithm for Parity Objectives

We now focus on the case of Parity objectives. Recall that Each player P has a colouring function $c_P : V \to \mathbb{N}$, inducing the parity objective Ω_A. Thus the payoff f_P assigns 1 to paths belonging to Ω_A and 0 to the others.

We now give an algorithm for the Nash equilibrium problem with parity objectives. Given a payoff for each player $(F_P)_{P \in \mathscr{P}}$, we can deduce from the previous theorem an algorithm that constructs a Nash equilibrium if there exists one. We construct the deviator game and note that we can reduce the number of vertices as follows: since $\mathsf{Dev}(\rho_{\leq k})$ is nondecreasing, we know that Eve wins whenever this set has at least two elements. In the construction, states with at least two deviators can be replaced by a sink vertex that is winning for Eve. This means that the constructed game has at most $n \times (|\mathscr{P}| + 1) + 1$ states.

The objective can be expressed as a Parity condition in the following way:

- for each vertex $v' = (v, \{P\})$, $c'(v') = c_P(v) + 1$ if $F_P = 0$ and $2 \cdot \max_v c_P(v)$ otherwise;
- for each vertex $v' = (v, D)$ with $|D| \neq 1$, $c'(v') = 2 \cdot \max_v c_P(v)$ i.e. it is winning for Eve.

Notice that the colouring function c' inverts the parity in the case where there is a single deviator who is losing in the prescribed strategy profile (that is, $F_P = 0$). In fact, when $F_P = 1$, the player cannot obtain more since they are already winning so the colour is set to $2 \cdot \max_v c_P(v)$ which is winning for Eve.

Lemma 148. *We have* $\mathrm{maxinf}(c'(\rho_i)) \in 2\mathbb{N}$ *if, and only if,* $\rho \in N(F)$*, where* $N(F)$ *is as defined in Theorem 139.*

Proof. For the implication, we will prove the contrapositive. Let ρ be a play not in $N(F)$, then since the deviators can only increase along a play, we have that $\mathsf{Dev}(\rho) = \{P\}$ for some player P and $f_P(\rho) > F_P$. This means $F_P = 0$ and $\mathrm{maxinf}(c_P(\rho_i)) \in 2\mathbb{N}$. By definition of c' this implies that $\mathrm{maxinf}(c'(\rho_i)) \in 2\mathbb{N}+1$ which proves the implication.

For the other implication, let ρ be such that $\mathrm{maxinf}(c'(\rho_i)) \in 2\mathbb{N}+1$. By definition of c' this means ρ contains infinitely many states of the form $(v,\{P\})$ with $F_P = 0$. Since the deviators only increase along the run, there is a player P such that ρ stays in the component $V \times \{P\}$ after some index k. Then for $i \geq k$, $c'(\rho_i) = c_P(\rho_i)+1$, hence $\mathrm{maxinf}(c'(\rho_i)) = \mathrm{maxinf}(c_P(\rho_i))+1$. Therefore $\mathrm{maxinf}(c_P(\rho_i)) \in 2\mathbb{N}$, which means $f_P(\rho) = 1 > F_P$. By definition of $N(F)$, $\rho \notin N(F)$. □

Given that the size of the game is polynomial and that parity games can be decided in quasi-polynomial time (see Chapter 3), the above lemma implies the following theorem.

Theorem 140. *For parity games, there is a quasi-polynomial algorithm to decide whether there is a Nash equilibrium with a given payoff.*

13.1.5 Extensions of Nash Equilibria

Subgame Perfect Equilibria

Nash equilibria present the disadvantage that once a player has deviated, the others will try to punish him, forgetting everything about their own objectives. If we were to observe the game after this point of deviation, it would not look like the players are playing rationally and in fact it would not correspond to a Nash equilibrium. The concept of *subgame perfect equilibria* refines the concept of Nash equilibrium by imposing that at each step of the history, the strategy behaves like Nash equilibrium if we were to start the game now. Formally, let us write $\sigma_P \circ h$ the strategy which maps all histories h' to $\sigma_P(h \cdot h')$, that is the strategy that behave like σ_P after h. Then $(\sigma_P)_{P\in\mathscr{P}}$ is a *subgame perfect equilibrium* if for all histories h, $(\sigma_P \circ h)_{P\in\mathscr{P}}$ is a Nash equilibrium.

Imposing such a strong restriction is justified by the fact that subgame perfect Nash equilibria exist for a large class of games. In particular subgame perfect equilibria always exist in turn-based games with reachability objectives.

Example 34. *Consider the example of Figure 13.3. There is a Nash equilibria whose outcome goes through states $v_0 \to v_1 \to \Omega_1$. In this equilibrium, Player 1 should play b in v_2, so that the best response of Player 2 is to play a at v_0. Intuitively, player 1 is threatening player 2, to make them both lose from v_2, but this threat is not credible, and the profile is not a subgame perfect equilibrium. In fact, once v_2 is reached it is better for Player 1 to play a so it is unlikely that the player will execute the said threat. The only subgame perfect equilibrium of this game ends in the vertex satisfying both Ω_1 and Ω_2.*

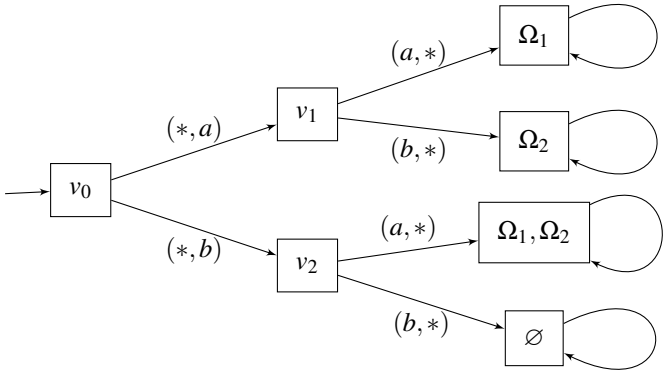


Figure 13.3: Two-player game with reachability objectives. The goal of player 1 is to reach a state labeled with Ω_1 and that of player 2 is to reach a state labeled with Ω_2.

Robust equilibria

The notion of robust equilibria refines Nash equilibria in two ways:

- a robust equilibrium is *resilient*, i.e. when a small coalition change its strategy, none of the players of the coalition improves their payoff;
- it is *immune*, i.e. when a small coalition changes its strategy, it does not decrease the payoffs of the non-deviating players.

The size of small coalitions is determined by parameter k for resilience and t for immunity. When a strategy is both k-resilient and t-immune, it is called a (k,t)-robust equilibrium.

The motivation behind this concept is to address these two weaknesses of Nash equilibria:

- There is no guarantee on payoffs when two (or more) players deviate together. Such a situation can occur in networks if the same person controls several devices (a laptop and a phone for instance) and can then coordinate their behaviours. In this case, the devices would be considered as different players and Nash equilibria can offer no guarantee.
- When a deviation occurs, the strategies of the equilibrium can punish the deviating user without any regard for the payoffs of the others. This can result in a situation where, because of a faulty device, nobody can use the protocol anymore.

By comparison, finding resilient equilibria with $k > 1$, ensures that clients have no interest in forming coalitions (up to size k), and finding immune equilibria with

$t > 0$ ensures that other clients will not suffer from some players (up to t) behaving differently from what was expected.

The deviator construction can be reused for finding such equilibria. We only need to adapt the objectives. Given a game $G = (\mathscr{A}, (f_P)_{P\in\mathscr{P}})$, a strategy profile $\sigma_{\mathscr{P}}$, and parameters k, t, we have

- The strategy profile $\sigma_{\mathscr{P}}$ is k-resilient if, and only if, strategy $\kappa(\sigma_{\mathscr{P}})$ is winning in $\mathsf{DevGame}(\mathscr{A})$ for the *resilience objective* $\mathscr{R}(k,F)$ where $F = (f_P(\mathsf{Out}_{\mathscr{A}}(v_0,\sigma_{\mathscr{P}})))_{P\in\mathscr{P}}$ is the payoff profile of $\sigma_{\mathscr{P}}$ and $\mathscr{R}(k,F)$ is defined by:

$$\begin{array}{rcl}\mathscr{R}(k,F) & = & \{\rho \in \mathsf{Out}_{\mathsf{DevGame}(\mathscr{A})} \mid |\mathsf{Dev}(\rho)| > k\} \\ & \cup & \{\rho \in \mathsf{Out}_{\mathsf{DevGame}(\mathscr{A})} \mid |\mathsf{Dev}(\rho)| = k \wedge \forall P \in \mathsf{Dev}(\rho).\ f_P(\mathsf{proj}_{\mathsf{Out}}(\rho)) \leq F_P\} \\ & \cup & \{\rho \in \mathsf{Out}_{\mathsf{DevGame}(\mathscr{A})} \mid |\mathsf{Dev}(\rho)| < k \wedge \forall P \in \mathscr{P}.\ f_P(\mathsf{proj}_{\mathsf{Out}}(\rho)) \leq F_P\}.\end{array}$$

- The strategy profile $\sigma_{\mathscr{P}}$ is t-immune if, and only if, strategy $\kappa(\sigma_{\mathscr{P}})$ is winning for the *immunity objective* $\mathscr{I}(t,F)$ where $F = (f(\mathsf{Out}_{\mathscr{A}}(v_0,\sigma_{\mathscr{P}})))_{P\in\mathscr{P}}$ is the payoff profile of $\sigma_{\mathscr{P}}$ and $\mathscr{I}(t,F)$ is defined by:

$$\begin{array}{rcl}\mathscr{I}(t,F) & = & \{\rho \in \mathsf{Out}_{\mathsf{DevGame}(\mathscr{A})} \mid |\mathsf{Dev}(\rho)| > t\} \\ & \cup & \{\rho \in \mathsf{Out}_{\mathsf{DevGame}(\mathscr{A})} \mid \forall P \in \mathscr{P} \setminus \mathsf{Dev}(\rho).\ F_P \leq f_P(\mathsf{proj}_{\mathsf{Out}}(\rho))\}.\end{array}$$

- The strategy profile $\sigma_{\mathscr{P}}$ is a (k,t)-robust profile in G if, and only if, $\kappa(\sigma_{\mathscr{P}})$ is winning for the *robustness objective*

$$\mathscr{R}(k,t,F) = \mathscr{R}(k,F) \cap \mathscr{I}(t,F),$$

where $F = (f_P(\mathsf{Out}_{\mathscr{A}}(v_0,\sigma_{\mathscr{P}})))_{P\in\mathscr{P}}$ is the payoff profile of $\sigma_{\mathscr{P}}$.

We omit the proof and encourage the reader to do it by themselves.

Extension to games with hidden actions

In most practical cases, players only have a partial view of the state of the system; so they may not be able, for instance, to detect a deviating player immediately. Studying equilibria in general imperfect information as in Chapter 8 would be well adapted in such situations. Unfortunately, these games are too powerful in general since the existence of Nash equilibria is undecidable in this case.

Nevertheless, the problem is decidable for a restricted form of imperfect information where the players observe the visited states but do not see the played actions; thus the actions are *hidden*.

We will thus consider strategies defined as functions from V^* to Act, which represents the fact that players' decision can only depend on observed sequence of states but not on other players' actions.

In this case, deviators cannot be defined as obviously as before, as it may not always be possible to identify one unique deviator responsible for a deviation. The construction will thus maintain a set of *suspects*, those players that might have been responsible

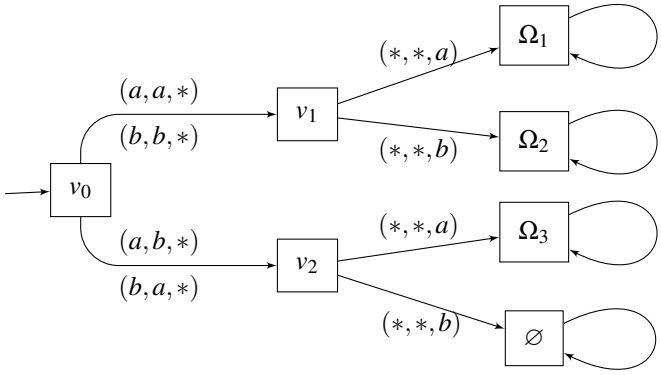


Figure 13.4: Three-player game with hidden actions. The goal of player i is to reach a state labeled with Ω_i.

for the observed deviation. Formally, suspects for an edge (v,v') with respect to a move $(a_P)_{P\in\mathscr{P}}$ are players P such that there is a'_P and $\Delta(a'_P,a_{-P})=(v,v')$. Rather than computing the union of deviators along a history, we now consider the intersection of suspects. That is, if at vertex v, the suspect set is S, and the strategy profile is $\sigma_{\mathscr{P}}$, and if the next vertex is v', then the suspect set becomes $S\cap\{P\in\mathscr{P}\mid\exists a'_P,\Delta(a'_P,a_{-P})=(v,v')\}$.

The *suspect game* can be defined just like the deviator game by replacing the deviators component by the suspects component. The objective for Eve is that no suspect player improves their payoff. In fact, in case of deviation, we know that the deviator belongs to the set of suspects although we cannot know which one has deviated for sure so Eve must ensure this for all suspects.

Example 35. *Consider the example of Figure 13.4. If actions were visible there would be an equilibrium ending in the state labeled with Ω_3: player 3 simply has to punish the player who would deviate from this path. But if we now consider hidden actions, in case of a deviation, player 3 would observe that the play went arrives in v_1 instead of v_2 and both Player 1 and Player 2 are suspects. Since Player 3 cannot punish both players at the same time, there is no Nash equilibrium ending satisfying Ω_3.*

13.2 Admissible strategies

Nash equilibria and their variants seen so far describe stable situations from which players have no incentive to deviate. This however is of limited use in some situations. First, the stability relies on the fact that all players are informed of the strategy profile to be played; that is, some central authority needs to publicly announce the strategies for all players. Second, each equilibrium describes a single possible situation. If there are several equilibria, it is not clear which one is to be chosen.

Table 13.3: A normal form game solvable by iterated elimination.

	A	B
C	3, 3	4, 1
D	0 , 4	0, 0

Rather than concentrating on particular equilibria, game theorists have studied reasonings players may follow in order to exclude some strategies that are necessarily worse than others. These worse strategies are called *dominated*. By formalizing dominated and non-dominated strategies for a given player, one can then predict the behaviour of rational players since such a player would never use a dominated strategy but rather always pick the best one available.

In this section, we will formalize dominated strategies and show how these can be computed in games. We then briefly show that this reasoning can be repeated, and present the iterated elimination of dominated strategies.

13.2.1 Definition

The notion of *dominance* is used to compare strategies with respect to payoffs they yield against the rest of the players' strategies. Consider the example of Table 13.3. Given a strategy of the second player, playing B is always at least as good as playing A for the first player. In fact, again C, B yields a payoff of 4 which is better than 3, the payoff of A; and against D, both yield 0. The strategy B is said to dominate A. Intuitively, B is either better or as good as A *in all situations*, so playing B is the rational choice for Player 1.

Furthermore, by this analysis, Player 2 knows that Player 1 will player B. Given this information, the best response of Player 1 is to play C. By *iterated elimination*, we established that (B,C) should be the only strategy profile to be played by players following this reasoning.

Let us formalize this notion.

Definition 36 (Dominance)**.** *Let $S \subseteq \mathscr{S}^{\mathscr{P}}$ be a set of the form $S = S_1 \times S_2 \times \cdots \times S_n$ which we will call a rectangular set. Let $\sigma, \sigma' \in S_i$. Strategy σ* very weakly dominates *strategy σ' with respect to S, written $\sigma_i \geq_S \sigma'_i$, if from all vertices v_0:*

$$\forall \sigma_{-i} \in S_{-i}, f_i(\mathit{Out}(v_0, \sigma'_i, \sigma_{-i})) \geq f_i(\mathit{Out}(v_0, \sigma_i, \sigma_{-i})).$$

Strategy σ_i weakly dominates *strategy σ'_i in S, written $\sigma >_S \sigma'$, if $\sigma \geq_S \sigma'$ and $\neg(\sigma' \leq_S \sigma)$. A strategy that is not weakly dominated in S is* admissible *in S. The subscripts on $\geq_S$ and $>_S$ are omitted when the sets of strategies are clear from the context.*

Algorithms rely on the notion of *optimistic* and *pessimistic value* of a history. The pessimistic value is the maximum payoff that a player can ensure in the worst case

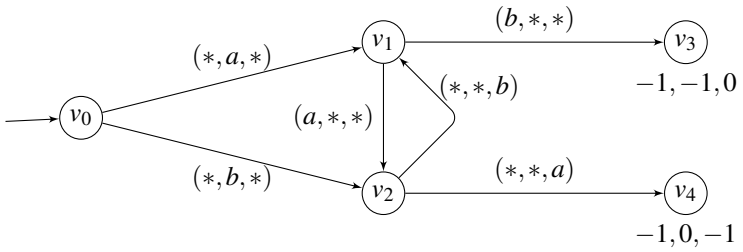


Figure 13.5: Example of a three-player turn-based simple safety game. Numbers below states describe the safety objective, for instance $-1,-1,0$ is losing for player 1 and 2.

within the strategy set S. The optimistic value is the best the player can achieve with the help of other players, given the strategy set S.

Definition 37 (Values)**.** *The* pessimistic value *of a strategy σ_i for a history h with respect to a rectangular set of strategies S, is*

- $\mathsf{pes}_i(S,h,\sigma_i) = \inf_{\sigma_{-i}\in S_{-i}} f_i(h \cdot \mathsf{Out}(\mathit{last}(h),\sigma_i,\sigma_{-i}))$.

The pessimistic value of a history *h for A_i with respect to a rectangular set of strategies S is given by:*

- $\mathsf{pes}_i(S,h) = \sup_{\sigma_i\in S_i} \mathsf{pes}_i(S,s,\sigma_i)$.

The optimistic value *of a strategy σ_i for a history h with respect to a rectangular set of strategies S is given by:*

- $\mathsf{opt}_i(S,h,\sigma_i) = \sup_{\sigma_{-i}\in S_{-i}} f_i(h_{\leq |h|-2} \cdot \mathsf{Out}(\mathit{last}(h),\sigma_i,\sigma_{-i}))$.

The optimistic value *of a history h for A_i with respect to a rectangular set of strategies S is given by:*

- $\mathsf{opt}_i(S,h) = \sup_{\sigma_i\in S_i} f_i(\mathsf{opt}_i(S,h,\sigma_i))$

We will first consider the case where S is the set of all strategies, and omit S in the above notations.

13.2.2 Simple Safety games

Simple safety games, are safety games in which there are no transitions from losing vertices to non-losing ones. Restricting to this particular class of game makes the problem simpler because the objective becomes prefix independent. Any safety game can be converted to an equivalent simple safety game by encoding in the states which players have visited so far a losing state. Note that this translation can be exponential in the number of players.

For simple safety games, the pessimistic and optimistic values do not depend on the full history but only on the last state: for all histories $\mathsf{pes}_i(h) = \mathsf{pes}_i(\text{last}(h))$ and $\mathsf{opt}_i(h) = \mathsf{opt}_i(\text{last}(h))$.

Note that in safety games (and any qualitative games) values can be only 1 (for winning) and 0 (for losing) and since the pessimistic value is always less than the optimistic one, the pair $(\mathsf{pes}_i, \mathsf{opt}_i)$ can only take three values: $(0,0)$, $(0,1)$ and $(1,1)$.

Intuitively, players should avoid decreasing this pair of values if they can. In fact, the characterization of admissible strategies we give below will be based on this simple observation.

Example 36. *An example of a simple safety game is given in Figure 13.5. In this game, player 1 controls v_1 where its optimistic value is 0, as it is possible that the outcome will never reach v_3 or v_4. However v_3 has optimistic value -1 for player 1, as it is a losing state for player 1. Going from v_1 to v_3 is a bad choice for player 1, and it is indeed dominated by the strategy that would always choose to go to v_2. In state v_3, player 3 has pessimistic value 0 since it can ensure not visiting v_4. The winning strategy for player 3 which is to always go to v_1 is also the only non-dominated strategy. For player 2, from state v_0, both choices lead to a state with values $(-1,0)$ so no choice is particularly better and both strategies are non-dominated.*

Definition 38. *Let D_i be the set of edges $(v,v') \in E$ such that $v \in V_i$ $\mathsf{pes}_i(v) > \mathsf{pes}_i(v')$ or $\mathsf{opt}_i(v) > \mathsf{opt}_i(v')$. These are called* dominated edges.

Theorem 141 (Characterisation of admissible strategies)**.** *Admissible strategies for player A_i are the strategies that never take actions in D_i.*

Proof. We show that if A_i plays an admissible strategy σ_i then the value cannot decrease on a transition controlled by A_i. Let $\rho \in \mathsf{Out}(\sigma_i, \sigma_{-i})$, and k an index such that $\rho_k \in V_i$. Let $(v,v') = \sigma_i(\rho_{\leq k})$:

- If $\mathsf{pes}_i(\rho_k) = 1$, then σ_i has to be winning against all strategies σ_{-i} of A_{-i}, otherwise it would be weakly dominated by such a strategy. Since there is no such strategy from a state with value $\mathsf{pes}_i \leq 0$, we must have $\mathsf{pes}_i(v') = 1$.
- If $\mathsf{opt}_i(\rho_k) = 1$, then there is a profile σ' such that $\rho = f(\mathsf{Out}(v, \sigma'))$, which is equal to 1. Assume that $\mathsf{opt}_i(v') = 0$. Then σ_i is dominated by the strategy σ_i'' obtained from σ_i by making it switch to σ' at v. In fact, σ_i is losing against all strategies of $-i$, while σ_i'' is winning at least against σ'_{-i}.
- If $\mathsf{pes}_i(v) = 0$ or $\mathsf{opt}_i(s) = 0$, then the value cannot decrease further.

In the other direction, let σ_i, σ_i' be two strategies of player A_i and assume $\sigma_i' >_S \sigma_i$. We will prove σ_i takes a transition in D_i at some point.

Let us fix some objects before developing the proof. There is a vertex v and strategy profile $\sigma_{-i} \in S_{-i}$ such that $f(\mathsf{Out}(v, \sigma_i', \sigma_{-i})) = 1 \wedge f(\mathsf{Out}(v, \sigma_i, \sigma_{-i}) = 0$. Let $\rho = \mathsf{Out}(v, \sigma_i, \sigma_{-i})$ and $\rho' = \mathsf{Out}(v, \sigma_i', \sigma_{-i})$. Consider the first position where these runs differ: write $\rho = w \cdot s' \cdot s_2 \cdot w'$ and $\rho' = w \cdot s' \cdot s_1 \cdot w''$.

The following are simple facts that can be seen easily:

- $s' \in V_i$, because the strategy of the other players are identical in the two runs.
- $\mathsf{opt}_i(s_1) = 1$ because $f(\mathsf{Out}(v, \sigma'_i, \sigma_{-i})) = 1$
- $\mathsf{pes}_i(s_2) = 0$ because $f(\mathsf{Out}(v, \sigma_i, \sigma_{-i})) = 0$

If $\mathsf{opt}_i(s_2) = 0$ or $\mathsf{pes}_i(s_1) = 1$ then $s' \to s_2 \in D_i$ so σ_i takes a transition of D_i. The remaining case to complete the proof is $\mathsf{opt}_i(s_2) = 1$ and $\mathsf{pes}_i(s_1) = 0$. Let us assume that σ_i does not take any edges from D_i. We will show that there is a strategy for $-i$ against which σ_i wins and σ'_i loses, which contradicts the hypothesis that σ'_i weakly dominates σ_i.

We first construct a profile $\sigma^2_{-i} \in S_{-i}$ such that $f(\mathsf{Out}(s_2, \sigma_i, \sigma^2_{-i})) = 1$. Strategy $\sigma^2_{-i} \in S_{-i}$ never decreases the optimistic value from 1 to 0 since the optimistic value is non-increasing. By assumption, σ_i itself does not decrease the value of A_i because it does not take transitions of D_i. So the outcome of $(\sigma_i, \sigma^2_{-i})$ never reaches a state of optimistic value 0. Hence it never reaches a state in Bad_i and therefore it is winning for A_i.

Let us now consider a profile $\sigma^1_{-i} \in S_{-i}$ such that $f(\sigma'_i, \sigma^1_{-i}) = 0$ from s_1. Such a strategy exists because $\mathsf{pes}_i(s_1) = 0$, so σ'_i is not a winning strategy. Then there exists a strategy profile σ^1_{-i} such that σ'_i loses from s_1.

Now consider strategy profile σ'_{-i} that plays like σ_{-i} if the play does not start with w; and otherwise switches to σ^1_{-i} at history ws_1 and to σ^2_{-i} at history ws_2. Formally, given a history h, $\sigma'_{-i}(h) =$

- $\sigma^1_{-i}(h')$ if $w \cdot s_1$ is a prefix of h and $w \cdot s_1 \cdot h' = h$
- $\sigma^2_{-i}(h')$ if $w \cdot s_2$ is a prefix of h and $w \cdot s_2 \cdot h' = h$
- $\sigma_{-i}(h)$ otherwise

Clearly we have $f_i(\mathsf{Out}_s(\sigma_i, \sigma'_{-i})) = 1 \wedge f_i(\mathsf{Out}_s(\sigma'_i, \sigma'_{-i})) = 0$, which contradicts $\sigma'_i \geq_S \sigma_i$. □

13.2.3 Parity games

The characterization given for simple safety game is not enough for parity objectives, as we will see in the following example.

Example 37. *Consider the example in Figure 13.6. In this example, although the strategy that always stays in v_0 does not decrease the value of player 1, it is dominated because it has no chance of winning. By contrast the strategy that always go to v_1 has a chance of being helped by player 2 and actually reaching Ω_1 it therefore dominates the first strategy.*

However the fact that an admissible strategy should not decrease its own value still holds. Assuming strategy σ_i of player P_i does not decrease its own value, we can classify its outcome in three categories according to their ultimate values.

- either ultimately $\mathsf{opt}_i = 0$, in which case all strategies are losing, and thus any strategy is admissible

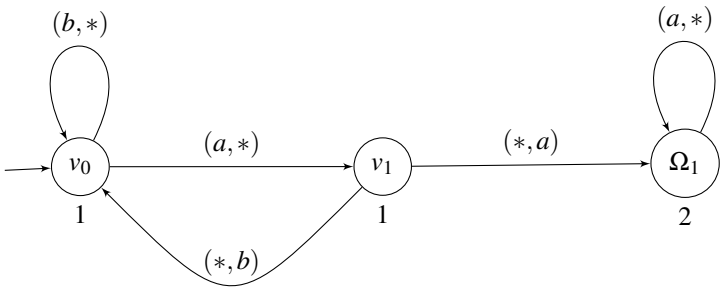


Figure 13.6: Parity game where the objective for player 1 is to visit Ω_1 infinitely often. Player 1 controls square vertices and player 2 round vertices.

- or ultimately $\mathsf{pes}_i = 1$, in which case admissible strategies are exactly the winning ones
- or ultimately $\mathsf{pes}_i = 0$ and $\mathsf{opt}_i = 1$. We will focus on this case which is more involved.

From a state of value 0, an admissible strategy of P_i should allow a winning play for P_i with the help of other players.

We write H_i for set of vertices v controlled by a player $P_j \neq P_i$ that have at least two successors of optimistic value 1. Formally, the *help-states* H_i of player P_i are defined as:

$$\bigcup_{P_j \in \mathscr{P} \setminus \{i\}} \left\{ s \in V_j \mid \exists s', s'',\ s' \neq s'' \wedge\ s \to s' \wedge s \to s'' \wedge\ \mathsf{opt}_i(s') = 1 \wedge\ \mathsf{opt}_i(s'') = 1 \right\}.$$

Intuitively, admissible strategies in the case satisfying $\mathsf{pes}_i = 0$ and $\mathsf{opt}_i = 1$ are those that visit infinitely often help states. In fact, letting other players make choices means that the player is allowing the possibility of them helping to achieve the objective. More precisely, we have the following property whose proof is omitted.

Lemma 149. *Let $v \in V$, $P_i \in \mathscr{P}$ and ρ a play be such that $\exists^\infty k.\mathsf{opt}_i(\rho_k) = 1$. There exists σ_i admissible such that $\rho \in \mathsf{Out}(v, \sigma_i)$ if, and only if, $f_i(\rho) = 1$ or $\exists^\infty k.\rho_k \in H_i$.*

13.2.4 Iterated elimination

Once each player is restricted to use admissible strategies, they can further refine their choices knowing that other players will not be using dominated strategies. We already saw this in the example of Table 13.3. In fact, once player 1 has eliminated strategy *A* (which is dominated by *B*), player 2 can use this information since its best response against the remaining strategies is *C*. In more complex games, this reasoning can be repeated and take several steps before converging. This repeated process is called *iterated elimination of dominated strategies*.

We now define this process formally.

Definition 39 (Iterated elimination)**.** *The* sequence of iterative elimination *is a sequence of rectangular strategy sets defined as follows.* $S^0 = (S_i^0)_{P_i \in \mathscr{P}}$ *is the set of all strategies. For* $k \geq 0$*, if we write* $S^k = (S_i)_{P_i \in \mathscr{P}}$*, then* S_i^{k+1} *is the set of strategies in* S_i^k *that are not dominated in* S^k*.*

Thus, the step 1 of the sequence of iterative elimination corresponds to admissible strategies defined above. Let us call them 1-admissible. In step 2, we again compute strategies that are dominated by only considering 1-admissible strategies for all players, and repeat. Strategies that survive all step of elimination are said *iteratively admissible*.

Theorem 142. *In parity games, the sequence of iterative elimination converges, and it reaches a non-empty fixed point.*

We prove this result only for simple safety games since the case of parity conditions is too complex for the scope of this book.

Intuitively, given a game G, Theorem 141 tells us that any strategy for player P_i that avoids using edges D_i is admissible. So if we remove all edges $\cup_{P_i \in \mathscr{P}} D_i$ from the game to obtain a new game called G_1, then all strategies of G_1 (for all players) are admissible in G, and conversely. We can then repeat this process to G_1: we construct G_2 by eliminating all dominated edges in G_1, and get that admissible strategies in G_1 are exactly all strategies of G_2, which correspond to 2-admissible strategies, and so on.

Since the size of the games decrease at each step, this process necessarily stops. It remains to show that the limit game G_∞ contains strategies. We will show that all vertices have at least one outgoing edge in G_∞. It suffices to show that the sets D_i never contains all edges leaving a vertex. Let us consider any game G_j. For a vertex $v \in V_i$ with $\mathsf{pes}_i(v) = 0$ and $\mathsf{opt}_i(v) = 0$, none of the edges are dominated. For a vertex $v \in V_i$ with $\mathsf{pes}_i(v) = 1$ (and necessarily $\mathsf{opt}_i(v) = 1$), there exists a winning strategy in G_j so there must be a successor v' with $\mathsf{pes}_i(v') = 1$ which is not dominated. Last, for a vertex $v \in V_i$ with $\mathsf{pes}_i(v) = 0$ and $\mathsf{opt}_i(v) = 1$, there exists a winning play from v so for some successor v' we must have $\mathsf{opt}_i(v') = 1$ which is an edge that is not dominated.

Bibliographic references

Most results about equilibria fall into two-categories: they either prove that equilibria always exist for some class of games, or they characterize the complexity of finding a particular one.

Existence results

Several authors have noticed that Nash equilibria always exist in turn-based game for some classes of objectives, in particular this is true of ω-regular objectives. The most general result of that kind shows that this is true for all objectives for which there exist finite memory optimal strategies [RP18].

The notion of equilibrium we now call Nash equilibrium was defined in the article of Nash [Nas50] in which he proves the existence for a class of normal form game. The Hawk-dove game we presented as an example in the first part of this chapter is

also called game of chicken. The first reference to this game was by Smith and Price [SP73]. The example of medium of access control we presented as a motivation was studied from a game theoric point of view in [MW03].

The notion of subgame perfect equilibria is interesting because in games on trees (or extensive games), for which they were originally introduced, they always exist. This results can be extended to games played on graphs. In particular subgame perfect equilibria always exist in reachability games [BBdPG12].

Algorithms and complexity results

The deviator construction and the algorithm presented in this chapter are based on [BBMU11, Bre12]. Algorithms on admissible strategies on infinite games and the complexity of the related problems were studied in [Ber07b, BRS14].

Imperfect information games in the context of multiplayer games are difficult. As soon as there are 'information forks' interesting problems are undecidable. Deciding whether two players can ensure an objective against a third player is undecidable. As a corollary the Nash equilibrium problem is also undecidable [PR90]. The problem of Nash equilibrium is also undecidable in stochastic games even with only three players [BMS14].

In the first section we presented a polynomial algorithm for finding pure Nash equilibria in normal form games. It is actually also possible to find a mixed Nash equilibria in polynomial time using linear programming. The same extends to finding memoryless mixed Nash equilibria in concurrent games, and even resilient equilibria [Bre16].

Action-graphs are succinct representation of matrix games. Indeed, representing games with matrices can be costly when the number of players increases. The size of the matrix is in fact exponential in the number of players: when each player has two strategies there are $2^{\mathscr{P}}$ cells in the table. The action-graph representation is more compact, and the representation can be exponentially smaller. Because of that, the algorithm is no longer polynomial. If there are no constraints on the Nash equilibrium we are looking for, the complexity of the problem cannot be characterized using classical classes like NP-completeness because equilibria always exist and thus the answer to the decision problem would always be true. The characterization of the complexity was done using the PPAD class [DGP09].

Nash equilibria with LTL objectives is expressible in logics such as strategy logic [CHP10] or ATL* [AHK02], as well as other extensions of this equilibria. However, satisfiability in these logic is difficult: it is 2EXP-complete for ATL* and undecidable for strategy logic in general. A decidable fragment of strategy logic has been identified [MMPV12], but remains difficult; it is 2EXP-complete.

Conclusions

The goal of this book was to give a technical account of the state of affairs on games on graphs. It emerged as a research topic already decades ago, and is now an active field with interactions with many other disciplines. In this conclusion, let us note that it has two appealing features:

- A well defined and small set of fundamental open problems. The most prominent ones are the complexity of solving parity games, mean payoff games, discounted games, and stochastic reachability games. Many others have been discussed in this book.
- A wealth of new models and directions. Let us cite as examples progress towards understanding the memory requirements [Ohl23, BRV23], bidding mechanisms [AHZ21], or distributed games [GMMW22].

Bibliography

[ABd03] Parosh Aziz Abdulla, Ahmed Bouajjani, and Julien d'Orso. Deciding monotonic games. In *Proceedings of the EACSL Annual Conference on Computer Science and Logic, CSL'03*, volume 2803. Springer-Verlag, 2003.

[ABDL18] Natasha Alechina, Nils Bulling, Stéphane Demri, and Brian Logan. On the complexity of resource-bounded logics. *Theoretical Computer Science*, 750, 2018.

[ABGJ14] Xavier Allamigeon, Pascal Benchimol, Stéphane Gaubert, and Michael Joswig. Combinatorial simplex algorithms can solve mean payoff games. *SIAM Journal on Control and Optimization*, 24, 2014.

[AC88] André Arnold and Paul Crubillé. A linear algorithm to solve fixed-point equations on transition systems. *Information Processing Letters*, 29, 1988.

[AČJT00] Parosh Aziz Abdulla, Kārlis Čerāns, Bengt Jonsson, and Yih-Kuen Tsay. Algorithmic analysis of programs with well quasi-ordered domains. *Information and Computation*, 160, 2000.

[ACTDG13] Marianne Akian, Jean Cochet-Terrasson, Sylvie Detournay, and Stéphane Gaubert. Solving multichain stochastic games with mean payoff by policy iteration. In *Proceedings of the IEEE Conference on Decision and Control, CDC'13*. IEEE Computer Society, 2013.

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126, 1994.

[ADD00] Robert B. Ash and Catherine A. Doléans-Dade. *Probability and Measure Theory*. Academic Press, 2000.

[AdMS21] David Auger, Xavier Badin de Montjoye, and Yann Strozecki. A generic strategy iteration method for simple stochastic games. In *Proceedings of*

the International Symposium on Mathematical Foundations of Computer Science, MFCS'21, volume 202. Schloß Dagstuhl, 2021.

[AG11] Krzysztof R. Apt and Erich Grädel. *Lectures in Game Theory for Computer Scientists*. Cambridge University Press, 2011.

[AGKS22] Xavier Allamigeon, Stéphane Gaubert, Ricardo D. Katz, and Mateusz Skomra. Universal complexity bounds based on value iteration and application to entropy games. In *Proceedings of the International Colloquium on Automata, Languages and Programming, ICALP'22*, volume 229. Schloß Dagstuhl, 2022.

[AHK02] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49, 2002.

[AHZ21] Guy Avni, Thomas A. Henzinger, and Dorde Zikelic. Bidding mechanisms in graph games. *Journal of Computer and System Sciences*, 119, 2021.

[AKV16] Shaull Almagor, Orna Kupferman, and Yaron Velner. Minimizing expected cost under hard Boolean constraints, with applications to quantitative synthesis. In *Proceedings of the International Conference on Concurrency Theory, CONCUR'16*, volume 59. Schloß Dagstuhl, 2016.

[AM09] Daniel Andersson and Peter Bro Miltersen. The complexity of solving stochastic games on graphs. In *Proceedings of the International Symposium on Algorithms and Computation, ISAAC'09*, volume 5878. Springer-Verlag, 2009.

[AMPS98] Eugene Asarin, Oded Maler, Amir Pnueli, and Joseph Sifakis. Controller synthesis for timed automata. In *Proceedings of the IFAC Conference on System Structure and Control, SSC'98*. Elsevier, 1998.

[AMSS13] Parosh Aziz Abdulla, Richard Mayr, Arnaud Sangnier, and Jeremy Sproston. Solving parity games on integer vectors. In *Proceedings of the International Conference on Concurrency Theory, CONCUR'13*, volume 8052. Springer-Verlag, 2013.

[AR17] Benjamin Aminof and Sasha Rubin. First-cycle games. *Information and Computation*, 254, 2017.

[Aum64] Robert J. Aumann. Mixed and behavior strategies in infinite extensive games. In *Advances in Game Theory*, volume 52. Princeton University Press, 1964.

[Bar68] Erwin H. Bareiss. Sylvester's identity and multistep integer-preserving Gaussian elimination. *Mathematics of Computation*, 22, 1968.

[BB79] Catriel Beeri and Philip A. Bernstein. Computational problems related to the design of normal form relational schemas. *ACM Transactions on Database Systems*, 4, 1979.

[BBC+14] Tomáš Brázdil, Václav Brožek, Krishnendu Chatterjee, Vojtech Forejt, and Antonín Kučera. Markov decision processes with multiple long-run average objectives. *Logical Methods in Computer Science*, 10, 2014.

[BBdPG12] Thomas Brihaye, Véronique Bruyère, Julie de Pril, and Hugo Gimbert. Subgame perfection for equilibria in quantitative reachability games. In *Proceedings of the International Conference on the Foundations of Software Science and Computational Structures, FoSSaCS'12*, 2012.

[BBEK11] Tomáš Brázdil, Václav Brožek, Kousha Etessami, and Antonín Kučera. Approximating the termination value of one-counter MDPs and stochastic games. In *Proceedings of the International Colloquium on Automata, Languages and Programming, ICALP'11*. Springer-Verlag, 2011.

[BBM06] Patricia Bouyer, Thomas Brihaye, and Nicolas Markey. Improved undecidability results on weighted timed automata. *Information Processing Letters*, 98, 2006.

[BBMU11] Patricia Bouyer, Romain Brenguier, Nicolas Markey, and Michael Ummels. Nash Equilibria in Concurrent Games with Büchi Objectives. In *Proceedings of the International Conference on the Foundations of Software Technology and Theoretical Computer Science, FSTTCS'11*, volume 13. Schloß Dagstuhl, 2011.

[BBR05] Thomas Brihaye, Véronique Bruyère, and Jean-François Raskin. On optimal timed strategies. In *Proceedings of the International Conference on Formal Modelling and Analysis of Timed Systems, FORMATS'05*, volume 3829. Springer-Verlag, 2005.

[BC18] Mikołaj Bojańczyk and Wojciech Czerwiński. An automata toolbox, 2018.

[BCD+07] Gerd Behrmann, Agnès Cougnard, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. UPPAAL-Tiga: Time for playing games! In *Proceedings of the International Conference on Computer Aided Verification, CAV'07*, volume 4590. Springer-Verlag, 2007.

[BCD+11] Luboš Brim, Jakub Chaloupka, Laurent Doyen, Rafaella Gentilini, and Jean-François Raskin. Faster algorithms for mean-payoff games. *Formal Methods for System Design*, 38, 2011.

[BCFK17] Tomáš Brázdil, Krishnendu Chatterjee, Vojtech Forejt, and Antonín Kučera. Trading performance for stability in Markov decision processes. *Journal of Computer and System Sciences*, 84, 2017.

[BCH+16] Romain Brenguier, Lorenzo Clemente, Paul Hunter, Guillermo A. Pérez, Mickael Randour, Jean-François Raskin, Ocan Sankur, and Mathieu Sassolas. Non-zero sum games for reactive synthesis. In *Proceedings of the International Conference on Language and Automata Theory and Applications, LATA'16*, volume 9618. Springer-Verlag, 2016.

[BCH+21] Christopher H. Broadbent, Arnaud Carayol, Matthew Hague, Andrzej S. Murawski, C.-H. Luke Ong, and Olivier Serre. Collapsible pushdown parity games. *ACM Transactions on Computational Logic*, 22, 2021.

[BCKN12] Tomáš Brázdil, Krishnendu Chatterjee, Antonín Kučera, and Petr Novotný. Efficient controller synthesis for consumption games with multiple resource types. In *Proceedings of the International Conference on Computer Aided Verification, CAV'12*, volume 7358. Springer, 2012.

[BCNV20] Tomáš Brázdil, Krishnendu Chatterjee, Petr Novotný, and Jiri Vahala. Reinforcement learning of risk-constrained policies in Markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI'20*. AAAI Press, 2020.

[BDMP03] Patricia Bouyer, Deepak D'Souza, Parthasarathy Madhusudan, and Antoine Petit. Timed control with partial observability. In *Proceedings of the International Conference on Computer Aided Verification, CAV'03*, volume 2725. Springer-Verlag, 2003.

[BDOR20] Thomas Brihaye, Florent Delgrange, Youssouf Oualhadj, and Mickael Randour. Life is random, time is not: Markov decision processes with window objectives. *Logical Methods in Computer Science*, 16, 2020.

[BEGM19] Endre Boros, Khaled Elbassioni, Vladimir A. Gurvich, and Kazuhisa Makino. A pseudo-polynomial algorithm for mean payoff stochastic games with perfect information and few random positions. *Information and Computation*, 267, 2019.

[Bel57] Richard Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, 1957.

[BEM97] Ahmed Bouajjani, Javier Esparza, and Oded Maler. Reachability analysis of pushdown automata: Application to model-checking. In *Proceedings of the International Conference on Concurrency Theory, CONCUR'97*, volume 1243. Springer, 1997.

[Ber07a] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2007.

[Ber07b] Dietmar Berwanger. Admissibility in infinite games. In *Proceedings of the International Symposium on Theoretical Aspects of Computer Science, STACS'07*, volume 4393. Springer-Verlag, 2007.

[BF68] David Blackwell and Chris Ferguson. The big match. *Annals of Mathematical Statistics*, 1968.

[BFL+08] Patricia Bouyer, Uli Fahrenberg, Kim G. Larsen, Nicolas Markey, and Jiří Srba. Infinite runs in weighted timed automata with energy constraints. In *Proceedings of the International Conference on Formal Modelling and Analysis of Timed Systems, FORMATS'08*, volume 5215. Springer-Verlag, 2008.

[BFM15] Patricia Bouyer, Erwin Fang, and Nicolas Markey. Permissive strategies in timed automata and games. In *Proceedings of the International Workshop on Automated Verification of Critical Systems, AVoCS'15*, volume 72. European Association of Software Science and Technology, 2015.

[BFRR17] Véronique Bruyère, Emmanuel Filiot, Mickael Randour, and Jean-François Raskin. Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games. *Information and Computation*, 254, 2017.

[BG11] Laura Bozzelli and Pierre Ganty. Complexity analysis of the backward coverability algorithm for VASS. In *Proceedings of the International Workshop on Reachability Problems, RP'11*, volume 6945. Springer-Verlag, 2011.

[BGG17] Nathalie Bertrand, Blaise Genest, and Hugo Gimbert. Qualitative determinacy and decidability of stochastic games with signals. *Journal of the ACM*, 64, 2017.

[BGHM17] Thomas Brihaye, Gilles Geeraerts, Axel Haddad, and Benjamin Monmege. Pseudopolynomial iterative algorithm to solve total-payoff games and min-cost reachability games. *Acta Informatica*, 54, 2017.

[BGMR17] Damien Busatto-Gaston, Benjamin Monmege, and Pierre-Alain Reynier. Optimal reachability in divergent weighted timed games. In *Proceedings of the International Conference on the Foundations of Software Science and Computational Structures, FoSSaCS'17*, volume 10203. Springer-Verlag, 2017.

[BGMR18] Patricia Bouyer, Mauricio González, Nicolas Markey, and Mickael Randour. Multi-weighted Markov decision processes with reachability objectives. In *Proceedings of the International Symposium on Games, Automata, Logics, and Formal Verification, GandALF'18*, volume 277, 2018.

[BGR20] Raphaël Berthon, Shibashis Guha, and Jean-François Raskin. Mixing probabilistic and non-probabilistic objectives in Markov decision processes. In *Proceedings of the Annual IEEE Symposium on Logic in Computer Science, LICS'20*. ACM, 2020.

[BHM+17] Patricia Bouyer, Piotr Hofman, Nicolas Markey, Mickael Randour, and Martin Zimmermann. Bounding average-energy games. In *Proceedings of the International Conference on the Foundations of Software Science and Computational Structures, FoSSaCS'17*, volume 10203, 2017.

[BHR16a] Véronique Bruyère, Quentin Hautem, and Mickael Randour. Window parity games: an alternative approach toward parity games with time bounds. In *Proceedings of the International Symposium on Games,*

Automata, Logics, and Formal Verification, GandALF'16, volume 226, 2016.

[BHR16b] Véronique Bruyère, Quentin Hautem, and Jean-François Raskin. On the complexity of heterogeneous multidimensional games. In *Proceedings of the International Conference on Concurrency Theory, CONCUR'16*, volume 59. Schloß Dagstuhl, 2016.

[BHRR19] Véronique Bruyère, Quentin Hautem, Mickael Randour, and Jean-François Raskin. Energy mean-payoff games. In *Proceedings of the International Conference on Concurrency Theory, CONCUR'19*, volume 140. Schloß Dagstuhl, 2019.

[BHSS12] Béatrice Bérard, Serge Haddad, Mathieu Sassolas, and Nathalie Sznadjer. Concurrent games on VASS with inhibition. In *Proceedings of the International Conference on Concurrency Theory, CONCUR'12*, volume 7454. Springer-Verlag, 2012.

[Bil95] Patrick Billingsley. *Probability and Measure*. John Wiley & Sons, 1995.

[BJK10] Tomáš Brázdil, Petr Jančar, and Antonín Kučera. Reachability games on extended vector addition systems with states. In *Proceedings of the International Colloquium on Automata, Languages and Programming, ICALP'10*, volume 6199. Springer-Verlag, 2010.

[BJM15] Patricia Bouyer, Samy Jaziri, and Nicolas Markey. On the value problem in weighted timed games. In *Proceedings of the International Conference on Concurrency Theory, CONCUR'15*, volume 42. Schloß Dagstuhl, 2015.

[BJW02] Julien Bernet, David Janin, and Igor Walukiewicz. Permissive strategies: from parity games to safety games. *Theoretical Informatics and Applications*, 36, 2002.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

[BK14] James Brotherston and Max I. Kanovich. Undecidability of propositional separation logic and its neighbours. *Journal of the ACM*, 61, 2014.

[BKN16] Tomáš Brázdil, Antonín Kučera, and Petr Novotný. Optimizing the expected mean payoff in energy Markov decision processes. In *Proceedings of the International Symposium on Automated Technology for Verification and Analysis, ATVA'16*, volume 9938, 2016.

[BL69] J. Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138, 1969.

[BM83] Bernard Berthomieu and Miguel Menasche. An enumerative approach for analyzing time Petri nets. In *Proceedings of the IFIP World Computer Congress, WCC'83*. North-Holland/IFIP, 1983.

[BM16] Manuel Bodirsky and Marcello Mamino. Max-closed semilinear constraint satisfaction. In *Proceedings of the International Computer Science Symposium in Russia, CSR'16*, volume 9691. Springer-Verlag, 2016.

[BMR+18] Patricia Bouyer, Nicolas Markey, Mickael Randour, Kim G. Larsen, and Simon Laursen. Average-energy games. *Acta Informatica*, 55, 2018.

[BMS14] Patricia Bouyer, Nicolas Markey, and Daniel Stan. Mixed Nash Equilibria in Concurrent Terminal-Reward Games. In *Proceedings of the International Conference on the Foundations of Software Technology and Theoretical Computer Science, FSTTCS'14*, 2014.

[BMS15] Patricia Bouyer, Nicolas Markey, and Ocan Sankur. Robust reachability in timed automata and games: A game-based approach. *Theoretical Computer Science*, 563, 2015.

[BORV21] Patricia Bouyer, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove. Arena-independent finite-memory determinacy in stochastic games. In *Proceedings of the International Conference on Concurrency Theory, CONCUR'21*, volume 203. Schloß Dagstuhl, 2021.

[BR15] Romain Brenguier and Jean-François Raskin. Pareto curves of multidimensional mean-payoff games. In *Proceedings of the International Conference on Computer Aided Verification, CAV'15*, volume 9207. Springer-Verlag, 2015.

[Bra79] Gilles Brassard. A note on the complexity of cryptography (corresp.). *ieee Trans. Information Theory*, 25, 1979.

[Bre12] Romain Brenguier. *Nash equilibria in concurrent games: applications to timed games*. PhD thesis, École Normale Supérieure de Cachan - ENS Cachan, 2012.

[Bre16] Romain Brenguier. Robust equilibria in mean payoff games. In *Proceedings of the International Conference on the Foundations of Software Science and Computational Structures, FoSSaCS'16*, 2016.

[BRO+22] Patricia Bouyer, Stéphane Le Roux, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove. Games where you can play optimally with arena-independent finite memory. *Logical Methods in Computer Science*, 18, 2022.

[BRR17] Raphaël Berthon, Mickael Randour, and Jean-François Raskin. Threshold constraints with guarantees for parity objectives in Markov decision processes. In *Proceedings of the International Colloquium on Automata,*

Languages and Programming, ICALP'17, volume 80. Schloß Dagstuhl, 2017.

[BRS14] Romain Brenguier, Jean-François Raskin, and Mathieu Sassolas. The complexity of admissibility in omega-regular games. In *Proceedings of the Joint Meeting of the EACSL Annual Conference on Computer Science and Logic and the Annual IEEE Symposium on Logic in Computer Science, CSL-LICS'14*. ACM, 2014.

[BRV22] Patricia Bouyer, Mickael Randour, and Pierre Vandenhove. The true colors of memory: A tour of chromatic-memory strategies in zero-sum games on graphs (invited talk). In *Proceedings of the International Conference on the Foundations of Software Technology and Theoretical Computer Science, FSTTCS'22*, volume 250. Schloß Dagstuhl, 2022.

[BRV23] Patricia Bouyer, Mickael Randour, and Pierre Vandenhove. Characterizing omega-regularity through finite-memory determinacy of games on infinite graphs. *TheoretiCS*, 2, 2023.

[BSV03] Henrik Björklund, Sven Sandberg, and Sergei Vorobyov. A discrete subexponential algorithm for parity games. In *Proceedings of the International Symposium on Theoretical Aspects of Computer Science, STACS'03*, volume 2607. Springer-Verlag, 2003.

[BSV04] Henrik Björklund, Sven Sandberg, and Sergei Vorobyov. Memoryless determinacy of parity and mean payoff games: a simple proof. *Theoretical Computer Science*, 310, 2004.

[BSW03] Alexis-Julien Bouquet, Olivier Serre, and Igor Walukiewicz. Pushdown games with unboundedness and regular conditions. In *Proceedings of the International Conference on the Foundations of Software Technology and Theoretical Computer Science, FSTTCS'03*, volume 2914. Springer-Verlag, 2003.

[BT97] Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific, 1997.

[Büc62] J. Richard Büchi. On a decision method in restricted second-order arithmetic. In *Proceedings of the International Congress of Logic, Methodology, and Philosophy of Science, CLMPS'60*. Stanford University Press, 1962.

[Büc64] J. Richard Büchi. Regular canonical systems. *Archive for Mathematical Logic*, 6, 1964.

[Büc77] J. Richard Büchi. Using determinancy of games to eliminate quantifiers. In *Fundamentals of Computation Theory77*, 1977.

[BV07] Henrik Björklund and Sergei Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. *Discrete Applied Mathematics*, 155, 2007.

[BY04] Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets*, volume 2098. Springer-Verlag, 2004.

[Cac02] Thierry Cachat. Two-way tree automata solving pushdown games. In *Automata, Logics, and Infinite Games: A Guide to Current Research*. Springer Berlin Heidelberg, 2002.

[Cac03] Thierry Cachat. *Games on pushdown graphs and extensions*. PhD thesis, RWTH Aachen, 2003.

[Cau88] Didier Caucal. Récritures suffixes de mots. Technical report, INRIA, 1988.

[CD12] Krishnendu Chatterjee and Laurent Doyen. Energy parity games. *Theoretical Computer Science*, 458, 2012.

[CDF+05] Franck Cassez, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. Efficient on-the-fly algorithms for the analysis of timed games. In *Proceedings of the International Conference on Concurrency Theory, CONCUR'05*, volume 3653. Springer-Verlag, 2005.

[CDF+19] Wojciech Czerwiński, Laure Daviaud, Nathanaël Fijalkow, Marcin Jurdziński, Ranko Lazić, and Paweł Parys. Universal trees grow inside separating automata: Quasipolynomial lower bounds for parity games. In *Proceedings of the Symposium on Discrete Algorithms, SODA'19*, 2019.

[CDH10a] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Qualitative analysis of partially-observable markov decision processes. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science, MFCS'10*. Springer-Verlag, 2010.

[CDH10b] Julien Cristau, Alexandre David, and Florian Horn. How do we remember the past in randomised strategies? In *Proceedings of the International Symposium on Games, Automata, Logics, and Formal Verification, GandALF'10*, volume 25, 2010.

[CDHR07] Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. Algorithms for omega-regular games of incomplete information. *Logical Methods in Computer Science*, 3, 2007.

[CDHR10] Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. Generalized mean payoff and energy games. In *Proceedings of the International Conference on the Foundations of Software Technology and Theoretical Computer Science, FSTTCS'10*, volume 8. Schloß Dagstuhl, 2010.

[CDL+07] Franck Cassez, Alexandre David, Kim G. Larsen, Didier Lime, and Jean-François Raskin. Timed control with observation based and stuttering invariant strategies. In *Proceedings of the International Symposium on Automated Technology for Verification and Analysis, ATVA'07*, volume 4762. Springer-Verlag, 2007.

[CDRR15] Krishnendu Chatterjee, Laurent Doyen, Mickael Randour, and Jean-François Raskin. Looking at mean payoff and total-payoff through windows. *Information and Computation*, 242, 2015.

[CDT02] Thierry Cachat, Jacques Duparc, and Wolfgang Thomas. Solving pushdown games with a Σ_3 winning condition. In *Proceedings of the EACSL Annual Conference on Computer Science and Logic, CSL'02*, volume 2471. Springer-Verlag, 2002.

[CENR18] Krishnendu Chatterjee, Adrián Elgyütt, Petr Novotný, and Owen Rouillé. Expectation optimization with probabilistic guarantees in POMDPs with discounted-sum objectives. In *International Joint Conference on Artificial Intelligence18*. ijcai.org, 2018.

[CF19] Thomas Colcombet and Nathanaël Fijalkow. Universal graphs and good for games automata: New tools for infinite duration games. In *Proceedings of the International Conference on the Foundations of Software Science and Computational Structures, FoSSaCS'19*, 2019.

[CFGO22] Thomas Colcombet, Nathanaël Fijalkow, Paweł Gawrychowski, and Pierre Ohlmann. The theory of universal graphs for infinite duration games. *Logical Methods in Computer Science*, 18, 2022.

[CH11] Krishnendu Chatterjee and Monika Henzinger. Faster and dynamic algorithms for maximal end-component decomposition and related graph problems in probabilistic verification. In *Proceedings of the Symposium on Discrete Algorithms, SODA'11*. Society for Industrial and Applied Mathematics, 2011.

[CH14] Arnaud Carayol and Matthew Hague. Saturation algorithms for model-checking pushdown systems. In *Proceedings of the International Conference on Automata and Formal Languages, AFL14*, volume 151, 2014.

[Cha13] Jakub Chaloupka. Z-reachability problem for games on 2-dimensional vector addition systems with states is in P. *Fundamenta Informatica*, 123, 2013.

[CHIJ17] Krishnendu Chatterjee, Thomas Dueholm Hansen, and Rasmus Ibsen-Jensen. Strategy complexity of concurrent safety games. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science, MFCS'17*, 2017.

[CHJ05] Krishnendu Chatterjee, Thomas A. Henzinger, and Marcin Jurdziński. Mean-payoff parity games. In *Proceedings of the Annual IEEE Symposium on Logic in Computer Science, LICS'05*. IEEE Computer Society, 2005.

[CHP08] Krishnendu Chatterjee, Thomas A. Henzinger, and Vinayak S. Prabhu. Trading infinite memory for uniform randomness in timed games. In *Proceedings of the International Conference on Hybrid Systems: Computation and Control, HSCC'08*, volume 4981. Springer-Verlag, 2008.

[CHP10] Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. Strategy logic. *Information and Computation*, 208, 2010.

[Chu57] Alonzo Church. Applications of recursive arithmetic to the problem of cricuit synthesis. In *Summaries of the Summer Institute of Symbolic Logic*, volume I. Cornell University, 1957.

[Chu62] Alonzo Church. Logic, arithmetic, and automata. In *Proceedings of the International Congress of Mathematicians, ICM'62*, 1962.

[CIJ15] Krishnendu Chatterjee and Rasmus Ibsen-Jensen. The value 1 problem under finite-memory strategies for concurrent mean payoff games. In *Proceedings of the Symposium on Discrete Algorithms, SODA'15*. Society for Industrial and Applied Mathematics, 2015.

[CJK+17] Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In *Proceedings of the Annual ACM Symposium on Theory of Computing, STOC'17*, 2017.

[CJL+09] Franck Cassez, Jan J. Jensen, Kim G. Larsen, Jean-François Raskin, and Pierre-Alain Reynier. Automatic synthesis of robust and optimal controllers – an industrial case study. In *Proceedings of the International Conference on Hybrid Systems: Computation and Control, HSCC'09*, volume 5469. Springer-Verlag, 2009.

[CJLS17] Thomas Colcombet, Marcin Jurdziński, Ranko Lazić, and Sylvain Schmitz. Perfect half-space games. In *Proceedings of the Annual IEEE Symposium on Logic in Computer Science, LICS'17*. IEEE Computer Society, 2017.

[CKK17] Krishnendu Chatterjee, Zuzana Kretínská, and Jan Kretínský. Unifying two views on multiple mean-payoff objectives in Markov decision processes. *Logical Methods in Computer Science*, 13, 2017.

[CLL+19] Wojciech Czerwiński, Sławomir Lasota, Ranko Lazić, Jérôme Leroux, and Filip Mazowiecki. The reachability problem for Petri nets is not elementary. In *Proceedings of the Annual ACM Symposium on Theory of Computing, STOC'19*. ACM Press, 2019.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Cliffor Stein. *Introduction to Algorithms, Third Edition*. MIT Press, 2009.

[Con92] Anne Condon. The complexity of stochastic games. *Information and Computation*, 96, 1992.

[CR15] Lorenzo Clemente and Jean-François Raskin. Multidimensional beyond worst-case and almost-sure problems for mean-payoff objectives. In *Proceedings of the Annual IEEE Symposium on Logic in Computer Science, LICS'15*. IEEE Computer Society, 2015.

[CR17] Carlo Comin and Romeo Rizzi. Improved pseudo-polynomial bound for the value problem and optimal strategy synthesis in mean payoff games. *Algorithmica*, 77, 2017.

[CRR14] Krishnendu Chatterjee, Mickael Randour, and Jean-François Raskin. Strategy synthesis for multi-dimensional quantitative objectives. *Acta Informatica*, 51, 2014.

[CS14] Jean-Baptiste Courtois and Sylvain Schmitz. Alternating vector addition systems with states. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science, MFCS'14*, volume 8634. Springer-Verlag, 2014.

[CS21] Arnaud Carayol and Olivier Serre. Higher-order recursion schemes and their automata models. In *Handbook of Automata Theory*. European Mathematical Society Publishing House, Zürich, Switzerland, 2021.

[dA97] Luca de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, 1997.

[dAFH$^+$03] Luca de Alfaro, Marco Faella, Thomas A. Henzinger, Rupak Majumdar, and Mariëlle Stoelinga. The element of surprise in timed games. In *Proceedings of the International Conference on Concurrency Theory, CONCUR'03*, volume 2761. Springer-Verlag, 2003.

[dAH00] Luca de Alfaro and Thomas A. Henzinger. Concurrent ω-regular games. In *Proceedings of the Annual IEEE Symposium on Logic in Computer Science, LICS'00*. IEEE Computer Society, 2000.

[dAHK98] Luca de Alfaro, Thomas A. Henzinger, and Orna Kupferman. Concurrent reachability games. In *Proceedings of the Annual Symposium on Foundations of Computer Science, FOCS'98*. IEEE Computer Society, 1998.

[Dan65] George B. Dantzig. *Linear programming and extensions*. Princeton University Press, 1965.

[DGP09] Constantinos Daskalakis, Paul Goldberg, and Christos H. Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, 39, 2009.

[Dic13] Leonard Eugene Dickson. Finiteness of the odd perfect and primitive abundant numbers with *n* distinct prime factors. *American Journal of Mathematics*, 35, 1913.

[Dil90] David L. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Proceedings of the International Workshop on Automatic Verification Methods for Finite State Systems, AVMFSS'89*, volume 407. Springer-Verlag, 1990.

[DJL+14] Alexandre David, Jan J. Jensen, Kim G. Larsen, Axel Legay, Didier Lime, Mathias Grund Sørensen, and Jakob Haahr Taankvist. On time with minimal expected cost! In *Proceedings of the International Symposium on Automated Technology for Verification and Analysis, ATVA'14*, volume 8837. Springer-Verlag, 2014.

[DJL+15] Alexandre David, Jan J. Jensen, Kim G. Larsen, Marius Mikučionis, and Jakob Haahr Taankvist. Uppaal stratego. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'15*, volume 9035. Springer-Verlag, 2015.

[DJL18] Laure Daviaud, Marcin Jurdziński, and Ranko Lazić. A pseudo-quasi-polynomial algorithm for mean payoff parity games. In *Proceedings of the Annual IEEE Symposium on Logic in Computer Science, LICS'18*. ACM Press, 2018.

[DJLL12] Stéphane Demri, Marcin Jurdziński, Oded Lachish, and Ranko Lazić. The covering and boundedness problems for branching vector addition systems. *Journal of Computer and System Sciences*, 79, 2012.

[DJW97] Stefan Dziembowski, Marcin Jurdziński, and Igor Walukiewicz. How much memory is needed to win infinite games? In *Proceedings of the Annual IEEE Symposium on Logic in Computer Science, LICS'97*. IEEE Computer Society, 1997.

[DKQR20] Florent Delgrange, Joost-Pieter Katoen, Tim Quatmann, and Mickael Randour. Simple strategies in multi-objective MDPs. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'20*, volume 12078. Springer, 2020.

[Ehr61] Andrzej Ehrenfeucht. An application of games to the completeness problem for formalized theories. *Comptes-rendus de l'Académie des Sciences*, 49, 1961.

[EJ88] E. Allen Emerson and Charanjit S. Jutla. The complexity of tree automata and logics of programs. In *Proceedings of the Annual Symposium on Foundations of Computer Science, FOCS'88*. IEEE Computer Society, 1988.

[EJ91] E. Allen Emerson and Charanjit S. Jutla. Tree automata, μ-calculus and determinacy. In *Proceedings of the Annual Symposium on Foundations of Computer Science, FOCS'91*. IEEE Computer Society, 1991.

[EJS93] E. Allen Emerson, Charanjit S. Jutla, and A. Prasad Sistla. On model-checking for fragments of μ-calculus. In *Proceedings of the International Conference on Computer Aided Verification, CAV'93*, volume 697. Springer-Verlag, 1993.

[EKVY08] Kousha Etessami, Marta Z. Kwiatkowska, Moshe Y. Vardi, and Mihalis Yannakakis. Multi-objective model checking of Markov decision processes. *Logical Methods in Computer Science*, 4, 2008.

[EM79] Andrzej Ehrenfeucht and Jan Mycielski. Positional strategies for mean payoff games. *International Journal of Game Theory*, 8, 1979.

[EMP10] Rüdiger Ehlers, Robert Mattmüller, and Hans-Jörg Peter. Combining symbolic representations for solving timed games. In *Proceedings of the International Conference on Formal Modelling and Analysis of Timed Systems, FORMATS'10*, volume 6246. Springer-Verlag, 2010.

[Esp98] Javier Esparza. Decidability and complexity of Petri net problems — an introduction. In *Lectures on Petri Nets I: Basic Models*, volume 1491. Springer-Verlag, 1998.

[Eve57] H. Everett. Recursive games. In *Contributions to the Theory of Games III*, volume 39. Princeton University Press, 1957.

[EY10] Kousha Etessami and Mihalis Yannakakis. On the complexity of Nash equilibria and other fixed points. *SIAM Journal on Computing*, 39, 2010.

[Fea10a] John Fearnley. Exponential lower bounds for policy iteration. In *Proceedings of the International Colloquium on Automata, Languages and Programming, ICALP'10*. Springer-Verlag, 2010.

[Fea10b] John Fearnley. *Strategy Iteration Algorithms for Games and Markov Decision Processes*. PhD thesis, University of Warwick, 2010.

[Fea17] John Fearnley. Efficient parallel strategy improvement for parity games. In *Proceedings of the International Conference on Computer Aided Verification, CAV'17*, 2017.

[FF03] Chris Ferguson and Chris Ferguson. On the borel and von neumann poker models. *Game Theory and Applications*, 9, 2003.

[FGMS20] John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul Savani. Unique end of potential line. *Journal of Computer and System Sciences*, 114, 2020.

[Fij18] Nathanaël Fijalkow. An optimal value iteration algorithm for parity games. *CoRR*, abs/1801.09618, 2018.

[FJLS11] Uli Fahrenberg, Line Juhl, Kim G. Larsen, and Jiří Srba. Energy games in multiweighted automata. In *Proceedings of the International Colloquium on Theoretical Aspects of Computing, ICTAC'11*, volume 6916. Springer-Verlag, 2011.

[FJS+17] John Fearnley, Sanjay Jain, Sven Schewe, Frank Stephan, and Dominik Wojtczak. An ordered approach to solving parity games in quasipolynomial time and quasilinear space. In *Proceedings of the International SPIN Workshop on Model Checking of Software, SPIN'17*, 2017.

[FKP12] Vojtech Forejt, Marta Z. Kwiatkowska, and David Parker. Pareto curves for probabilistic model checking. In *Proceedings of the International Symposium on Automated Technology for Verification and Analysis, ATVA'12*, volume 7561. Springer, 2012.

[FM13] Søren Kristoffer Stiil Frederiksen and Peter Bro Miltersen. Approximating the value of a concurrent reachability game in the polynomial time hierarchy. In *Proceedings of the International Symposium on Algorithms and Computation, ISAAC'*, 2013.

[FMR68] Patrick C. Fischer, Albert R. Meyer, and Dinah Rosenberg. Counter machines and counter languages. *Mathematical Systems Theory*, 2, 1968.

[FNS07] Tomas Feder, Hamid Nazerzadeh, and Amin Saberi. Approximating Nash equilibria using small-support strategies. In *Proceedings of the ACM conference on Electronic Commerce, EC'07*. ACM Press, 2007.

[Fra50] Roland Fraïssé. Sur une nouvelle classification des systèmes de relations. In *Comptes Rendus de l'Académie des Sciences*, volume 230, 1950.

[Fra53] Roland Fraïssé. Sur quelques classifications des systèmes de relations. In *Publications Scientifiques de l'Université d'Alger*, 1953.

[Fri11] Oliver Friedmann. An exponential lower bound for the latest deterministic strategy iteration algorithms. *Logical Methods in Computer Science*, 7, 2011.

[FS01] Alain Finkel and Phhilippe Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256, 2001.

[FV96] Jerzy Filar and Koos Vrieze. *Competitive Markov Decision Processes*. Springer-Verlag, 1996.

[FWW97] Alain Finkel, Bernard Willems, and Pierre Wolper. A direct symbolic approach to model checking pushdown systems. *Electronic Notes in Theoretical Computer Science*, 9, 1997.

[Gal58] Tibor Gallai. Maximum-minimum sätze über graphen. *Acta Mathematica Academiae Scientiarum Hungaricae*, 9, 1958.

[Gar07] David J. H. Garling. *Inequalities: A Journey Into Linear Analysis*. Cambridge University Press, 2007.

[GH82] Yuri Gurevich and Leo Harrington. Trees, automata, and games. In *Proceedings of the Annual ACM Symposium on Theory of Computing, STOC'82*. ACM Press, 1982.

[GH08] Hugo Gimbert and Florian Horn. Simple stochastic games with few random vertices are easy to solve. In *Proceedings of the International Conference on the Foundations of Software Science and Computational Structures, FoSSaCS'08*, volume 4962. Springer-Verlag, 2008.

[Gil57] Dean Gillette. Stochastic games with zero stop probability. In *Contributions to the Theory of Games III*, volume 39. Princeton University Press, 1957.

[Gim04] Hugo Gimbert. Parity and exploration games on infinite graphs. In *Proceedings of the EACSL Annual Conference on Computer Science and Logic, CSL'04*, volume 3210. Springer-Verlag, 2004.

[GKK88] Vladimir A. Gurvich, Alexander V. Karzanov, and Leonid G. Khachiyan. Cyclic games and an algorithm to find minimax cycle means in directed graphs. *USSR Computational Mathematics and Mathematical Physics*, 28, 1988.

[GMMW22] Hugo Gimbert, Corto Mascle, Anca Muscholl, and Igor Walukiewicz. Distributed controller synthesis for deadlock avoidance. In *Proceedings of the International Colloquium on Automata, Languages and Programming, ICALP'22*, volume 229. Schloß Dagstuhl, 2022.

[Gon92] Clovis C. Gonzaga. Path-following methods for linear programming. *SIAM Review*, 34, 1992.

[Grä02] Erich Grädel. Model checking games. *Electronic Notes in Theoretical Computer Science*, 67, 2002.

[Gre78] Sheila A. Greibach. Remarks on blind and partially blind one-way multicounter machines. *Theoretical Computer Science*, 7, 1978.

[GS53] David Gale and F.M. Stewart. Infinite games with perfect information. *Annals of Mathematical Studies*, 28, 1953.

[GS09a] Thomas Martin Gawlitza and Helmut Seidl. Games through nested fixpoints. In *Proceedings of the International Conference on Computer Aided Verification, CAV'09*, volume 5643. Springer-Verlag, 2009.

[GS09b] Vincent Gripon and Olivier Serre. Qualitative concurrent stochastic games with imperfect information. In *Proceedings of the International Colloquium on Automata, Languages and Programming, ICALP'09*, volume 5556. Springer, 2009.

[GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke. *Automata, Logics, and Infinite Games: A Guide to Current Research (outcome of a Dagstuhl seminar, February 2001)*, volume 2500. Springer-Verlag, 2002.

[GVF+18] Giuseppe De Giacomo, Moshe Y. Vardi, Paolo Felli, Natasha Alechina, and Brian Logan. Synthesis of orchestrations of transducers for manufacturing. In *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI'18*. AAAI Publications, 2018.

[GZ04] Hugo Gimbert and Wiesław Zielonka. When can you play positionally? In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science, MFCS'04*, volume 3153. Springer-Verlag, 2004.

[GZ05] Hugo Gimbert and Wiesław Zielonka. Games where you can play optimally without any memory. In *Proceedings of the International Conference on Concurrency Theory, CONCUR'05*, volume 3653. Springer-Verlag, 2005.

[Hab97] Peter Habermehl. On the complexity of the linear-time μ-calculus for Petri nets. In *Proceedings of the International Conference on Application and Theory of Petri Nets, Petri nets'97*, volume 1248. Springer-Verlag, 1997.

[Hal07] Nir Halman. Simple stochastic games, parity games, mean payoff games and discounted payoff games are all LP-type problems. *Algorithmica*, 49, 2007.

[Hau14] Felix Hausdorff. Grundzüge der Mengenlehre. *Leipzig*, 1914.

[HD05] Paul Hunter and Anuj Dawar. Complexity bounds for regular games. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science, MFCS'05*, volume 3618. Springer-Verlag, 2005.

[HDJ12] Romain Hollanders, Jean-Charles Delvenne, and Raphaël M. Jungers. The complexity of policy iteration is exponential for discounted Markov decision processes. In *Proceedings of the IEEE Conference on Decision and Control, CDC'12*. IEEE Computer Society, 2012.

[HIJK16] Thomas Dueholm Hansen, Rasmus Ibsen-Jensen, and Michal Koucký. The big match in small space. In *Algorithmic Game Theory*, 2016.

[HIJN18] Thomas Dueholm Hansen, Rasmus Ibsen-Jensen, and Abraham Neyman. The big match with a clock and a bit of memory. In *Proceedings of the ACM conference on Electronic Commerce, EC'18*, 2018.

[HIJPT13] Thomas Dueholm Hansen, Rasmus Ibsen-Jensen, Vladimir V. Podolskii, and Elias Tsigaridas. Patience of matrix games. *Discrete Applied Mathematics*, 2013.

[HK15] Christoph Haase and Stefan Kiefer. The odds of staying on budget. In *Proceedings of the International Colloquium on Automata, Languages and Programming, ICALP'15*, volume 9135. Springer, 2015.

[HKL+11] Thomas Dueholm Hansen, Michal Koucký, Niels Lauritzen, Peter Bro Miltersen, and Elias Tsigaridas. Exact algorithms for solving stochastic games: Extended abstract. In *Proceedings of the Annual ACM Symposium on Theory of Computing, STOC'11*, 2011.

[HKM09] Thomas Dueholm Hansen, Michal Koucký, and Peter Bro Miltersen. Winning concurrent reachability games requires doubly-exponential patience. In *Proceedings of the Annual IEEE Symposium on Logic in Computer Science, LICS'09*, 2009.

[HMZ13] Thomas Dueholm Hansen, Peter Bro Miltersen, and Uri Zwick. Strategy iteration is strongly polynomial for 2-player turn-based stochastic games with a constant discount factor. *Journal of the ACM*, 60, 2013.

[HO09] Matthew Hague and C.-H. Luke Ong. Winning regions of pushdown parity games: A saturation method. In *Proceedings of the International Conference on Concurrency Theory, CONCUR'09*, 2009.

[Hod93] Wilfrid Hodges. *Model theory*, volume 42. Cambridge University Press, 1993.

[Hor08] Florian Horn. Explicit muller games are PTIME. In *Proceedings of the International Conference on the Foundations of Software Technology and Theoretical Computer Science, FSTTCS'08*, volume 2. Schloß Dagstuhl, 2008.

[HP79] John E. Hopcroft and Jean-Jacques Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8, 1979.

[HPRV76] C. J. Himmelberg, T. Parthasarathy, T. E. S. Raghavan, and F. S. Van Vleck. Existence of p-equilibrium and optimal stationary strategies in stochastic games. *Transactions of the American Mathematical Society*, 1976.

[Hun15] Paul Hunter. Reachability in succinct one-counter games. In *Proceedings of the International Workshop on Reachability Problems, RP'15*, volume 9328. Springer-Verlag, 2015.

[IJ12] Rasmus Ibsen-Jensen. *Strategy complexity of two-player, zero-sum games*. PhD thesis, Aarhus University, 2012.

[IJM12] Rasmus Ibsen-Jensen and Peter Bro Miltersen. Solving simple stochastic games with few coin toss positions. In *Proceedings of the European Symposium on Algorithms, ESA'12*, volume 7501. Springer-Verlag, 2012.

[Jan15] Petr Jančar. On reachability-related games on vector addition systems with states. In *Proceedings of the International Workshop on Reachability Problems, RP'15*, volume 9328. Springer-Verlag, 2015.

[JL17] Marcin Jurdziński and Ranko Lazić. Succinct progress measures for solving parity games. In *Proceedings of the Annual IEEE Symposium on Logic in Computer Science, LICS'17*, 2017.

[JLS08] Marcin Jurdziński, François Laroussinie, and Jeremy Sproston. Model checking probabilistic timed automata with one or two clocks. *Logical Methods in Computer Science*, 4, 2008.

[JLS15] Marcin Jurdziński, Ranko Lazić, and Sylvain Schmitz. Fixed-dimensional energy games are in pseudo-polynomial time. In *Proceedings of the International Colloquium on Automata, Languages and Programming, ICALP'15*, volume 9135. Springer-Verlag, 2015.

[JLSØ13] Jan J. Jensen, Kim G. Larsen, Jiří Srba, and Lars Kærlund Østergaard. Local model checking of weighted CTL with upper-bound constraints. In *Proceedings of the International SPIN Workshop on Model Checking of Software, SPIN'13*, volume 7976. Springer-Verlag, 2013.

[JMT22] Marcin Jurdziński, Rémi Morvan, and K. S. Thejaswini. Universal algorithms for parity games and nested fixpoints. In *Principles of Systems Design - Essays Dedicated to Thomas A. Henzinger on the Occasion of His 60th Birthday*, volume 13660. Springer-Verlag, 2022.

[JOS18] Petr Jančar, Petr Osička, and Zdeněk Sawa. EXPSPACE-complete variant of countdown games, and simulation on succinct one-counter nets. In *Proceedings of the International Workshop on Reachability Problems, RP'18*, volume 11123. Springer-Verlag, 2018.

[JPZ08] Marcin Jurdziński, Mike Paterson, and Uri Zwick. A deterministic subexponential algorithm for solving parity games. *SIAM Journal on Computing*, 38, 2008.

[Jur00] Marcin Jurdziński. Small progress measures for solving parity games. In *Proceedings of the International Symposium on Theoretical Aspects of Computer Science, STACS'00*, 2000.

[Kan95] Max I. Kanovich. Petri nets, Horn programs, linear logic and vector games. *Annals of Pure and Applied Logic*, 75, 1995.

[Kan16] Max I. Kanovich. The undecidability theorem for the Horn-like fragment of linear logic (revisited). *Mathematical Structures in Computer Science*, 26, 2016.

[Kar78] Richard M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23, 1978.

[Kar84] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *combinatorica*, 4, 1984.

[KBB+08] Leonid G. Khachiyan, Endre Boros, Konrad Borys, Khaled Elbassioni, Vladimir A. Gurvich, Gabor Rudolf, and Jihui Zhao. On short paths interdiction problems: Total and node-wise limited interdiction. *Theory of Computing Systems*, 43, 2008.

[Kha79] Leonid G. Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20, 1979.

[Kie13] Stefan Kiefer. BPA bisimilarity is EXPTIME-hard. *Information Processing Letters*, 113, 2013.

[KL22] Zhuan Khye Koh and Georg Loho. Beyond value iteration for parity games: Strategy iteration with universal trees. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science, MFCS'22*, volume 241. Schloß Dagstuhl, 2022.

[KM03] Stephen Kwek and Kurt Mehlhorn. Optimal search for rationals. *Information Processing Letters*, 86, 2003.

[KMvS94] Daphne Koller, Nimrod Megiddo, and Bernhard von Stengel. Fast algorithms for finding randomized strategies in game trees. In *Proceedings of the Annual ACM Symposium on Theory of Computing, STOC'94*. ACM Press, 1994.

[Koh74] Elon Kohlberg. Repeated games with absorbing states. *Annals of Mathematical Statistics*, 1974.

[Kön36] Dénes König. *Theorie der endlichen und unendlichen Graphen*. Akademische Verlagsgesellschaft Leipzig, 1936.

[Kop06] Eryk Kopczyński. Half-positional determinacy of infinite games. In *Proceedings of the International Colloquium on Automata, Languages and Programming, ICALP'06*, volume 4052. Springer-Verlag, 2006.

[Kop08] Eryk Kopczyński. *Half-positional Determinacy of Infinite Games*. PhD thesis, University of Warsaw, 2008.

[Kos82] S. Rao Kosaraju. Decidability of reachability in vector addition systems. In *Proceedings of the Annual ACM Symposium on Theory of Computing, STOC'82*. ACM Press, 1982.

[Koz21] Alexander Kozachinskiy. Continuous positional payoffs. In *Proceedings of the International Conference on Concurrency Theory, CONCUR'21*, volume 203. Schloß Dagstuhl, 2021.

[KPR18] Jan Kretínský, Guillermo A. Pérez, and Jean-François Raskin. Learning-based mean-payoff optimization in an unknown MDP under omega-regular constraints. In *Proceedings of the International Conference on Concurrency Theory, CONCUR'18*, volume 118. Schloß Dagstuhl, 2018.

[Kru72] Joseph B. Kruskal. The theory of well-quasi-ordering: A frequently discovered concept. *Journal of Combinatorial Theory, Series A*, 13, 1972.

[KS76] John G. Kemeny and J. Laurie Snell. *Finite Markov Chains*. Springer-Verlag, 1976.

[KS88] S. Rao Kosaraju and Gregory Sullivan. Detecting cycles in dynamic graphs in polynomial time. In *Proceedings of the Annual ACM Symposium on Theory of Computing, STOC'88*. ACM Press, 1988.

[Kuč11] Antonín Kučera. Turn-based stochastic games. In *Lectures in Game Theory for Computer Scientists*. Cambridge University Press, 2011.

[KV00] Orna Kupferman and Moshe Y. Vardi. An automata-theoretic approach to reasoning about infinite-state systems. In *Proceedings of the International Conference on Computer Aided Verification, CAV'00*, volume 1855. Springer, 2000.

[Lad75] Richard E. Ladner. The Circuit Value Problem is Log Space Complete for PTIME. *SIGACT News*, 7, 1975.

[Lam92] Jean-Luc Lambert. A structure to decide reachability in Petri nets. *Theoretical Computer Science*, 99, 1992.

[Lan67] Lawrence H. Landweber. Finite state games–a solvability algorithm for restricted second-order arithmetic. *Notices of the American Mathematical Society*, 14, 1967.

[Leh18] Karoliina Lehtinen. A modal-μ perspective on solving parity games in quasi-polynomial time. In *Proceedings of the Annual IEEE Symposium on Logic in Computer Science, LICS'18*, 2018.

[Ler11] Jérôme Leroux. Vector addition system reachability problem: a short self-contained proof. In *Proceedings of the Annual ACM Symposium on Principles of Programming Languages, POPL'11*. ACM Press, 2011.

[Lip76] Richard J. Lipton. The reachability problem requires exponential space. Technical report, Yale University, 1976.

[LL69] Thomas M. Liggett and Steven A Lippman. Stochastic games with perfect information and time average payoff. *SIAM Review*, 11, 1969.

[LLTW14] Kim G. Larsen, Axel Legay, Louis-Marie Traonouez, and Andrzej Wąsoswki. Robust synthesis for real-time systems. *Theoretical Computer Science*, 515, 2014.

[LMSS92] Patrick Lincoln, John Mitchell, Andre Scedrov, and Narayan Shankar. Decision problems for propositional linear logic. *Annals of Pure and Applied Logic*, 56, 1992.

[LP07] Yuri M. Lifshits and Dmitri S. Pavlov. Potential theory for mean payoff games. *Journal of Mathematical Sciences*, 145, 2007.

[LPSW22] Karoliina Lehtinen, Paweł Parys, Sven Schewe, and Dominik Wojtczak. A recursive approach to solving parity games in quasipolynomial time. *Logical Methods in Computer Science*, 18, 2022.

[LS98] Xinxin Liu and Scott A. Smolka. Simple linear-time algorithms for minimal fixed points. In *Proceedings of the International Colloquium on Automata, Languages and Programming, ICALP'98*, volume 1443. Springer-Verlag, 1998.

[LS19] Ranko Lazić and Sylvain Schmitz. The ideal view on Rackoff's coverability technique. *Information and Computation*, 2019.

[LWG13] Dominique Larchey-Wendling and Didier Galmiche. Nondeterministic phase semantics and the undecidability of Boolean BI. *ACM Transactions on Computational Logic*, 14, 2013.

[LY94] Richard J. Lipton and Neal E. Young. Simple strategies for large zero-sum games with applications to complexity theory. In *Proceedings of the Annual ACM Symposium on Theory of Computing, STOC'94*. ACM Press, 1994.

[Mar75] George R.R. Martin. Borel determinacy. *Annals of Mathematics*, 102, 1975.

[Mat07] Jiří Matoušek. *Understanding and Using Linear Programming*. Springer-Verlag, 2007.

[May81] Richard Mayr. An algorithm for the general Petri net reachability problem. In *Proceedings of the Annual ACM Symposium on Theory of Computing, STOC'81*. ACM Press, 1981.

[McN65] Robert McNaughton. Finite-state infinite games. Technical report, Massachusetts Institute of Technology, 1965.

[McN66] Robert McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Computation*, 9, 1966.

[McN93] Robert McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65, 1993.

[Min67] Marvin L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall, 1967.

[MMPV12] Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y. Vardi. What makes ATL* decidable? a decidable fragment of strategy logic. In *Proceedings of the International Conference on Concurrency Theory, CONCUR'12*, 2012.

[MN81] Jean-François Mertens and Abraham Neyman. Stochastic games. *International Journal of Game Theory*, 10, 1981.

[Mos84] Andrzej W. Mostowski. Regular expressions for infinite trees and a standard form of automata. In *Computation Theory*, 1984.

[Mos91] Andrzej W. Mostowski. Games with forbidden positions. Technical report, University of Gdansk, 1991.

[MPS95] Oded Maler, Amir Pnueli, and Joseph Sifakis. On the synthesis of discrete controllers for timed systems (an extended abstract). In *Proceedings of the International Symposium on Theoretical Aspects of Computer Science, STACS'95*, volume 900. Springer-Verlag, 1995.

[MR22] James C. A. Main and Mickael Randour. Different strokes in randomised strategies: Revisiting Kuhn's theorem under finite-memory assumptions. In *Proceedings of the International Conference on Concurrency Theory, CONCUR'22*, volume 243. Schloß Dagstuhl, 2022.

[MRS21] James C. A. Main, Mickael Randour, and Jeremy Sproston. Time flies when looking out of the window: Timed games with window parity objectives. In *Proceedings of the International Conference on Concurrency Theory, CONCUR'21*, volume 203. Schloß Dagstuhl, 2021.

[MS98] Ashok P. Maitra and William D. Sudderth. Finitely additive stochastic games with borel measurable payoffs. *International Journal of Game Theory*, 27, 1998.

[Mul63] David E. Muller. Infinite sequences and finite machines. In *Proceedings of the Fourth Annual Symposium on Switching Circuit Theory and Logical Design*. IEEE Computer Society, 1963.

[MW03] Allen B. MacKenzie and Stephen B. Wicker. Stability of multipacket slotted aloha with selfish users and perfect information. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, 2003.

[Nas50] John F. Nash. Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 36, 1950.

[Nor98] J.R. Norris. *Markov Chains*, volume 2. Cambridge University Press, 1998.

[Nov15] Petr Novotný. *Controller Synthesis for Resource-Aware Systems*. PhD thesis, Masaryk University, 2015.

[NPR16] Reino Niskanen, Igor Potapov, and Julien Reichert. Undecidability of two-dimensional robot games. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science, MFCS'16*, volume 58. Schloß Dagstuhl, 2016.

[Ohl21] Pierre Ohlmann. *Monotonic graphs for parity and mean-payoff games. (Graphes monotones pour jeux de parité et à paiement moyen)*. PhD thesis, Université Paris Cité, 2021.

[Ohl22] Pierre Ohlmann. The GKK algorithm is the fastest over simple mean-payoff games. In *Proceedings of the International Computer Science Symposium in Russia, CSR'22*, volume 13296. Springer, 2022.

[Ohl23] Pierre Ohlmann. Characterizing positionality in games of infinite duration over infinite graphs. *TheoretiCS*, 2, 2023.

[ORS14] Youssouf Oualhadj, Pierre-Alain Reynier, and Ocan Sankur. Probabilistic robust timed games. In *Proceedings of the International Conference on Concurrency Theory, CONCUR'14*, volume 8704. Springer-Verlag, 2014.

[Par73] T. Parthasarathy. Discounted, positive, and non-cooperative stochastic games. *International Journal of Game Theory*, 1973.

[Par19] Paweł Parys. Parity games: Zielonka's algorithm in quasi-polynomial time. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science, MFCS'19*, 2019.

[Par20] Paweł Parys. Parity games: Another view on lehtinen's algorithm. In *Proceedings of the EACSL Annual Conference on Computer Science and Logic, CSL'20*, 2020.

[PEM11] Hans-Jörg Peter, Rüdiger Ehlers, and Robert Mattmüller. Synthia: Verification and synthesis for timed automata. In *Proceedings of the International Conference on Computer Aided Verification, CAV'11*, volume 6806. Springer-Verlag, 2011.

[Pet62] Carl Adam Petri. *Kommunikation mit Automaten*. PhD thesis, Universität Bonn, 1962.

[Pin21] Jean-Éric Pin. *Handbook of Automata Theory*. European Mathematical Society Publishing House, Zürich, Switzerland, 2021.

[PR90] Amir Pnueli and Roni Rosner. Distributed reactive systems are hard to synthesize. In *Proceedings of the Annual Symposium on Foundations of Computer Science, FOCS'31*, 1990.

[Pur95] Anuj Puri. *Theory of Hybrid Systems and Discrete Event Systems*. PhD thesis, University of California at Berkeley, 1995.

[Put05] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2005.

[Rab69a] Michael O. Rabin. Automata on infinite objects and Church's problem. *Transactions of the American Mathematical Society*, 141, 1969.

[Rab69b] Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141, 1969.

[Rac78] Charles Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6, 1978.

[Ran14] Mickael Randour. *Synthesis in Multi-Criteria Quantitative Games*. PhD thesis, UMONS – Université de Mons, 2014.

[Rei84] John H. Reif. The complexity of two-player games of incomplete information. *Journal of Computer and System Sciences*, 29, 1984.

[Ren09] Jérôme Renault. Repeated games with incomplete information. In *Encyclopedia of Complexity and Systems Science*. Springer, 2009.

[Ros06] Jeffrey S. Rosenthal. *A First Look at Rigorous Probability Theory*. World Scientific Publishing, 2006.

[RP18] Stéphane Le Roux and Arno Pauly. Extending finite-memory determinacy to multi-player games. *Information and Computation*, 2018.

[RPR18] Stéphane Le Roux, Arno Pauly, and Mickael Randour. Extending finite-memory determinacy by Boolean combination of winning conditions. In *Proceedings of the International Conference on the Foundations of Software Technology and Theoretical Computer Science, FSTTCS'18*, volume 122. Schloß Dagstuhl, 2018.

[RRS14] Mickael Randour, Jean-François Raskin, and Ocan Sankur. Variations on the stochastic shortest path problem. In *Proceedings of the International Conference on Verification, Model Checking, and Abstract Interpretation, VMCAI'15*, volume 8931. Springer-Verlag, 2014.

[RRS17] Mickael Randour, Jean-François Raskin, and Ocan Sankur. Percentile queries in multi-dimensional Markov decision processes. *Formal Methods for System Design*, 50, 2017.

[RSB05] Jean-François Raskin, Mathias Samuelides, and Laurent Van Begin. Games for counting abstractions. In *Proceedings of the International Workshop on Automated Verification of Critical Systems, AVoCS'04*, volume 128. Elsevier, 2005.

[RSV06] Dinah Rosenberg, Eilon Solan, and Nicolas Vieille. Stochastic games with imperfect monitoring. In *Advances in Dynamic Games: Applications to Economics, Management Science, Engineering, and Environmental Management*. Birkhäuser Boston, 2006.

[Sav70] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4, 1970.

[SB18] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[SE84] Robert S. Streett and E. Allen Emerson. The propositional mu-calculus is elementary. In *Proceedings of the International Colloquium on Automata, Languages and Programming, ICALP'84*, 1984.

[SE89] Robert S. Streett and E. Allen Emerson. An automata theoretic decision procedure for the propositional mu-calculus. *Information and Computation*, 81, 1989.

[Ser03] Olivier Serre. Note on winning positions on pushdown games with ω-regular conditions. *Information Processing Letters*, 85, 2003.

[Ser04] Olivier Serre. *Contribution à l'étude des jeux sur des graphes de processus à pile*. PhD thesis, Université Paris 7, 2004.

[Ser06a] Olivier Serre. Games with winning conditions of high borel complexity. *Theoretical Computer Science*, 350, 2006.

[Ser06b] Olivier Serre. Parity games played on transition graphs of one-counter processes. In *Proceedings of the International Conference on the Foundations of Software Science and Computational Structures, FoSSaCS'06*, volume 3921. Springer-Verlag, 2006.

[Sha53] Lloyd S. Shapley. Stochastic games. In *Proceedings of the National Academy of Science*, volume 39, 1953.

[Sko21] Mateusz Skomra. Optimal bounds for bit-sizes of stationary distributions in finite Markov chains. Technical report, ArXiv, 2021.

[SP73] John Maynard Smith and George R. Price. The logic of animal conflict. *Nature*, 246, 1973.

[SS12] Sylvain Schmitz and Phhilippe Schnoebelen. Algorithmic aspects of wqo theory. Lecture notes, European Summer School on Logic, Language and Information, ESSLLI'12, 2012.

[Str81] Robert S. Streett. A propositional dynamic logic of looping and converse. Technical report, Massachusetts Institute of Technology, 1981.

[SW89] Colin Stirling and David Walker. Local model checking in the modal mu-calculus. In *Proceedings of the International Joint Conference on Theory and Practice of Software Development, TAPSOFT'89*, 1989.

[TA99] Stavros Tripakis and Karine Altisen. On-the-fly controller synthesis for discrete and dense-time systems. In *Proceedings of the World Congress on Formal Methods, FM'99*, volume 1708. Springer-Verlag, 1999.

[Tho95] Wolfgang Thomas. On the synthesis of strategies in infinite games. In *Proceedings of the International Symposium on Theoretical Aspects of Computer Science, STACS'95*, volume 900. Springer-Verlag, 1995.

[Tho97] Wolfgang Thomas. Languages, automata, and logic. In *Handbook of Formal Languages, Volume 3: Beyond Words*. Springer-Verlag, 1997.

[Tho09] Wolfgang Thomas. Facets of synthesis: Revisiting Church's problem. In *Proceedings of the International Conference on the Foundations of Software Science and Computational Structures, FoSSaCS'09*, 2009.

[Tse90] Paul Tseng. Solving H-horizon, stationary Markov decision problems in time proportional to $\log(H)$. *Operation Research Letters*, 9, 1990.

[Urq99] Alasdair Urquhart. The complexity of decision procedures in relevance logic II. *Journal of Symbolic Logic*, 64, 1999.

[Var98] Moshe Y. Vardi. Reasoning about the past with two-way automata. In *Proceedings of the International Colloquium on Automata, Languages and Programming, ICALP'98*, volume 1443. Springer, 1998.

[VCD+15] Yaron Velner, Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, Alexander Rabinovich, and Jean-François Raskin. The complexity of multi-mean payoff and multi-energy games. *Information and Computation*, 241, 2015.

[Vel15] Yaron Velner. Robust multidimensional mean-payoff games are undecidable. In *Proceedings of the International Conference on the Foundations of Software Science and Computational Structures, FoSSaCS'15*, volume 9034. Springer-Verlag, 2015.

[VJ00] Jens Vöge and Marcin Jurdziński. A discrete strategy improvement algorithm for solving parity games. In *Proceedings of the International Conference on Computer Aided Verification, CAV'00*, 2000.

[vNM44] John von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

[Wag79] Klaus W. Wagner. On omega-regular sets. *Information and Computation*, 43, 1979.

[Wal96] Igor Walukiewicz. Pushdown processes: Games and model checking. In *Proceedings of the International Conference on Computer Aided Verification, CAV'96*, volume 1102. Springer-Verlag, 1996.

[Wal01] Igor Walukiewicz. Pushdown processes: Games and model-checking. *Information and Computation*, 164, 2001.

[Wal02] Igor Walukiewicz. Monadic second-order logic on tree-like structures. *Theoretical Computer Science*, 275, 2002.

[Wil91] David Williams. *Probability with Martingales*. Cambridge University Press, 1991.

[Ye11] Yinyu Ye. The simplex and policy-iteration methods are strongly polynomial for the Markov decision problem with a fixed discount rate. *Mathematics of Operations Research*, 36, 2011.

[Zer13] Ernst Zermelo. Uber eine anwendung der mengenlehre auf die theorie des schachspiels. In *Proceedings of the International Congress of Mathematicians, ICM'13*, volume 2. Cambridge University Press, 1913.

[Zie98] Wiesław Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200, 1998.

[ZP96] Uri Zwick and Mike Paterson. The complexity of mean payoff games. *Theoretical Computer Science*, 158, 1996.