Towards Synthesizing Open Systems: Tableaux For Multi-Agent Temporal Logics

Amélie David



Université Paris-Saclay, Technical University of Denmark

École doctorale STIC N°580
Sciences et technologies de l'information et de la communication
Laboratoire IBISC – Équipe COSMO

THÈSE

préparée à l'Université d'Évry-Val d'Essonne,
présentée et soutenue publiquement le 30 septembre 2015
pour l'obtention du grade de

Docteur de l'Université Paris-Saclay

Discipline ou Spécialité: Informatique

par

Amélie David

TOWARDS SYNTHESIZING OPEN SYSTEMS: TABLEAUX FOR MULTI-AGENT TEMPORAL LOGICS

COMPOSITION DU JURY

Président:	M. MARCHÉ Claude	Université Paris-Sud
Directeurs:	Mme CERRITO Serenella	Université d'Évry Val d'Essonne
	M. GORANKO Valentin	Technical University of Denmark
Rapporteurs:	M. GORÉ Rajeev	The Australian National University
	M. DEMRI Stéphane	LSV, CNRS & ENS de Cachan
Examinateurs:	M. DIMA Catalin	Université Paris-Créteil
	M. POMMEREAU Franck	Université d'Évry Val d'Essonne

TABLE OF CONTENTS

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

Several years ago, Hugo[1] was at the train station, when a guy, let us call him Bob, asked to help him cheat the French railway company. Hugo accepted to do it. Note that we do not approve this behaviour, but it is a good introductory case for this thesis. What had been asked from Hugo was that he logged into a booking automaton with a given booking number in order to print the ticket. At the same time, the cheater also logged to another booking automaton with the same booking number, and asked for the reimbursement of the ticket. And it worked perfectly, the cheater managed to get both the ticket travel and the reimbursement. Clearly, this unwilling behaviour of the booking system is not possible when using only one automaton. Indeed, this automaton seems to have been designed as follows: a user enters a booking number on the automaton and gets access to his ticket. The user can then print the ticket, which disables the booking number, or, after being logged, the user can ask the reimbursement of the ticket, which also disables the booking number, as depicted in Figure 1.1.

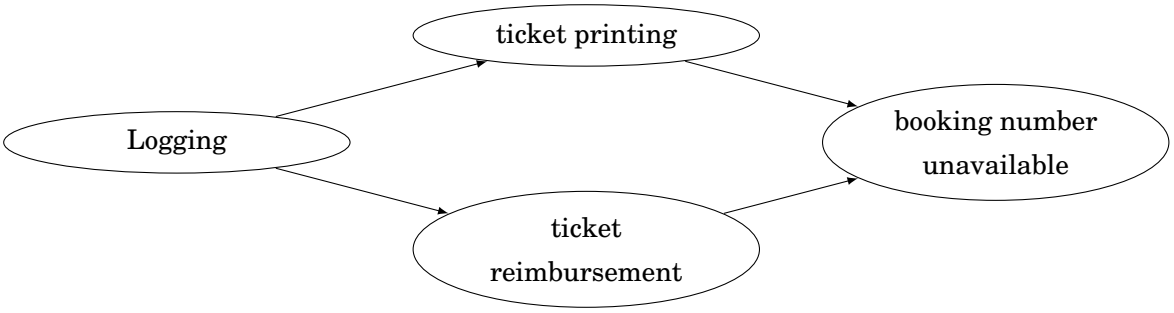


Figure 1.1: Partial model of the French railway company's booking automaton

[1]Name has been changed to preserve the protagonist's anonymity.

But in this design, what has not been taken into account is the fact that the booking automaton is not the only one with the same functionalities in the station. This is a typical case of *multi-agent systems*, where the behaviour of the system depends on the behaviour of the components of the system. A special case of multi-agent systems are *open systems*, that is systems that interact with their environment. In this case, the environment can be considered as an agent. In our introductory case, the damage was not very important: the French railway company has lost a few hundred Euros. But on more critical systems, the consequences of such a bug can be disastrous. Given the generalisation of mobile terminals, such as smartphones or tablets, the number of applications based on open systems increases, and therefore the risk of severe bugs increases. This thesis is about designing safe open systems. This field of research is very wide, since it refers to both multi-agent systems and formal verification. In this introduction, we give some background about open systems and multi-agent systems, temporal logics, verification and the satisfiability problem.

1.1 Open Systems / Multi-Agent Systems

Until the early 80's, computer science was mainly dealing with closed systems, that is systems whose behaviour only depends on the system itself. The behaviour of such systems is entirely controlled and there are no interactions with the environment. A typical example of closed system is, for instance, a vehicle control system.

Open systems interact with their environment, so the system (or the components of the system) can be seen as an agent evolving in an environment. We will see with ATL that the environment is also considered as an agent. Therefore the notion of open system is strongly linked to the notion of multi-agent systems.

There exist various kinds of multi-agent systems used in various domains such as economy [5, 34], social science [6], ecology [45, 49], epidemiology [59], robotics [12, 43], and of course, computer science. Nevertheless, the common point between all multi-agent systems is that their agents interact with their environment in order to achieve their objectives, where the environment may be constituted by other agents. For that purpose, agents have to resolve their *internal choices* no matter how the environment behaves, that is whichever way the environment resolves the *external choices* [4].

Multi-agent systems can be more or less complex. In [67], M. Wooldrigde explains that the complexity of a multi-agent system is linked to the "complexity of the action selection process" by an agent, which is "affected by a number of different environmental properties". These properties have been first described in [60]. Following this classification, we then describe the environment of the multi-agent systems in which we are interested in this thesis as:

- *fully accessible*: agents have a complete information about the current state of the environment and in our case, the current state of the system.

- *non-deterministic*: an action of an agent does not necessary have a single guaranteed effect. The result depends on the action of the other agents and on the environment.
- *non-episodic*: the current decisions made by agents are likely to affect future decisions.
- *static*: The environment stays unchanged during the decision process of the agent.
- *discrete*: the treatment of time is discrete. The number of distinct states of the environment and the number of distinct actions are finite.

Agents themselves can be classified depending on their relation with the environment: *cognitive agents* or *reactive agents*. Cognitive agents have the ability to reason about the world in which they are evolving, and by consequence, each agent can be asked complex tasks that it can solve by itself. On the contrary, reactive agents, as the name suggests, react to stimuli or to the state of the system and the environment. This classification of agents can be refined by their behaviour as in [25] and leads to the Table 1.1. The behaviour of agents is said to be *teleonomic* if the source of agents' motivation comes from the agents themselves, whereas the behaviour of agents is said to be *reflex* if the source of motivation comes from the environment.

In this thesis, we are interested in reactive agents.

Relation with environment / Behaviours	cognitive agents	reactive agents
teleonomic	intentional agents	instinctual agents
reflex/reaction	"module" agents	tropic agents

Table 1.1: Classification of agents depending on their relation with the environment and their behaviour [25]

Finally, we can say that agents have different ways to interact with each other. In [25], J. Ferber makes a distinction between situations where the goals of the agents are compatible: collaboration, coordination; and situations where the goals of the agents are incompatible: competition or conflict. Situations are also affected by the available resources: are there enough resources for all agents?, and by the competences owned by agents: is an agent able to achieve its goal by itself? In our introductory example, we have seen that the goals of Bob and Hugo are compatible, and that there are enough resources, but that Bob is not able to achieve its goal alone. This is why Bob needs to cooperate with Hugo to get both the ticket and his reimbursement.

Properties of the multi-agent systems in which we are interested can easily be catched by temporal logics. The temporal logic ATL was precisely introduced for that purpose in 1997, to specify and verify open systems [3].

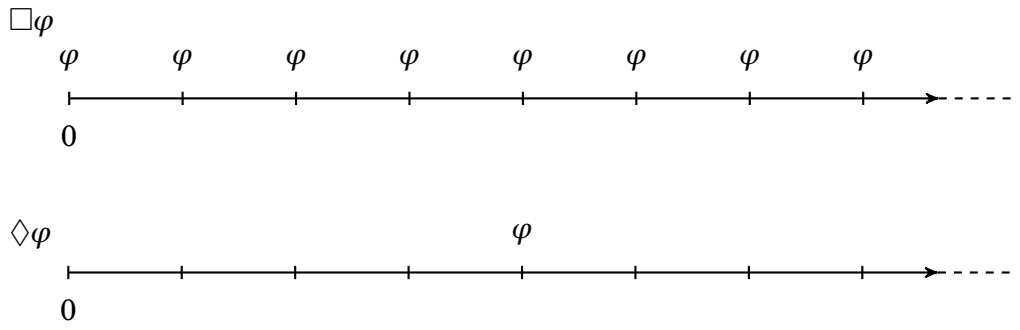


Figure 1.2: The operators "always" □ and "eventually" ◊

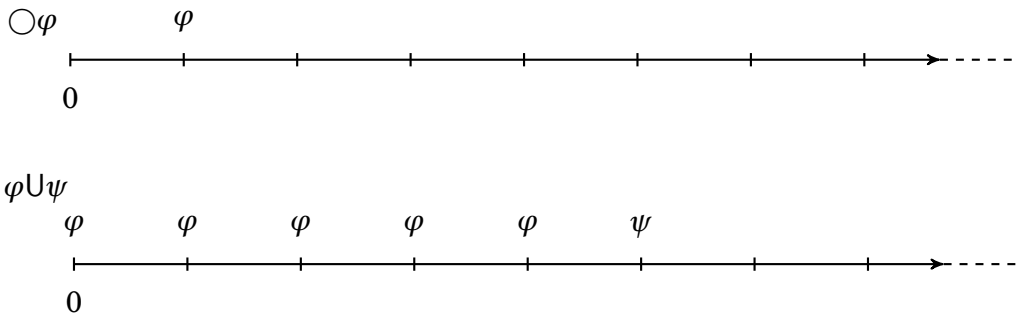


Figure 1.3: The operators *Next* ◯ and *Until* U

1.2 Temporal Logics and ATL

Temporal logics are variants of modal logics where the operators □ (*Necessarily*) and ◊ (*Possibly*) have received different meanings reflecting the notion of linearity of time, that is *Always* and *Sometime*, respectively. In temporal logics, when we write $\Box\varphi$, we mean that φ is true at all moments in the future including the current state, and when we write $\Diamond\varphi$, we mean that there is a moment, which can be the current state, where φ is true, as represented in Figure 1.2. This modification of meanings was introduced by A.N. Prior in 1957 [54] and is considered as the start of modern temporal logics. A chronology of the introduction of the other temporal operators is given in [39]. In 1965, G. von Wright introduced the ancestor of the unary operator *Next* [64], whose symbol is ◯, which was a binary operator called *And Next*. Then, in 1968, H.W. Kamp introduced the binary temporal operator *Until* [36], whose symbol is U. Today, when we write $\bigcirc\varphi$, we mean that φ is true at the next moment and when we write $\varphi \mathsf{U} \psi$, we mean that there is a moment, which can be the current state, where ψ is true and φ is true at all the preceding moments, as represented in Figure 1.3. While during that period, different logics based on these temporal operators appeared in the literature, the most important one, still intensively studied, is the linear temporal logic, in short LTL. It is not very clear when LTL was first defined as we know it today. Indeed, in [51], which is considered as the foundation paper for LTL, the operations

Next and *Until* are absent. Even in [38, 52, 53], while the operator *Next* is used, the operator *Until* is still missing.

Temporal linear logics, including LTL, allow one to define properties on a sequence (or a path) of states, so that these logics and most particularly LTL, have soon been employed with the idea of verifying programs, first in [11] by R. Burstall and then in [51, 52] by A. Pnueli.

It is worth noticing that, with LTL, it is possible to express properties that we want to be true in every possible execution of a program. But, in the early 1980's, the idea of expressing temporal properties that can be true only on some execution of the program appeared, and we then talked about temporal branching logics. The most famous representative of this kind of logics is CTL introduced in 1981 by E. Clarke and E. Emerson [18], which allows to combine a *path quantifier* E, meaning "there exists a path such that", or A, meaning "all paths are such that", and a temporal operator. However, CTL was not the first branching temporal logic; we can mention its ancestor CTF, also introduced by E. Emerson in [23], where fixed point characterisations of temporal operators are also defined. In 1986, E. Clarke and E. Emerson then proposed a more powerful version of CTL, namely CTL* [24], which combines features from CTL and LTL.

In 1997 [3] and again in 2002 [4], a new paradigm for branching temporal logic is proposed by R. Alur, T. Henzinger and O. Kupferman in order to described reactive open systems and multi-agent systems. They introduced both a way to model multi-agent systems, namely *concurrent game models* and a logic to describe properties on these models, namely ATL and its extension ATL*. concurrent game models are transition systems where each transition to a unique successor state results from the combination of actions chosen by all the agents (components and/or environment) of the system. ATL and ATL* can be seen as a generalisation of CTL and CTL*, respectively, where the notion of path quantifiers is replaced by the notion of *strategic quantifiers*. These strategic quantifiers are $\langle\langle A\rangle\rangle$ and $[\![A]\!]$, where A is a coalition of agents. The expression $\langle\langle A\rangle\rangle F$ means that the coalition A has a strategy to enforce the property F, while $[\![A]\!]F$ means that the coalition A cannot cooperate to make the property F false, that is, the coalition A cannot avoid the property F. It is worth noticing that every CTL (respectively CTL*) formula can be written in ATL (respectively ATL*), by considering only one agent a, then the path quantifier A is equivalent to $\langle\langle\emptyset\rangle\rangle$ and the path quantifier E is equivalent to $\langle\langle\{a\}\rangle\rangle$. The inclusions of the logics LTL, CTL, CTL *, ATL and ATL* are illustrated in Figure 1.4.

Around that time, M. Pauly introduced the *coalition logic* in order to make a link between logic and game theory, and also to show that the coalition logic could be used to specify and verify social choice procedures [47]. The coalition logic is strictly included in ATL, in fact, coalition logic can be written in ATL using only the temporal operator $\bigcirc$ ($\Diamond$ and $\Box$ are not necessary).

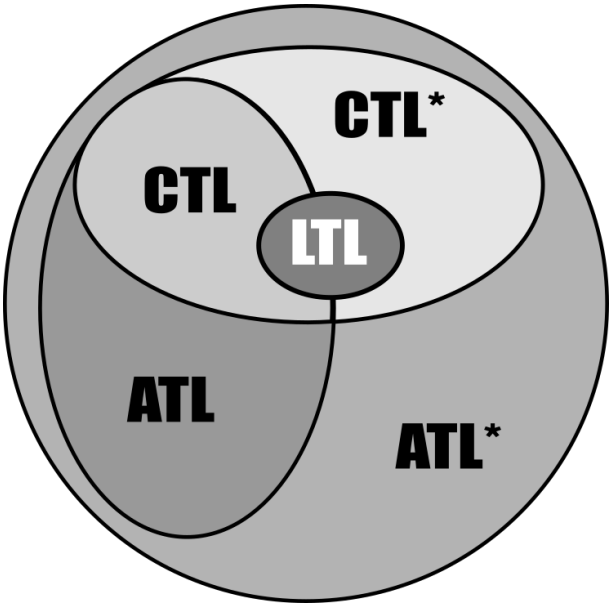


Figure 1.4: Different temporal logics

1.3 Verification of Multi-Agent Systems Using ATL

The objective of the verification of systems in general, and multi-agent systems in particular, is to obtain the safest possible systems, and to avoid problems during utilization of the system such as the one seen in our introductory case, or in more critical systems in aeronautics for instance. In order to achieve this, we first need **a)** to model the system that we want to create or that we have already created, and then **b)** to define the properties we want the system to have in an adequate language in order to obtain a specification. We have seen several logics in the previous section that can describe different kinds of systems and different kinds of temporal properties.

Next, we need some, preferably automated, tools to check that the model that we have designed respects the given specification. This corresponds to the method called *model checking*, which was developed simultaneously by E. Emerson and E. Clarke in the USA [18, 23] and by J.P. Queille and J. Sifakis in France [55, 62]. It is worth noticing that the expression "model checking" appeared for the first time in [18]. These works are at the origin of an important field of research in computer science, which is still very dynamic, and in 2007, E. Clarke, E. Emerson and J. Sifakis received the Turing Award "for their role in developing model checking into a highly effective verification technology"[2].

A model checking algorithm for ATL is given in [4], and runs in polynomial time. However, the model checking problem for ATL* is 2EXPTIME-complete [4], which is the same complexity as that of the satisfiability problem for ATL* [61], which is described in next section.

Therefore, for ATL*, it might be interesting to see the problem of designing safe systems the

[2]A.M. Turing Award: `http://amturing.acm.org/award_winners/clarke_1167964.cfm`

other way round. That is, instead of verifying a specification on an already existing model, why not create a model directly from the specification, which ensures that the specification is met? This second method is called *model synthesis* [40, 50]. A way to transform a specification into a model is to use constructive procedures to solve the satisfiability problem, such as tableau-based decision procedures. With this method, we kill two birds with one stone, since we can check that there is at least one model for the given specification, and if it is the case, we can obtain one of these models.

1.4 The Satisfiability Problem

The notion of satisfiability has existed informally from the beginning of the history of logic with Aristotle (384–322 B.C.) to the 1930's and 1940's when A. Tarski clearly enunciated the difference between syntax and semantics and then worked out a precise notion of satisfiability. Indeed, satisfiability is the semantic version of consistency: $\{p_1,\ldots,p_n\}$ is consistent iff $\{p_1,\ldots,p_n\}$ is satisfiable.

> A set of formulae is said to be satisfiable if there is some structure in which all its component formulae are true: that is, $\{p_1,\ldots,p_n\}$ is satisfiable if and only if, for some structure $\mathscr{A}$, $\mathscr{A} \models p_1$ and ... and $\mathscr{A} \models p_n$. [27]

The dual problem of satisfiability is validity:

> Intuitively, an argument is valid if whenever the premises are true, so is the conclusion. More precisely, the argument from $p_1,\ldots,p_n$ to q is valid (in symbols, $p_1,\ldots,p_n \models q$) if and only if, for all structures $\mathscr{A}$, if $\mathscr{A} \models p_1,\ldots \mathscr{A} \models p_n$, then $\mathscr{A} \models q$. [27]

Therefore, the argument from $p_1,\ldots,p_n$ to q is valid if and only if the set $\{p_1,\ldots,p_n,\neg q\}$ is unsatisfiable.

In our case, we want to check that, for a given formula θ in ATL or one of its extensions, there exists at least one concurrent game model in which θ is true, and when possible, we want to extract one of these concurrent game models.

In order to test satisfiability, several methods have been developed. We give a non-exhaustive list of these methods. First we can mention the method known by all students in computer science, which is the truth table, that the value of the formula computes for each combination of values for every proposition. The drawbacks of this method is that it is applicable only to Boolean logic, since we need to know all the possible values taken by the variables, and it is not efficient, indeed this method is exponential in the number of variables, whereas the problem of satisfiability for Boolean logic, known as *SAT*, is NP-complete. A large community works on this problem and manages to find very powerful solvers for SAT, which are used to solve a wide range of problems, including model checking of linear temporal logic [17].

Another method to test satisfiability is *resolution*, which consists in an inference rule for Boolean logic and several inference rules for more complex logics like the coalition logic [44], leading to a refutation theorem-proving technique. This technique is attributed to M. Davis and H. Putman [21], and was improved by J. Robinson [58]. This last method has eliminated the combinatorial explosion of its predecessors. The resolution method is at the origin of the programming language Prolog.

We can also mention *automata-based decision procedures*, which are widely used for both model checking and satisfiability of temporal logics. In particular, there exists an automata-based decision procedure for ATL, proposed by V. Goranko and G. van Drimmelen [32] and one for ATL* [61], proposed by S. Schewe. Finally, we mention the tableaux, on which this thesis is based. It is worth noting that tableaux and automata are sometimes used together to decide satisfiability, as in [28] for CTL*.

1.5 Tableau Methods

Tableaux have the great advantage of being intuitive, as they follow the semantics of the logic for which they are developed, and easily implementable. These are two reasons among others that make us focus on tableau methods.

Even if tableaux have different forms, as we will see, they all try, by applying different rules, to decompose formulae into "simpler" ones in order to get a contradiction at one point or another. If a contradiction is found, this means that the formula is unsatisfiable.

First, tableaux have been developed simultaneously by Beth, Hintikka and Schütte in the middle the 1950's for the Boolean logic and similar logics.

Beth tableaux [7, 8] are represented as tables, where the left part contains valid expressions and the right part contains invalid ones. To obtain a positive result in terms of satisfiability, the same expression must not appear in both sides of the tableaux. It is worth noticing that the French translation of the word "table" is, in this context, "*tableau*" and that it also appears that E. Beth spoke French, so we think that is probably the reason why tableaux are called tableaux.

Hintikka tableaux [33] are trees, whose nodes are a set of formulae and the root is the set that contains only the input formula. The input formula is unsatisfiable if for every branch of the tree, there exists an inconsistent node, that is, a node which is not a Hintikka set.

In 1968, R. Smullyan improved the tableau methods of Beth and Hintikka, and also adapted them to first order logic [63]. Its tableaux are also represented as trees, where each node contains a formula preceded by "T" or "F". The formula is unsatisfiable if for every branch of the tree, the same proposition is preceded by "F" on some node of the branch and by "T" on another node of this same branch. When talking about modal logics, the tableau method has been widely developed by M. Fitting [26].

Then, in 1985, P. Wolper transformed Smullyan tableaux for LTL [66]. In that purpose, he

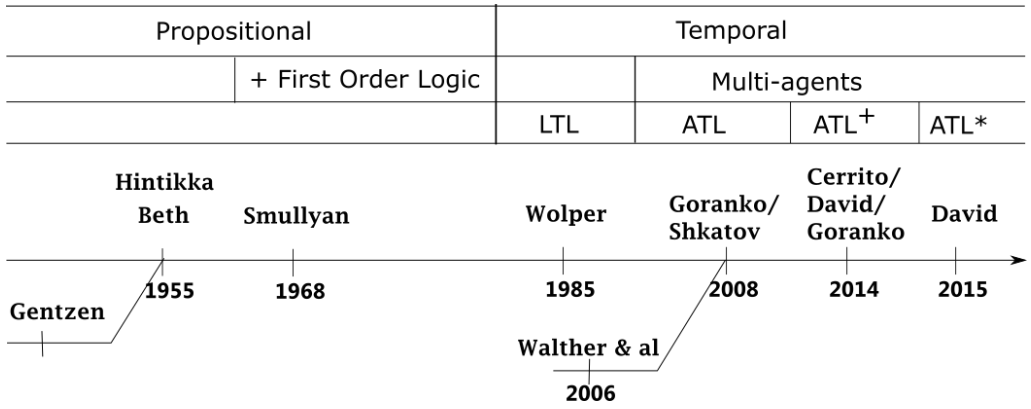


Figure 1.5: A chronology of tableau methods

employed fixed point equivalences to deal with temporal operators. One step is needed to make time moves one step forward, which is done by a dynamic rule, called *next*. Also, because of the linear and infinite semantics of LTL, the construction phase, where formulae are decomposed, is followed by an elimination phase to get rid of paths where formulae of the form $\Diamond\varphi$ or $\psi \mathsf{U} \varphi$ are not fulfilled, which are paths where φ never appears. It is worthwhile to remark that with P. Wolper, tableaux become graphs, because of the need of dealing with cycling operators such as $\Box$ (*Always*).

Tableaux for ATL were introduced in 2009, by V. Goranko and D. Shkatov [31]. The structure of these tableaux is slightly different from the one of Wolper, since decomposition of formulae is condensed in one step. However, the main difference comes from the treatment of coalitions and in particular from the dynamic rule, which is needed to compute the different actions and transitions for agents involved in the analysed formula. This method is presented with full details in Chapter 3. This tableau-based decision procedure for ATL is the basis of our tableau procedures for ATL^+ and ATL^*.

1.6 Our Contribution

The aim of this thesis is to give tools for the design of open systems. The more refined idea behind this ambitious goal is to use constructive procedures for deciding satisfiability of a given specification (that is, in our case, a formula in ATL or in its extensions) first to make sure that the specification is coherent and therefore implementable, and second to be able to directly produce a model corresponding to the specification. This refined goal is partially achieved in [31] by giving a procedure to decide satisfiability for ATL formulae using tableaux and also by giving a way to extract models from the tableau, in the case where the formulae are satisfiable. However, this tableau procedure only works for ATL formulae and the extracted models are awfully huge. This gives us two areas of improvement:

- extend the tableau-based procedure for ATL to its extensions ATL^+ and ATL^*;
- improve the extraction of models from tableaux.

This thesis focuses on the first point, as it was a good start to fully understand the concurrent game models and the logic. The second point is an ongoing work and will be discussed in the conclusion and perspectives of this thesis.

This thesis is composed of two parts: *Preliminaries* and *Deciding* ATL^+ *and* ATL^* *satisfiability by tableaux*. In Part **I**, Chapter 2, we describe models adapted to multi-agent systems, and in particular *concurrent game models*. Then we present the syntax and semantics of ATL and its extensions. Therefore Chapter 2 gives key notions to specify multi-agent systems.

In Chapter 3, we present the tableau-based decision procedure introduced by V. Goranko and D. Shkatov [31] to check satisfiability of ATL formulae.

In Part **II**, Chapter 4, we present our contribution, consisting in sound, complete, and optimal tableau-based decision procedures for ATL^+ and ATL^*. In Chapter 5, we present the implementation of the procedure for ATL^*, which includes the procedure for ATL^+.

Finally, Appendices contain additional definitions that are needed for proofs, proofs of soundness and completeness of our tableau-based decision procedure for ATL^*, and the list of all symbols that are used in this thesis.

Part I

Preliminaries

CHAPTER 2

OPEN SYSTEMS: MODELS AND LOGICS

In this chapter, we present two appropriate models for multi-agent systems, namely the *alternating transition system* (ATS) and the *concurrent game models* (CGM). These models have been elaborated as support for the semantics of the *alternating-time temporal logic* (ATL)[4] and its extensions ATL* [4], ATL$^+$ [41] and EATL [42]. Within the family of alternating-time temporal logics, one can express properties such as $\langle\langle A\rangle\rangle F$, which means "the coalition A of agents has a strategy, no matter what the agents outside the coalition do, to achieve the goal F". In ATL, every temporal operator is directly associated to a path quantifier $\langle\langle A\rangle\rangle$, the EATL version of ATL adds fairness constraints, whereas the ATL$^+$ version allows Boolean combination of temporal operators. Finally, ATL* is the full version of ATL, and allows Boolean combination and nesting of temporal operators. After having given syntaxes and semantics of these different extensions, we present different variations of the semantics of ATL based on the set of agents considered in the model or the size of the memory agents have. All these variations have an impact on the class of satisfiable formulae.

2.1 Modelling of Multi-Agents / Open Systems

Two main formalisms have been described to model multi-agent systems:

- *alternating transition system*, in short ATS, introduced in [3], and
- *concurrent game models*, in short CGM, introduced in [4].

In the following, we give their formal definitions, as well as some examples based on our booking automata case. Although they are different formalisms, one can easily transform an ATS into a CGM in exponential time, and it has been shown that a CGM can be transformed into an ATS in cubic time [30, 42]. The tableaux-based decision procedures for the family of

alternating-time temporal logics try to build a CGM like structure of a given input formula (see Chapters 3 and 4), so the semantics of ATL and its extensions will be given over CGM.

Note that another model for multi-agent system is the *game frame* introduced in [48] for *coalition logic*, and extended into *multi-player game models* (MGM) in [29] by adding a labelling function in order to associate propositions with states. The semantics of ATS, CGM and MGM are compared in [30].

2.1.1 Alternating Transition Systems

Definition 2.1 (Alternating Transition System)**.** An alternating transition system [3], in short ATS, is a structure where, at a given state, each agent can choose among several sets of states. The intersection of all selected sets of states gives a transition to a unique state.

An *alternating transition system* (ATS) is a tuple $\langle \mathbb{A}, \mathbb{S}, \mathscr{C}, \mathbb{P}, L \rangle$ where:

- $\mathbb{A}$ is a non-empty set of agents;
- $\mathbb{S}$ is a non-empty set of states;
- $\mathscr{C} : \mathbb{A} \times \mathbb{S} \to \mathscr{P}(\mathscr{P}(\mathbb{S}))$ provides for each agent $\mathsf{a} \in \mathbb{A}$ and state $s \in \mathbb{S}$ a set of actions playable by a at s. Note that for all states $s \in \mathbb{S}$ the intersection $\bigcap_{\mathsf{a} \in \mathbb{A}} \mathscr{C}(\mathsf{a}, s)$ must be a singleton in order to get a deterministic ATS;
- $\mathbb{P}$ is a non-empty set of atomic propositions
- $L : \mathbb{S} \to \mathscr{P}(\mathbb{P})$ is a labelling function.

Example 2.1. Let us model the booking automata of the introductory case as an alternating transition system:

$$\mathscr{A} = \langle \mathbb{A}, \mathbb{S}, \mathscr{C}, \mathbb{P}, L \rangle, \text{ where}$$

- $\mathbb{A} = \{H, B\}$, with H corresponds to Hugo and B to the cheater Bob;
- $\mathbb{S} = \{s_1, s_2, s_3, s_4, s'_4, s_5, s'_5, s_6, s_7, s_8, s_9, s_{10}, s_{11}\}$;
- the mapping $\mathscr{C}$ is given in Table 2.1;
- $\mathbb{P} = \{\text{logged}, \text{ticket\_printed}, \text{ticket\_reimbursed}, \text{ticket\_unavailable}\}$;
- the labelling function is
  - $L(s_0) = \emptyset$
  - $L(s_1) = L(s_2) = L(s_3) = \{\text{logged}\}$
  - $L(s_4) = L(s'_4) = L(s_6) = L(s_8) = \{\text{ticket\_printed}, \text{logged}\}$
  - $L(s_5) = L(s'_5) = L(s_7) = L(s_9) = \{\text{ticket\_reimbursed}, \text{logged}\}$
  - $L(s_{10}) = \{\text{ticket\_printed}, \text{ticket\_reimbursed}, \text{logged}\}$
  - $L(s_{11}) = \{\text{ticket\_unavailable}\}$.

2.1.2 Concurrent Games Models

A concurrent game model [4], in short CGM, is a state space over which a game between (one or) several agents is played. At each step of the game, independently and synchronously, every

$\mathscr{C}$	H (Hugo)	B (Bob)
s_0	$\{\{s_0,s_1\},\{s_2,s_3\}\}$	$\{\{s_0,s_2\},\{s_1,s_3\}\}$
s_1	$\{\{s_1,s_4,s_5\},\{s_3,s_6,s_7\}\}$	$\{\{s_1,s_3\},\{s_4,s_6\},\{s_5,s_7\}\}$
s_2	$\{\{s_2,s_3\},\{s_4,s_8\},\{s_5,s_9\}\}$	$\{\{s_2,s_4,s_5\},\{s_3,s_8,s_9\}\}$
s_3	$\{\{s_3,s_4,s_5\},\{s'_4,s_{10}\},\{s'_5,s_{10}\}\}$	$\{\{s_3,s'_4,s'_5\},\{s_4,s_{10}\},\{s_5,s_{10}\}\}$
s_4 / s'_4	$\{\{s_{11}\}\}$	$\{\{s_{11}\}\}$
s_5 / s'_5	$\{\{s_{11}\}\}$	$\{\{s_{11}\}\}$
s_6 / s_7	$\{\{s_2\}\}$	$\{\{s_2\}\}$
s_8 / s_9	$\{\{s_1\}\}$	$\{\{s_1\}\}$
s_{10} / s_{11}	$\{\{s_{11}\}\}$	$\{\{s_{11}\}\}$

Table 2.1: The mappings of possible sets of choices for Hugo and Bob at each state

agent chooses an action available to him. A state transition in a CGM is the combination of every agent's choices. More specifically, a CGM is a directed graph whose vertices represent states of the game and are labelled by a (possibly empty) set of propositions which are true at these states, and edges are labelled by a vector containing every agent's actions.

Over time, different terms have appeared in the literature to describe concurrent game models. In addition to concurrent game models, we may find concurrent game structures (CGS) or concurrent game frames (CGF). Definitions of these three notions sometimes overlap. In this thesis, we discard the term of concurrent game frame and define the two others as follows:

Definition 2.2 (Concurrent Game Structure)**.** A *concurrent game structure* (CGS) is a tuple $\mathscr{S} = \langle \mathbb{A}, \mathbb{S}, \{\mathsf{Act}_\mathsf{a}\}_{\mathsf{a}\in\mathbb{A}}, \{\mathsf{act}_\mathsf{a}\}_{\mathsf{a}\in\mathbb{A}}, \mathsf{out}\rangle$ where:

- $\mathbb{A} = \{1,\dots,k\}$ is a finite non-empty set of *agents* (or *players*);
- $\mathbb{S}$ is a non-empty set of *states*;
- for each agent $\mathsf{a}\in\mathbb{A}$, Act_a is a non-empty set of *actions*. For any *coalition* $A\subseteq\mathbb{A}$ we denote $\mathsf{Act}_A := \prod_{\mathsf{a}\in A}\mathsf{Act}_\mathsf{a}$ and use σ_A to denote a tuple from Act_A. Let us call this tuple an *A-action*. In particular, $\mathsf{Act}_\mathbb{A}$ is the set of all possible *action vectors* in $\mathscr{S}$. Also, we denote by $\sigma_A(\mathsf{a})$ the action of the agent a in the A-action σ_A;
- for each agent $\mathsf{a}\in\mathbb{A}$, $\mathsf{act}_\mathsf{a}:\mathbb{S}\to\mathscr{P}(\mathsf{Act}_\mathsf{a})-\emptyset$ is a map defining for each state s the actions available to a at s. Moreover, $\mathsf{act}_A:\mathbb{S}\to\mathscr{P}(\mathsf{Act}_A)-\emptyset$ maps a set of A-actions to every state s, i.e. $\mathsf{act}_A(s)=\prod_{\mathsf{a}\in A}\mathsf{act}_\mathsf{a}(s)$;
- out is a transition function that assigns to every state $s\in\mathbb{S}$ and every action vector $\sigma_\mathbb{A}=\langle\sigma_1,\dots,\sigma_k\rangle\in\mathsf{act}_\mathbb{A}(s)$ a state $\mathsf{out}(s,\sigma_\mathbb{A})\in\mathbb{S}$ that results from s if every agent $\mathsf{a}\in\mathbb{A}$ plays action σ_a.

Notation 2.1. we will often use the expressions "$s\in\mathscr{S}$" and "$s\in\mathscr{M}$" instead of "$s\in\mathbb{S}$ in $\mathscr{S}$" and "$s\in\mathbb{S}$ in $\mathscr{M}$", respectively.

Definition 2.3 (Outcome of σ_A)**.** Let $\mathscr{S}$ be a CGS and $s \in \mathscr{S}$ a state . Let $A \subseteq \mathbb{A}$ and $\sigma_A \in \mathsf{act}_A(s)$ be an A-action. The *outcome of* σ_A *at* s, denoted $\mathsf{Out}(s, \sigma_A)$, is the set of states $\mathsf{out}(s, \sigma_{\mathbb{A}})$ for all $\sigma_{\mathbb{A}}$ such that $\sigma_{\mathbb{A}}(\mathsf{a}) = \sigma_A(\mathsf{a})$ for every $\mathsf{a} \in A$.

Definition 2.4 (Concurrent Game Model)**.** A *concurrent game model* (CGM) is a tuple $\mathscr{M} = \langle \mathbb{A}, \mathbb{S}, \{\mathsf{Act}_\mathsf{a}\}_{\mathsf{a}\in\mathbb{A}}, \{\mathsf{act}_\mathsf{a}\}_{\mathsf{a}\in\mathbb{A}}, \mathsf{out}, \mathbb{P}, L\rangle$ where :

- $\langle \mathbb{A}, \mathbb{S}, \{\mathsf{Act}_\mathsf{a}\}_{\mathsf{a}\in\mathbb{A}}, \{\mathsf{act}_\mathsf{a}\}_{\mathsf{a}\in\mathbb{A}}\rangle$ is a concurrent game structure;
- $\mathbb{P}$ is a non-empty set of atomic propositions;
- $L : \mathbb{S} \to \mathscr{P}(\mathbb{P})$ is a labelling function.

Example 2.2. Let us model the booking automata of the introductory case with the following concurrent game model:

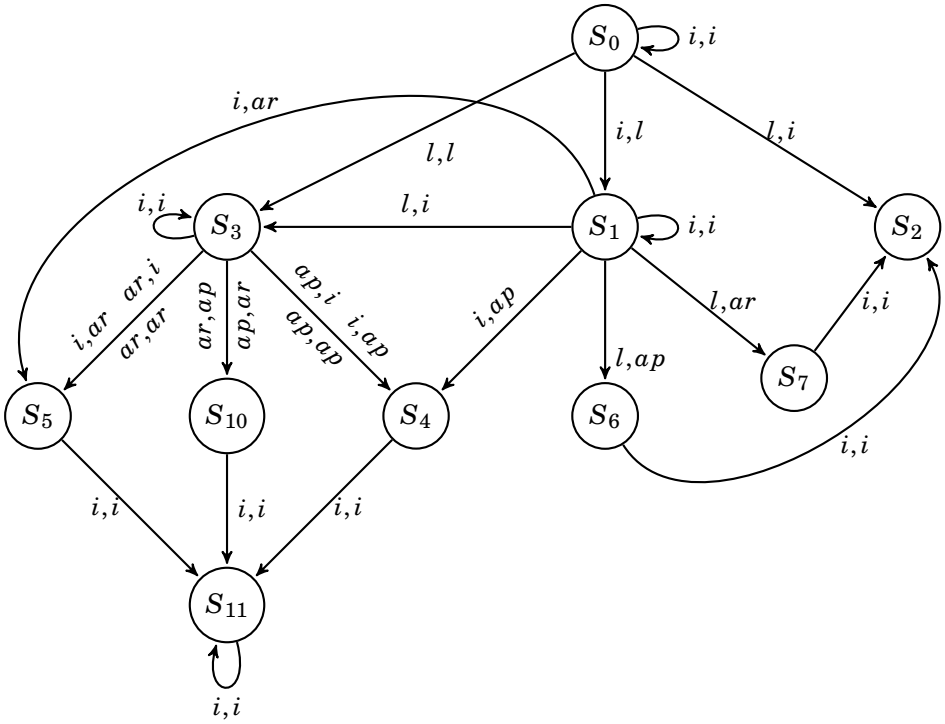
$$\mathscr{M}_{ticket} = \langle \mathbb{A}, \mathbb{S}, \{\mathsf{Act}_\mathsf{a}\}_{\mathsf{a}\in\mathbb{A}}, \{\mathsf{act}_\mathsf{a}\}_{\mathsf{a}\in\mathbb{A}}, \mathsf{out}, \mathbb{P}, L\rangle, \text{ where}$$

- $\mathbb{A} = \{H, B\}$, where H corresponds to Hugo and B to the cheater Bob;
- $\mathbb{S} = \{s_1, \ldots, s_{11}\}$;
- for both agents Hugo and Bob, the actions are "idle" (do nothing), "login" (enter a booking number), "ask_print" (ask for the printing of the ticket) and "ask_reimbursement" (ask for the reimbursement of the ticket);
- the mapping defining the actions available at each state for Hugo and Bob is given in Table 2.2;

	act_H	act_B
s_0	idle, login	idle, login
s_1	idle, login	idle, ask_print, ask_reimbursement
s_2	idle, ask_print, ask_reimbursement	idle, login
s_3	idle, ask_print, ask_reimbursement	idle, ask_print, ask_reimbursement
s_4 /.../s_{11}	idle	idle

Table 2.2: The mappings act_H and act_B defining the actions available to Hugo and Bob at each state

- the transition function is given in Figure 2.1.
- $\mathbb{P} = \{\text{logged}, \text{ticket\_printed}, \text{ticket\_reimbursed}, \text{ticket\_unavailable}\}$;
- the labelling function is
  - $L(s_0) = \emptyset$
  - $L(s_1) = L(s_2) = L(s_3) = \{\text{logged}\}$
  - $L(s_4) = L(s_6) = L(s_8) = \{\text{ticket\_printed}, \text{logged}\}$
  - $L(s_5) = L(s_7) = L(s_9) = \{\text{ticket\_reimbursed}, \text{logged}\}$
  - $L(s_{10}) = \{\text{ticket\_printed}, \text{ticket\_reimbursed}, \text{logged}\}$
  - $L(s_{11}) = \{\text{ticket\_unavailable}\}$



In order to keep this CGM readable, we haven't represented the transitions from s_2, since they are symmetric to the outgoing transitions from s_1. Indeed these transitions are as follows:

- $\text{out}(s_2, \langle \text{idle}, \text{idle} \rangle) = S_2$
- $\text{out}(s_2, \langle \text{idle}, \text{login} \rangle) = S_3$
- $\text{out}(s_2, \langle \text{ask\_print}, \text{idle} \rangle) = S_4$
- $\text{out}(s_2, \langle \text{ask\_print}, \text{login} \rangle) = S_8$
- $\text{out}(s_2, \langle \text{ask\_reimbursement}, \text{idle} \rangle) = S_5$
- $\text{out}(s_2, \langle \text{ask\_reimbursement}, \text{login} \rangle) = S_9$
- $\text{out}(s_8, \langle \text{ask\_reimbursement}, \text{idle} \rangle) = S_1$
- $\text{out}(s_9, \langle \text{ask\_reimbursement}, \text{login} \rangle) = S_1$

Figure 2.1: Transitions in the CGM $\mathscr{M}_{ticket}$

2.2 ATL: A Logic for Multi-Agents Systems

The *alternating-time temporal logic*, in short ATL, was first introduced in 1997 [3] by Alur, Henzinger and Kupferman, and then modified in 2002 [4]. The first version was based on alternating transition systems, whereas the second was based on concurrent game models. ATL is a member of the temporal logics' family, as LTL and CTL. In fact, it directly extends CTL with the notions of agents and coalitions of agents.

Different versions of the alternating-time temporal logic exist, increasing or decreasing the expressiveness of the logic and therefore also increasing or decreasing its complexity [41, 61]. For instance, the version EATL allows one to express fairness constraints.

The next subsection describes the syntax and specificity of several versions of ATL.

2.2.1 Syntax of Different ATL Versions

Although the syntax of ATL, ATL$^+$ or ATL* is similar to the one of LTL or CTL, the new kind of path quantifiers, called *strategic quantifiers*, makes a big difference. It is now possible to specify which coalition of agents must achieve a given property. In our game, agents of the coalition are considered as proponents and all the other agents of the game as opponents. There are two different path quantifiers. The first one, denoted $\langle\langle A \rangle\rangle$ where A is a coalition of agents, means that the coalition A has a strategy no matter the actions chosen by opponents. When we write $\langle\langle A \rangle\rangle \Phi$, we mean that the coalition A can enforce the property (or goal) Φ. On the contrary, the strategic quantifier $[\![A]\!]$ means that the opponents have at least a riposte to any strategy of the coalition A that enforces Φ. So a formula $[\![A]\!]\Phi$ means that the coalition A cannot avoid the property Φ.

First, we recall the meaning of several temporal operators and equivalences between them. The main operators in temporal logic are $\bigcirc$, $\Box$, U whose significance is "*Next*", "*Always*" and "*Until*" respectively. Other operators exist and can be defined with the previous three ones as follows: "*True*" $\top := p \vee \neg p$, "*False*" $\bot := \neg\top$, "*Sometime*" $\Diamond\varphi := \top \mathsf{U} \varphi$ (but also $\Diamond\varphi := \neg\Box\neg\varphi$), "*Release*" $\psi \mathsf{R} \varphi := \Box\varphi \vee \varphi \mathsf{U} (\varphi \wedge \psi)$. In the literature, the symbols $\bigcirc$, $\Box$ and $\Diamond$ are sometimes replaced by X, G and F respectively.

We now give the syntax of different members of the family of alternating-time temporal logics. The first one is ATL, also known as "*vanilla* ATL". In this version, temporal operators are immediately preceded by a strategic quantifier. The second one is ATL*, also known as "*full* ATL": in the scope of a strategic quantifier, temporal operators can be combined with Boolean connectors and nested one inside another. Between ATL and full ATL, there are several versions reducing possibilities of combination and/or nesting. We will particularly focus on ATL$^+$ and also give as an example the syntax of EATL. Both versions are presented in [42]. The fragment ATL$^+$ only allows Boolean combinations of temporal operators, while the fragment EATL allows two sorts of temporal operator's nesting: "*Infinitely often*" $\overset{\infty}{\Diamond}$ (or GF) and "*Sometime always*" $\overset{\infty}{\Box}$ (or FG),

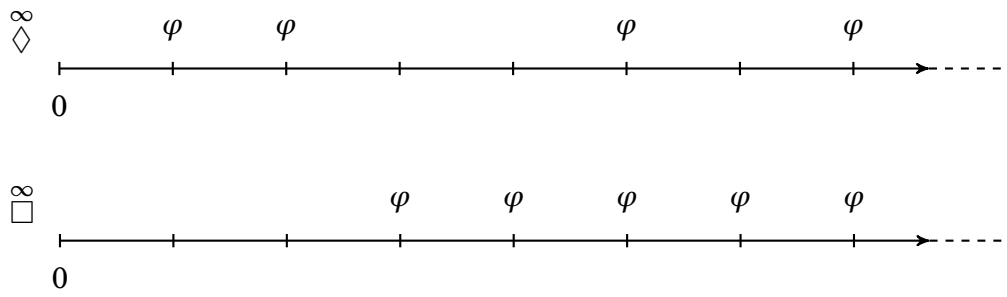


Figure 2.2: The temporal operators $\overset{\infty}{\Diamond}$ ("*Infinitely often*") and $\overset{\infty}{\Box}$ ("*Sometime always*")

see Figure 2.2.

Let $\mathbb{A}$ be any finite set of agents. In the following syntaxes, p is an atomic proposition from a given set $\mathbb{P}$, l is a literal, that is p or $\neg p$, and A is a coalition of agents from $\mathbb{A}$, that is A ranges over $\mathscr{P}(\mathbb{A})$.

Syntax of ATL**:**

(2.1) $$\text{ATL-formula } \varphi := p \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid \langle\langle A\rangle\rangle \bigcirc \varphi \mid \langle\langle A\rangle\rangle \Box\varphi \mid \langle\langle A\rangle\rangle \varphi \mathsf{U} \varphi$$

Ex. Hugo and Bob do not have a strategy to enforce the system to be in a state where a ticket is printed and the ticket is reimbursed, is expressed by

$$\neg\langle\langle H,B\rangle\rangle \Diamond(\text{ticket\_printed} \wedge \text{ticket\_reimbursed})$$

Remark 2.1. To have more readable formulae, we omit parentheses where there is no ambiguity. Also, in examples, we write $\langle\langle H,B\rangle\rangle$ as a shortcut of $\langle\langle\{H,B\}\rangle\rangle$.

Syntax of EATL**:**

(2.2) $$\text{EATL-formula } \varphi := p \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid \langle\langle A\rangle\rangle \bigcirc \varphi \mid \langle\langle A\rangle\rangle \Box\varphi \mid \langle\langle A\rangle\rangle \varphi \mathsf{U} \varphi \mid \langle\langle A\rangle\rangle \overset{\infty}{\Diamond} \varphi \mid \langle\langle A\rangle\rangle \overset{\infty}{\Box} \varphi$$

Ex. Hugo gets log infinitely often with a booking number to the booking automaton:

$$\langle\langle H\rangle\rangle \overset{\infty}{\Diamond} \text{logged}$$

Differently from ATL or EATL, the syntax for ATL$^+$ or ATL* is expressed in term of *state formulae* evaluated at a given state and *path formulae* evaluated on a given path (see subsection 2.2.2).

Notation 2.2. In order to differentiate state formulae from path formulae, we use the lower case Greek letters φ, ψ, θ, ξ to denote state formulae, and the capital Greek letters Φ, Ψ for path formulae.

In order to simplify the description of the tableau-based decision procedure for ATL$^+$ and ATL*, we give the syntax in negation normal form, that is all negations are pushed next to propositions. Therefore $\neg\langle\langle A\rangle\rangle\Phi$ is transformed into $[\![A]\!] \sim \Phi$, and $\neg[\![A]\!]\Phi$ into $\langle\langle A\rangle\rangle \sim \Phi$, where $\sim\Phi$ is the negation normal form of $\neg\Phi$.

Remark 2.2. The syntax of ATL cannot be put into negation normal form using the symbol $[\![]\!]$, because $\sim\varphi \mathsf{U} \psi \equiv \neg\varphi \mathsf{R} \neg\psi \equiv \Box\neg\psi \vee \neg\psi \mathsf{U}(\neg\psi \wedge \neg\varphi)$ is not well-formed in ATL.

Syntax of ATL$^+$:

$$\begin{aligned} &\text{ATL}^+\text{- state formula } \varphi := l \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid \langle\langle A\rangle\rangle\Phi \mid [\![A]\!]\Phi \\ &\text{ATL}^+\text{- path formula } \Phi := \varphi \mid (\Phi \wedge \Phi) \mid (\Phi \vee \Phi) \mid \bigcirc\varphi \mid \Box\varphi \mid (\varphi \mathsf{U} \varphi) \end{aligned} \tag{2.3}$$

Ex. Hugo has a strategy to print his ticket once he is logged with the booking number to the automaton:

$$\langle\langle H\rangle\rangle(\text{logged} \rightarrow \Diamond\text{ticket\_printed})$$

Syntax of ATL*:

$$\begin{aligned} &\text{ATL}^*\text{- state formula } \varphi := l \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid \langle\langle A\rangle\rangle\Phi \mid [\![A]\!]\Phi \\ &\text{ATL}^*\text{- path formula } \Phi := \varphi \mid (\Phi \wedge \Phi) \mid (\Phi \vee \Phi) \mid \bigcirc\Phi \mid \Box\Phi \mid (\Phi \mathsf{U} \Phi) \end{aligned} \tag{2.4}$$

Ex. Hugo should infinitely often be allowed to print his ticket after he is logged to the automaton, and Bob should infinitely often be allowed to reimburse his ticket after being logged to the automaton.

$$\langle\langle H\rangle\rangle\Box\Diamond(\text{logged} \rightarrow \bigcirc\Diamond\text{ticket\_printed}) \wedge \langle\langle B\rangle\rangle\Box\Diamond(\text{logged} \rightarrow \bigcirc\Diamond\text{ticket\_reimbursed})$$

Remark 2.3. The grammar of state formulae is the same for ATL$^+$ and ATL*, and it is the definition of path formulae which makes the difference.

2.2.2 Semantics

In this section, we will see how to interpret ATL formulae over a concurrent game model.

First we need to define what are a play, a history and a strategy in concurrent game models.

A *play*, denoted by λ, in a CGM is an infinite sequence $s_0, s_1, s_2, \ldots$ of states such that there exists an action vector $\sigma_{\mathbb{A}}$ such that $\mathsf{Out}(s_i, \sigma_{\mathbb{A}}) = s_{i+1}$ for each $i \geq 0$.

On a given play λ, we denote by λ_0 its initial state, by λ_i its $(i+1)$th state, by $\lambda_{\leq i}$ the prefix $\lambda_0 \ldots \lambda_i$ of λ, by $\lambda_{\geq i}$ the suffix $\lambda_i \lambda_{i+1} \ldots$ of λ, and in general by $\lambda_{i\sim j}$ the sub-sequence $\lambda_i \ldots \lambda_j$, where $0 \leq i \leq j$. For any sub-sequence $\lambda_{i\sim j}$, we say that its length is $\ell = j - i$ and we write $|\lambda_{i\sim j}| = \ell$ and $\mathsf{last}(\lambda_{i\sim j}) = \lambda_j$. A *history*, denoted h, is a finite sub-sequence of a play.

Given a CGM $\mathscr{M}$, we denote by $\mathsf{Plays}_{\mathscr{M}}$ the set of plays, and for a state $s \in \mathbb{S}$, by $\mathsf{Plays}_{\mathscr{M}}(s)$ the set of plays with initial state s.

For a given coalition A, an *A-strategy*, denoted F_A, associates an action vector $\sigma_A \in \mathsf{Act}_A$ with each state $s \in \mathbb{S}$ and each possible history at that state. The kind of history on which the strategy is based may consist of only the current state, in that case we said that the strategy is *memoryless* or *positional*. On the opposite side, agents may be able to remember all the history of the play up to the current state s, we then speak of *perfect-recall strategy*. Between these two extremes, we find *bounded-recall strategies*: if b is a given bound on the memory, then agents are able to remember the b previous states in addition to the current state. In the extreme cases of memoryless and perfect-recall strategies, $b = 1$ and $b = \omega$ respectively. Then the set of all histories in $\mathscr{M}$ is defined as $\mathsf{Hist}_{\mathscr{M}}[b] = \bigcup_{1 \leq n < 1+b} \mathbb{S}^n$.

Formally, we define an A-strategy by $F_A[b] : \mathsf{Hist}_{\mathscr{M}}[b] \to \mathsf{Act}_A$ such that $F_A[b](h) \in \mathsf{act}_A(\mathsf{last}(h))$ for every $h \in \mathsf{Hist}_{\mathscr{M}}[b]$. We also denote by $\mathsf{Strat}_{\mathscr{M}}(A)[b]$ the set of collective strategies bounded by b of a coalition A in $\mathscr{M}$. The set of plays starting at s *consistent* with an A-strategy $F_A[b]$, denoted $\mathsf{Plays}_{\mathscr{M}}(s, F_A[b])$, is the set of all plays $\lambda \in \mathsf{Plays}_{\mathscr{M}}(s)$ such that $\lambda_{i+1} \in \mathsf{Out}(\lambda_i, F_A[b](\lambda_{j\sim i}))$ for all $i \geq 0$, where $j = \max\{i - b + 1, 0\}$.

For any coalition $A \subseteq \mathbb{A}$, a given CGM $\mathscr{M}$ and a state $s \in \mathscr{M}$, an *A-co-action* at s in $\mathscr{M}$ is a mapping $\mathsf{Act}^{\mathsf{c}}{}_A : \mathsf{Act}_A \to \mathsf{Act}_{\mathbb{A}-A}$. An A-co-action assigns to every collective action of A at state s in $\mathscr{M}$ a collective action at s for the complementary coalition $\mathbb{A} - A$.

Also, we define an *A-co-strategy* in $\mathscr{M}$ as a mapping $F^c_A[b] : \mathsf{Strat}_{\mathscr{M}}(A)[b] \times \mathsf{Hist}_{\mathscr{M}}[b] \to \mathsf{Act}_{\mathbb{A}-A}$ that assigns to every collective strategy of A and every history $h \in \mathsf{Hist}_{\mathscr{M}}[b]$ a collective action at $last(h)$ for $\mathbb{A} - A$. The set of plays starting at s consistent with an A-co-strategy $F^c_A[b]$, denoted $\mathsf{Plays}_{\mathscr{M}}(s, F^c_A[b])$, is the set of all plays $\lambda \in \mathsf{Plays}_{\mathscr{M}}(s)$ such that $\lambda_{i+1} \in \mathsf{Out}(\lambda_i, F^c_A[b](F_A[b], \lambda_{j\sim i}))$ for all $i \geq 0$, where $j = \max\{i - b + 1, 0\}$.

We usually write F_A, $\mathsf{Strat}_{\mathscr{M}}(A)$, F^c_A and $\mathsf{Hist}_{\mathscr{M}}$ instead of $F_A[b]$, $\mathsf{Strat}_{\mathscr{M}}(A)[b]$, $F^c_A[b]$ and $\mathsf{Hist}_{\mathscr{M}}$ [b] respectively when b is understood from the context.

The semantics of the different versions of ATL is defined over a given CGM $\mathscr{M}$, a state $s \in \mathscr{M}$, and if necessary, a play $\lambda \in \mathscr{M}$.

Semantics of ATL

- $\mathscr{M}, s \models p$ iff $p \in L(s)$, for any proposition $p \in \mathbb{P}$;
- $\mathscr{M}, s \models \neg\varphi$ iff $\mathscr{M}, s \not\models \varphi$;
- $\mathscr{M}, s \models \varphi \wedge \psi$ iff $\mathscr{M}, s \models \varphi$ and $\mathscr{M}, s \models \psi$;
- $\mathscr{M}, s \models \langle\langle A \rangle\rangle \bigcirc \varphi$ iff there exists an A-action $\sigma_A \in \mathsf{act}_A(s)$ such that $\mathscr{M}, s' \models \varphi$ for all $s' \in \mathsf{Out}(s, \sigma_A)$;
- $\mathscr{M}, s \models \langle\langle A \rangle\rangle \Box \varphi$ iff there exists an A-strategy F_A such that $\mathscr{M}, \lambda_i \models \varphi$ for all $\lambda \in \mathsf{Plays}_{\mathscr{M}}(s, F_A)$ and all $i \geq 0$;
- $\mathscr{M}, s \models \langle\langle A \rangle\rangle \varphi \mathsf{U} \psi$ iff there exists an A-strategy F_A such that, for all $\lambda \in \mathsf{Plays}_{\mathscr{M}}(s, F_A)$, there exists an $i \geq 0$ with $\mathscr{M}, \lambda_i \models \psi$ and $\mathscr{M}, \lambda_j \models \varphi$ for all $0 \leq j < i$.

Remark 2.4. In the case of the next operator, the strategic quantifier $\langle\langle A\rangle\rangle$ still denotes the existence of a A-strategy, but in this particular case a A-strategy is specified by a single local A-action at s.

Semantics of ATL* Here, we generalize the semantics for ATL to ATL*, which covers the semantics of EATL and ATL$^+$.

- $\mathcal{M}, s \models p$ iff $p \in L(s)$, for any proposition $p \in \mathbb{P}$;
- $\mathcal{M}, s \models \neg p$ iff $\mathcal{M}, s \not\models p$, for any proposition $p \in \mathbb{P}$;
- $\mathcal{M}, s \models \varphi \wedge \psi$ iff $\mathcal{M}, s \models \varphi$ and $\mathcal{M}, s \models \psi$;
- $\mathcal{M}, s \models \varphi \vee \psi$ iff $\mathcal{M}, s \models \varphi$ or $\mathcal{M}, s \models \psi$;
- $\mathcal{M}, s \models \langle\langle A\rangle\rangle\Phi$ iff there exists an A-strategy F_A such that, for all $\lambda \in \mathsf{Plays}_{\mathcal{M}}(s, F_A)$, $\mathcal{M}, \lambda \models \Phi$;
- $\mathcal{M}, s \models [\![A]\!]\Phi$ iff there exists an A-co-strategy F_A^c such that, for all $\lambda \in \mathsf{Plays}_{\mathcal{M}}(s, F_A^c)$, $\mathcal{M}, \lambda \models \Phi$;
- $\mathcal{M}, \lambda \models \varphi$ iff $\mathcal{M}, \lambda_0 \models \varphi$;
- $\mathcal{M}, \lambda \models \Phi \wedge \Psi$ iff $\mathcal{M}, \lambda \models \Phi$ and $\mathcal{M}, \lambda \models \Psi$;
- $\mathcal{M}, \lambda \models \Phi \vee \Psi$ iff $\mathcal{M}, \lambda \models \Phi$ or $\mathcal{M}, \lambda \models \Psi$;
- $\mathcal{M}, \lambda \models \bigcirc\Phi$ iff $\mathcal{M}, \lambda_{\geq 1} \models \Phi$;
- $\mathcal{M}, \lambda \models \Box\Phi$ iff $\mathcal{M}, \lambda_{\geq i} \models \Phi$ for all $i \geq 0$;
- $\mathcal{M}, \lambda \models \Phi \mathsf{U} \Psi$ iff there exists an $i \geq 0$ where $\mathcal{M}, \lambda_{\geq i} \models \Psi$ and for all $0 \leq j < i$, $\mathcal{M}, \lambda_{\geq j} \models \Phi$.

2.2.3 Satisfiability and Validity

We give a general definition of the notion of satisfiability and validity for ATL and its extensions. This definition will be refined in the next subsection.

Let φ be an ATL* formula, then φ is *satisfiable* if there exists a CGM $\mathcal{M}$ and a state $s \in \mathcal{M}$ such that $\mathcal{M}, s \models \varphi$.

Let φ be an ATL* formula, then φ is *valid* if for any CGM $\mathcal{M}$ and any state $s \in \mathcal{M}$, $\mathcal{M}, s \models \varphi$.

The satisfiability problem consists in answering the following question:

Given a formula φ, is φ satisfiable ?

2.2.4 Different Variations on ATL

By modifying some parameters, the answer to the satisfiability problem for a given ATL formula may differ, and the problem may also become undecidable. These parameters concern

- the set of agents that is considered in the model [31, 65]. That is, do we consider only the agents that occurs in the given formula or more agents?
- the type of memory held by agents. That is, in order to decide a strategy to apply, agents may refer only to the current state (*memoryless strategies*) or to the history of the play. We

say that agents use *bounded-recall strategy* if they remember only a part of the history, and use *perfect-recall strategy* if they remember the whole history of the play.

- how much information is available to the agents [35]. That is, do agents always know what is the current state of the system?

2.2.4.1 Which Sets of Agents are Considered?

Three types of satisfiability for ATL have been defined in [65] and named in [31] as *tight-satisfiability*, $\mathbb{A}$-*satisfiability* and *general satisfiability*. They are formally defined as follows:

Notation 2.3. Let θ be a formula of the family of alternating-time temporal logics, we denote by $\mathbb{A}_\theta$ the set of agents occurring θ.

Definition 2.5. A formula θ is *tightly-satisfiable* if θ is satisfiable in a CGM $\mathscr{M} = \langle \mathbb{A}_\theta, \mathbb{S}, \{\mathsf{Act}_\mathsf{a}\}_{\mathsf{a}\in\mathbb{A}}, \{\mathsf{act}_\mathsf{a}\}_{\mathsf{a}\in\mathbb{A}}, \mathsf{out}, \mathbb{P}, L\rangle$.

Definition 2.6. A formula θ is $\mathbb{A}$-*satisfiable*, for some $\mathbb{A} \supseteq \mathbb{A}_\theta$, if θ is satisfiable in a CGM $\mathscr{M} = \langle \mathbb{A}, \mathbb{S}, \{\mathsf{Act}_\mathsf{a}\}_{\mathsf{a}\in\mathbb{A}}, \{\mathsf{act}_\mathsf{a}\}_{\mathsf{a}\in\mathbb{A}}, \mathsf{out}, \mathbb{P}, L\rangle$.

Definition 2.7. A formula θ is *generally-satisfiable* if θ is satisfiable in a CGM $\mathscr{M} = \langle \mathbb{A}', \mathbb{S}, \{\mathsf{Act}_\mathsf{a}\}_{\mathsf{a}\in\mathbb{A}}, \{\mathsf{act}_\mathsf{a}\}_{\mathsf{a}\in\mathbb{A}}, \mathsf{out}, \mathbb{P}, L\rangle$ for some $\mathbb{A}'$ with $\mathbb{A}_\theta \subseteq \mathbb{A}'$.

As far as the problem of satisfiability for ATL or its extensions is concerned, tight-satisfiability may return different results compared to the two other types of satisfiability, whereas $\mathbb{A}$-satisfiability and general satisfiability can be reduced to the same $\mathbb{A}$-satisfiability where $\mathbb{A} = \mathbb{A}_\theta \cup \{\mathsf{a}\}$. This special kind of $\mathbb{A}$-satisfiability is called *loose satisfiability*.

Example 2.3. Let us consider the following formula:

$$\neg\langle\langle \mathsf{a} \rangle\rangle \bigcirc p \wedge \neg\langle\langle \mathsf{a} \rangle\rangle \bigcirc q \wedge \langle\langle \mathsf{a} \rangle\rangle \bigcirc (p \vee q)$$

which means that the agent a does not have any strategy to make p true at next state, the agent a does not have any strategy to make q true at next state, and the agent a has a strategy to make either p or q true at the next state.

If the agent a is the only agent of the model (tight satisfiability), it is impossible to make this formula satisfiable. However, if another agent is added to the model (loose satisfiability), the formula becomes satisfiable as shown in Fig. 2.3, where 0 and 1 are the two actions available to agent 1 and 2.

2.2.4.2 What Memory do Agents Have ?

As seen in the semantics, there exist different kinds of strategies depending on the memory of agents. In the case of ATL and EATL, the class of satisfiable formulae does not depend on the type of memory that agents can use. This is not true when we deal with ATL$^+$ and ATL* formulae. The following example [35] illustrates this difference.

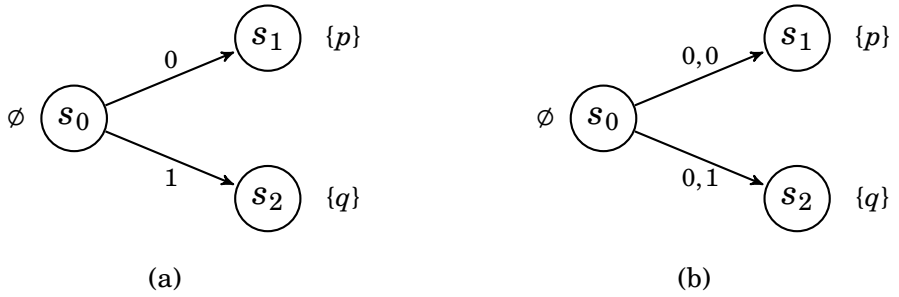


Figure 2.3: Influence of other agents on satisfiability. (a) An attempt to make the formula $\neg\langle\langle \mathsf{a}\rangle\rangle \bigcirc p \wedge \neg\langle\langle \mathsf{a}\rangle\rangle \bigcirc q \wedge \langle\langle \mathsf{a}\rangle\rangle \bigcirc (p \vee q)$ satisfiable with only one agent, which, of course, fails. (b) A model for the formula $\neg\langle\langle \mathsf{a}\rangle\rangle \bigcirc p \wedge \neg\langle\langle \mathsf{a}\rangle\rangle \bigcirc q \wedge \langle\langle \mathsf{a}\rangle\rangle \bigcirc (p \vee q)$ when an agent is added.

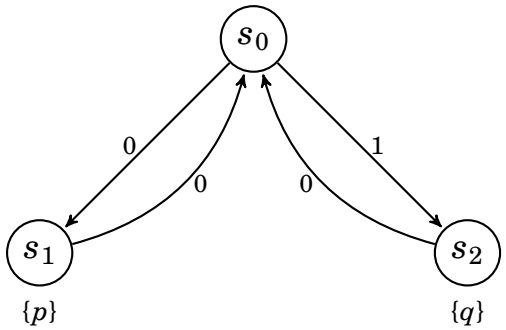


Figure 2.4: Model for the formula $\theta = \langle\langle \mathsf{a}\rangle\rangle(p \wedge \langle\langle \mathsf{a}\rangle\rangle \Diamond q) \wedge [\![\mathsf{a}]\!] \Diamond (\Box \neg p \vee \Box \neg q)$ with memoryless strategy

Example 2.4. Let us consider the formula

$$\theta = \langle\langle \mathsf{a}\rangle\rangle \Diamond (p \wedge \langle\langle \mathsf{a}\rangle\rangle \Diamond q) \wedge [\![\mathsf{a}]\!](\Box \neg p \vee \Box \neg q)$$

which means that the agent a has a strategy to eventually make p true while having some strategy for a that eventually makes q true, and the agent a cannot avoid making p always false or making q always false.

This formula is satisfiable if the agent a uses memoryless strategies, indeed a model for this formula is given in Fig. 2.4. Strategies for the different occurrences of the agent a in order to satisfy the formula θ are the following:

- for the first of a: $F_{\mathsf{a}}(s_0) = 0$, $F_{\mathsf{a}}(s_1) = 0$, $F_{\mathsf{a}}(s_2) = 0$;
- for the second occurrence of a: $F_{\mathsf{a}}(s_0) = 1$, $F_{\mathsf{a}}(s_1) = 0$, $F_{\mathsf{a}}(s_2) = 0$;
- for the third occurrence of a: $F_{\mathsf{a}}(s_0) = 0$, $F_{\mathsf{a}}(s_1) = 0$, $F_{\mathsf{a}}(s_2) = 0$ or $F_{\mathsf{a}}(s_0) = 1$, $F_{\mathsf{a}}(s_1) = 0$, $F_{\mathsf{a}}(s_2) = 0$, indifferently.

With perfect-recall strategies, the strategies for the third occurrence of the agent a can be transformed as $F_{\mathsf{a}}(\ldots, s_1, s_0) = 1$ and $F_{\mathsf{a}}(\ldots, s_2, s_0) = 0$. In this way, the agent a can avoid making p (resp. q) always true. Actually, the formula θ is unsatisfiable with perfect-recall strategies.

Indeed, to satisfy both $\langle\langle a\rangle\rangle\Diamond(p \wedge \langle\langle a\rangle\rangle\Diamond q)$ and $[\![a]\!](\Box\neg p \vee \neg q)$, the properties p and q must be on two different "branches" of a graph, as done in Fig. 2.4, but this gives the possibility to the agent a to change the branch it comes across each time it goes through the state s_0.

If we delve a little bit further, we see that this difference, w.r.t. satisfiability according to the kind of memory taken into account, comes from the possibility of connecting temporal operators with Boolean operators and of applying different strategies to different occurrences of the same agent. The first possibility is not allowed in ATL and EATL. The second possibility can be suppressed from ATL* by using *irrevocable strategies* [2] or *strategy contexts* [9] where agents must decide the same unique choice for all strategies in a given formula or a given part of a formula, respectively. In these cases, satisfiability is independent of the size of agent's memory.

In this thesis, we will focus on perfect-recall strategies, and when we will use the term "strategy", we will implicitly refer to "perfect-recall strategy".

2.2.4.3 What Information do Agents Have?

In the model we are interested in, agents have a full knowledge of the system. That is agents always know in which state of the system they are. On the contrary, in case of *imperfect information*, two states of the system may be indistinguishable from the point of view of the agents. To work with imperfect information, one needs to work with a different model, namely *imperfect information concurrent game models* (iCGM) [35]. Up to now, it is not known whether the satisfiability problem is decidable or not with imperfect information. In this thesis, we work always under the hypothesis of perfect information.

2.3 Conclusion

In this chapter, we have presented two models for describing multi-agent systems: alternating transition systems and concurrent game models. In the rest of this thesis, we will focus on the latter. We have given the syntax and semantics of ATL and of several of its extensions, namely EATL, ATL$^+$ and ATL*. Finally, we have discussed different variants of ATL and have seen that the set of agents considered in the models and the memory of agents have an impact on the class of satisfiable formulae for ATL$^+$ and ATL*. In the following, we will consider the tight and loose satisfiability problem for ATL and its extensions where agents use perfect-recall strategies and have perfect information about the system.

We are now ready to present a tableau-based decision procedure for ATL that will be the basis for our tableau procedures for ATL$^+$ and ATL*.

CHAPTER 3

TABLEAU-BASED DECISION PROCEDURE FOR ATL

In this chapter we describe the two-phase tableau-based decision procedure for the version ATL proposed in [31]. Introduced in 2009, historically this procedure is the third method that has been proposed to decide satisfiability of an ATL formula. The first method is an automata-based decision procedure [22, 32], while the second is a top-down tableau-like decision procedure [65] that was proposed to prove that the complexity of the satisfiability problem for ATL is EXPTIME regardless tight or loose satisfiability. The top-down approach exhaustively creates all the consistent subsets of the closure of the input formula, connects them according to the semantics, and, finally, tests if a model can be extracted. Obviously, a lot of redundant nodes can be created, in such an approach, that resumes to a sort of exhaustive search on the whole space of possible states. We recall that the kind of memory used by agents to answer the satisfiability question for ATL has no consequence on the class of satisfiable formulae and on the complexity of the problem. Indeed, in that case, memoryless and perfect-recall strategies are equivalent.

We present the procedure here with slight modifications w.r.t. [31] in order to be coherent with the new procedures we propose for ATL^+ and ATL^*. This procedure is illustrated by the following examples:

$$\theta_1 = \langle\langle H\rangle\rangle \bigcirc \text{logged} \wedge \neg\langle\langle H,B\rangle\rangle\Diamond(\text{ticket\_printed} \wedge \text{ticket\_reimbursed}) \wedge \neg\langle\langle H\rangle\rangle \bigcirc \text{ticket\_printed} \wedge \neg\langle\langle B\rangle\rangle \bigcirc \text{ticket\_reimbursed}$$

which means that Hugo can log at the next step, that Hugo and Bob do not have the possibility to print and reimburse a ticket at the same time, that Hugo cannot be sure to be able to print a ticket at the next step, and that Bob cannot be sure to be able to reimburse a ticket at the next

step. In order to save space in examples, we rewrite this formula as

$$\theta_1 = \langle\langle H\rangle\rangle \bigcirc l \wedge \neg\langle\langle H,B\rangle\rangle \Diamond (p \wedge r) \wedge \neg\langle\langle H\rangle\rangle \bigcirc p \wedge \neg\langle\langle B\rangle\rangle \bigcirc r \tag{3.1}$$

This formula can be seen a specification of the booking automata where we want to avoid the fact that two people can make a coalition to get a ticket and also to get it reimbursed. Of course, this is only a part of the specification, as we want to be able to compute it by hand. Moreover, this specification could be improved, using for instance ATL^+ or ATL^* formulae, but it is sufficient to illustrate the first phase of the procedure for ATL.

Since the above formula is satisfiable and the second phase is better illustrated with unsatisfiable formulae, we will also use the following formula:

$$\theta_2 = \langle\langle H\rangle\rangle \Box \neg \text{ticket\_reimbursed} \wedge \langle\langle B\rangle\rangle \Diamond \text{ticket\_reimbursed}$$

abbreviated as

$$\theta_2 = \langle\langle H\rangle\rangle \Box \neg r \wedge \langle\langle B\rangle\rangle \Diamond r \tag{3.2}$$

which means that Hugo has a strategy to never reimburse a ticket and that Bob has a strategy to reimburse a ticket.

3.1 General Description of the Procedure of V. Goranko and D. Shkatov

This tableau-based decision procedure attempts to build, step-by-step, from an initial formula θ, a rooted directed graph from which it is possible to extract a CGM satisfying θ. If the construction of such graph is possible, we say that the formula θ is satisfiable. Otherwise the formula θ is unsatisfiable.

In this directed graph, nodes are labelled by set of formulae (state formulae in the case of ATL^+ and ATL^*, see Chapter 4) and edges can be either of the form $\Longrightarrow$ or of the form $\xrightarrow{\sigma}$ where σ is an action vector. Nodes of the graph are partitioned in two categories: prestates and states. Prestates can be seen as nodes where the information contained in its formulae is "implicit". When we decompose all the formulae of a prestate and saturate the prestate, we obtain one or several states, that is prestates are treated in a static manner that just spells out the truth of the formulae they contain. States have the particularity of containing formulae of the form $\langle\langle A\rangle\rangle \bigcirc \varphi$ or $\neg\langle\langle A\rangle\rangle \bigcirc \varphi$ from which it is possible to compute the next steps of the tableau construction. Intuitively, a state is handled in a dynamical way, creating possible successors to the state (world) in the candidate model $\mathscr{M}$ one is trying to build. All prestates have states as successors and directed edges from prestates to states are of the form $\Longrightarrow$; on the other hand, all states have prestates as successors and directed edges from states to prestates are of the form $\xrightarrow{\sigma}$, where σ is an action vector.

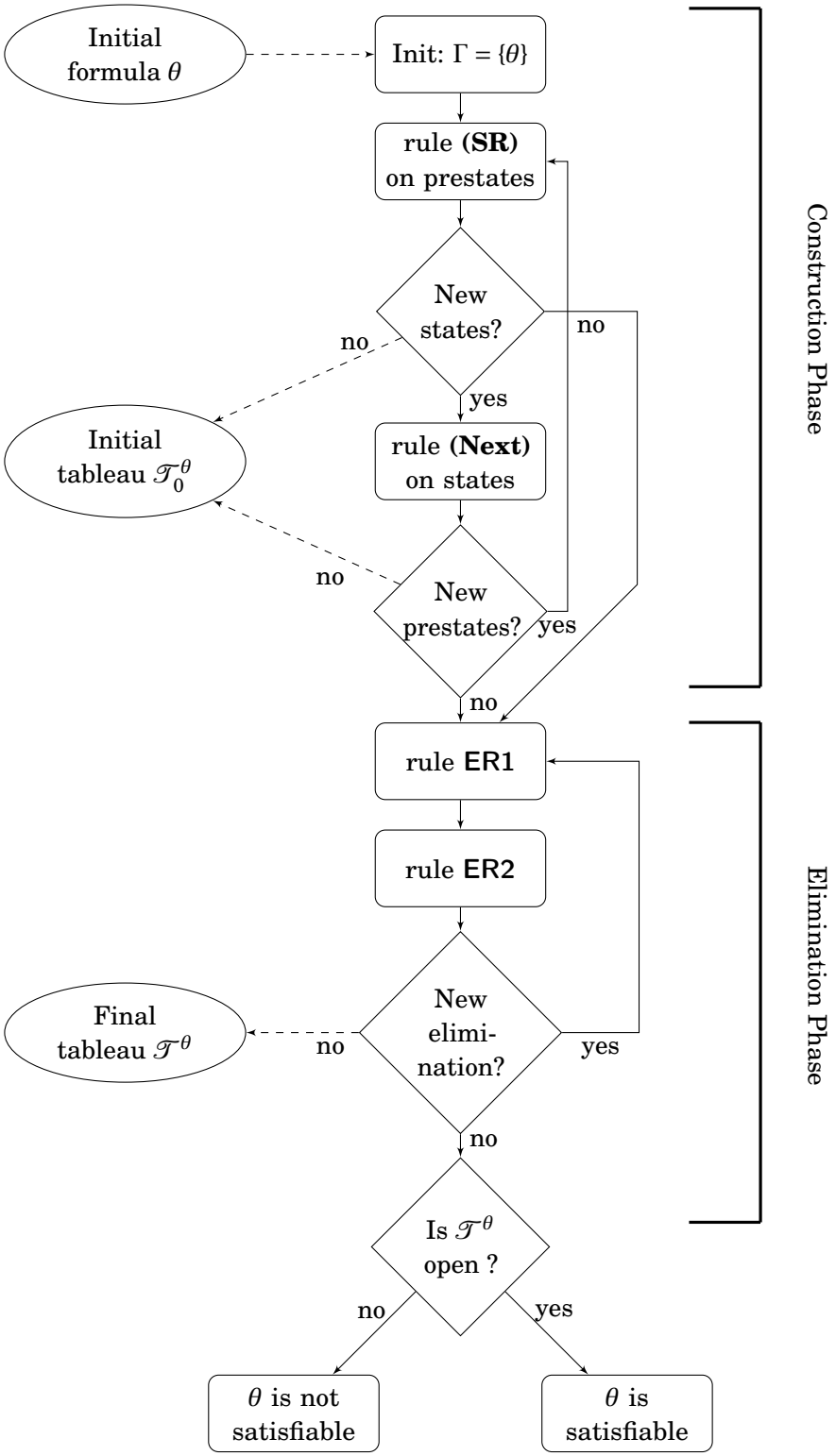


Figure 3.1: The general structure of the tableau-based decision procedure for the family of alternating-time temporal logics. The procedure starts with an initial formula θ and answers to the question: is θ satisfiable?

Figure 3.1 represents the different steps of the procedure. The procedure is in two phases: the construction phase and the elimination phase. First, we create an initial node, that is a prestate containing the initial formula θ to be tested w.r.t. satisfiability, and we construct the graph by expanding prestates into states via a static rule called **SR**, and by computing prestates from states with a dynamic rule called **Next**. The rule **SR** decomposes each formula of a prestate, and then saturates the prestate into new states. Explanations of the rules **SR** and **Next** can be found in Section 3.2.

The procedure avoids creation of duplicated nodes (a form of loop check), and consequently, ensures the termination of the procedure since the number of formulae that can appear in a node is finite (see Subsection 3.2). The construction phase ends when no new nodes can be added to the graph. The graph obtained at the end of the construction phase is called the *initial tableau for* θ, and is denoted by $\mathscr{T}_0^\theta$.

The second phase of the procedure eliminates via the rule **ER1** all nodes with missing successors, that is prestates with no more successors at all, or states with at least one missing action vector among its outcome edges, that is at least one missing successor. Also, by means of a rule called **ER2**, it eliminates all states with "unrealized eventualities", that is states which cannot ensure that all the objectives it contains will be fulfilled eventually. The graph obtained at the end of the elimination phase is called the *final tableau for* θ, also noted $\mathscr{T}^\theta$. An explanation of the elimination phase is given in Section 3.3.

The impact of the difference between tight satisfiability and loose satisfiability on the tableau procedure is only noticeable in the rule **Next**, see Section 3.2.3. When using the symbol $\mathbb{A}$ we refer to the set $\mathbb{A}_\theta$ of agents mentioned in θ in case of tight satisfiability and to the set $\mathbb{A}_\theta \cup \{k+1\}$, where k is the number of agents in θ, in case of loose satisfiability.

It is worth noting that CGMs are defined as having a non-empty set of agents, and that formula mentioning no agents may give wrong results if we want to decide tight satisfiability. For instance, the formula $\neg\langle\langle\emptyset\rangle\rangle\bigcirc p \wedge \neg\langle\langle\emptyset\rangle\rangle\bigcirc\neg p$, corresponding to the CTL formula $\neg AXp \wedge \neg AX\neg p$, will be wrongly declared unsatisfiable. Therefore, in that special case, one must necessarily check loose satisfiability to obtain the good result.

3.2 Construction Phase

The construction phase starts with a prestate containing the initial formula θ. The construction phase consists in the alternation of static analyses and dynamic analyses of formulae. The static analysis of formulae consists in extracting all the information encapsulated in prestate's formulae, in particular, information about the possible future, which is used during the dynamic analysis to build next states of the tableau.

α	α_1	α_2
$\neg\neg\varphi$	φ	φ
$\varphi_1 \wedge \varphi_2$	φ_1	φ_2
$\neg\langle\langle\mathbb{A}\rangle\rangle\bigcirc\varphi$	$\langle\langle\emptyset\rangle\rangle\bigcirc\neg\varphi$	$\langle\langle\emptyset\rangle\rangle\bigcirc\neg\varphi$
$\langle\langle A\rangle\rangle\Box\varphi$	φ	$\langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\Box\varphi$
β	β_1	β_2
$\neg(\varphi_1 \wedge \varphi_2)$	$\neg\varphi_1$	$\neg\varphi_2$
$\langle\langle A\rangle\rangle(\varphi_1 \mathsf{U} \varphi_2)$	φ_2	$\varphi_1 \wedge \langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle(\varphi_1 \mathsf{U} \varphi_2)$
$\neg\langle\langle A\rangle\rangle(\varphi_1 \mathsf{U} \varphi_2)$	$\neg\varphi_1 \wedge \neg\varphi_2$	$\neg\varphi_2 \wedge \neg\langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle(\varphi_1 \mathsf{U} \varphi_2)$
$\neg\langle\langle A\rangle\rangle\Box\varphi$	$\neg\varphi$	$\neg\langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\Box\varphi$

Table 3.1: Decomposition of α-formulae and β-formulae for ATL

3.2.1 Decomposition of ATL Formulae

The preliminary step towards prestate saturation and the rule **SR** consists in decomposing semantically complex ATL formulae into simpler ones. The simplest ATL formulae we can obtain are called *primitive formulae* (or *basic formulae*) and correspond to $\top$, p, $\neg p$, $\langle\langle A\rangle\rangle\bigcirc\varphi$ and $\neg\langle\langle A'\rangle\rangle\bigcirc\psi$ where $p \in \mathbb{P}$ and $A' \neq \mathbb{A}$. Formulae of the form $\langle\langle A\rangle\rangle\bigcirc\varphi$ and $\neg\langle\langle A\rangle\rangle\bigcirc\psi$ are called *positive successor formulae* and *proper negative successor formulae* respectively. Moreover, φ and $\neg\psi$ are called the *positive successor component* and *negative successor component*, respectively. These successor formulae have a major role in the dynamic stage of the construction phase, see Subsection 3.2.3.

Non-primitive formulae are partitioned in two categories: α-formulae and β-formulae. We will see in Chapter 4 a third category for ATL$^+$ and ATL* formulae. The α-formulae correspond to formulae for which the decomposition is conjunctive ($\alpha \equiv \alpha_1 \wedge \alpha_2$) and the β-formulae to the ones for which the decomposition is disjunctive ($\beta \equiv \beta_1 \vee \beta_2$). Decomposition of α-formulae is called *α-decomposition* and decomposition of β-formulae is called *β-decomposition*. Both kinds of decomposition are given in the table 3.1.

Each formula appearing in a tableau node whose initial formula is θ belongs to the so called closure of θ:

Closure Let θ be an ATL formula. The closure of θ, denoted by $\mathsf{cl}(\theta)$, is the least set of formulae such that

- $\theta \in \mathsf{cl}(\theta)$;
- $\mathsf{cl}(\theta)$ is closed under sub-formulae;
- $\mathsf{cl}(\theta)$ is closed under α-decomposition and β-decomposition;
- if $\varphi \in \mathsf{cl}(\theta)$ then $\varphi, \neg\varphi \in \mathsf{cl}(\theta)$, and $\neg\neg\varphi$ is always replaced by φ;
- $\top, \langle\langle\mathbb{A}\rangle\rangle\bigcirc\top \in \mathsf{cl}(\theta)$.

Let us observe that, for any θ, $\mathsf{cl}(\theta)$ is finite and $\mathsf{cl}(\theta) < c.|\theta|$ where $c \geq 1$ and $|\theta|$ is the size of θ [31].

Example 3.1. The closure of θ_1 (Formula 3.1) is

$$\begin{aligned}\mathsf{cl}(\theta_1) = \{&\theta_1, \langle\langle H\rangle\rangle \bigcirc l, \neg\langle\langle H\rangle\rangle \bigcirc l, l, \neg l, \neg\langle\langle H,B\rangle\rangle \Diamond(p \wedge r), \langle\langle H,B\rangle\rangle \Diamond(p \wedge r),\\ &\neg\langle\langle H,B\rangle\rangle \bigcirc \langle\langle H,B\rangle\rangle \Diamond(t \wedge r), \langle\langle H,B\rangle\rangle \bigcirc \langle\langle H,B\rangle\rangle \Diamond(p \wedge r), p \wedge r, \neg(p \wedge r),\\ &\neg\langle\langle H\rangle\rangle \bigcirc p, \langle\langle H\rangle\rangle \bigcirc p, \neg\langle\langle B\rangle\rangle \bigcirc r, \langle\langle B\rangle\rangle \bigcirc r, p, r, \neg p, \neg r\}\end{aligned}$$

and the closure of the θ_2 (Formula 3.2) is

$$\begin{aligned}\mathsf{cl}(\theta_2) = \{&\theta_2, \langle\langle H\rangle\rangle \Box\neg r, \neg\langle\langle H\rangle\rangle \Box\neg r, \langle\langle H\rangle\rangle \bigcirc \langle\langle H\rangle\rangle \Box\neg r, \neg\langle\langle H\rangle\rangle \bigcirc \langle\langle H\rangle\rangle \Box\neg r,\\ &\langle\langle B\rangle\rangle \Diamond r, \neg\langle\langle B\rangle\rangle \Diamond r, \langle\langle B\rangle\rangle \bigcirc \langle\langle B\rangle\rangle \Diamond r, \neg\langle\langle B\rangle\rangle \bigcirc \langle\langle B\rangle\rangle \Diamond r, r, \neg r\}\end{aligned}$$

3.2.2 Saturation of Prestates

Once we are able to decompose every non-primitive ATL formulae, it is possible to saturate a given set of ATL formulae using the following definition:

Definition 3.1 (full saturated sets of ATL formulae)**.** Let Γ, Δ be sets of ATL formulae and $\Gamma \subseteq \Delta \subseteq \mathsf{cl}(\Gamma)$.

1. Δ is *patently inconsistent* if it contains a pair of formulae φ and $\neg\varphi$.
2. Δ is a *full saturated set of* Γ if it is not patently inconsistent and satisfies the following closure conditions:
   - if $\varphi \wedge \psi \in \Delta$ then $\varphi \in \Delta$ and $\psi \in \Delta$;
   - if $\varphi \vee \psi \in \Delta$ then $\varphi \in \Delta$ or $\psi \in \Delta$;

The family of all full saturated sets of a set Γ is denoted $\mathsf{FS}(\Gamma)$.

The rule SR adds to the tableau all the full saturated sets of a prestate Γ as successor states of Γ, avoiding duplicated states.

Rule SR: Given a prestate Γ, do the following:

1. For each full saturated set Δ of Γ add to the initial tableau a state with label Δ.
2. For each of the added states Δ, if Δ does not contain any formulae of the form $\langle\langle A\rangle\rangle \bigcirc \varphi$ or $\neg\langle\langle A\rangle\rangle \bigcirc \varphi$, add the formula $\langle\langle \mathbb{A}\rangle\rangle \bigcirc \top$ to it;
3. For each state Δ obtained at steps 1 and 2, link Γ to Δ via a $\Longrightarrow$ edge;
4. If, however, the pretableau already contains a state Δ' with label Δ, do not create another copy of it but only link Γ to Δ' via a $\Longrightarrow$ edge.

Example 3.2. We start the construction of the tableau for θ_1 (Formula 3.1) by the creation of the initial node $\Gamma_0 = \langle\langle H\rangle\rangle \bigcirc l \wedge \neg\langle\langle H,B\rangle\rangle \Diamond(p \wedge r) \wedge \neg\langle\langle H\rangle\rangle \bigcirc p \wedge \neg\langle\langle B\rangle\rangle \bigcirc r$. Then we apply the rule SR on Γ_0, which results in the successor states shown in Figure 3.2:

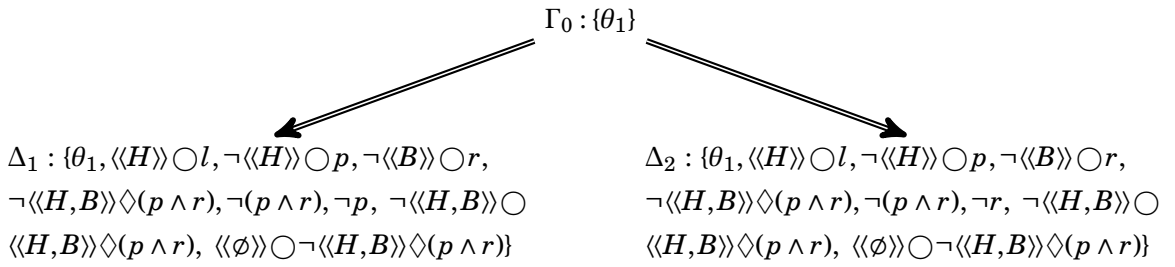


Figure 3.2: Application of the rule SR on prestate $\Gamma_0 = \{\theta_1\}$

3.2.3 Dynamic Analysis of Successor Formulae

In a tableau for LTL [66], the third construction rule creates one prestate from a state by keeping only $\bigcirc$-formulae (successor formulae) and removing their outermost $\bigcirc$, which gives a simple dynamic rule:

$$\frac{E, \bigcirc\varphi_1, \ldots, \bigcirc\varphi_n}{\varphi_1, \ldots, \varphi_1} \tag{3.3}$$

where E is a set of "marked formulae", that is formulae already statically analysed.

For ATL the problematic is different since transitions are labelled by action vectors and lead to different prestates. First, we provide each agent with enough available actions at the current state Δ. Then we appropriately define the outcome prestate of each action vector resulting from these actions.

First, we arrange all successor formulae of Δ into a list $\mathbb{L}$ where all formulae of the form $\langle\langle A\rangle\rangle\bigcirc\varphi$ (positive successor formulae) precede all formulae of the form $\neg\langle\langle A\rangle\rangle\bigcirc\varphi$ (proper negative successor formulae). The idea here is to consider the enforcing of each formula of $\mathbb{L}$ as an action available to every agent at the current state, the position of the formula being the number of that action. Each resulting action vector can be seen as a program encoding which successor components will be in the corresponding prestate. Note that in [31, 65], any action vector is seen as a "collective vote" made by all agents. This program (or "vote") ensures that for each $\langle\langle A\rangle\rangle\bigcirc\varphi$ from $\mathbb{L}$ there is a respective A-action at Δ that guarantees φ in the label of every corresponding successor prestate, and that for every $\neg\langle\langle A'\rangle\rangle\bigcirc\varphi$ from $\mathbb{L}$, there is an A'-co-action at Δ that ensures $\neg\varphi$ in the label of the corresponding successor prestates.

In order to have coherence between the action vectors and the properties we want to obtain on the linked prestates, the program selects successor components such that coalitions of the corresponding positive successor formulae and counter-coalition of the corresponding negative successor formulae do not intersect, as illustrated in Figure 3.3.

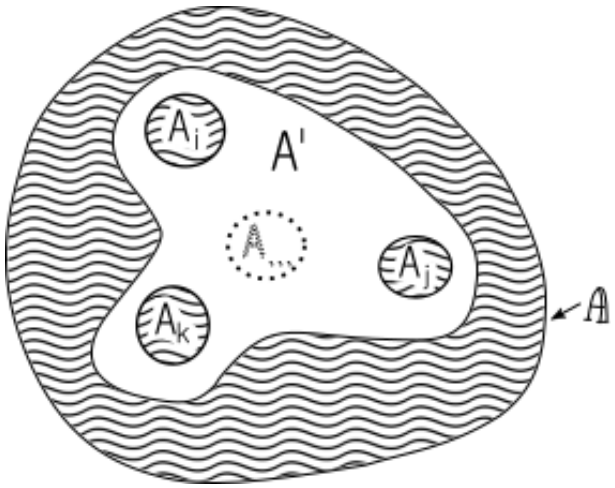


Figure 3.3: Coalitions A_i, A_j and A_k corresponding to positive successor formulae and the counter-coalition $\mathbb{A} - A'$ corresponding to the selected proper negative successor formula must not intersect.

This is why the rule **Next** ensures for any successor prestate Γ of Δ that

- at most one negative successor formula is present in any successor prestate;
- if $\{\langle\langle A_i\rangle\rangle\bigcirc\varphi_i, \langle\langle A_j\rangle\rangle\bigcirc\varphi_j\} \subseteq \Delta$ and $\{\varphi_i, \varphi_j\} \subseteq \Gamma$, then $A_i \cap A_j = \emptyset$;
- if $\{\langle\langle A_i\rangle\rangle\bigcirc\varphi_i, [\![A']\!]\bigcirc\psi\} \subseteq \Delta$ and $\{\varphi_i, \psi\} \subseteq \Gamma$, then $A_i \subseteq A'$.

(See [31], Remark 4.3)

Selection of a positive successor formula $\langle\langle A\rangle\rangle\bigcirc\varphi$ in a given prestate is pretty natural: the rule **Next** must ensure that there is at least one A-action enforcing each positive formula of the form $\langle\langle A\rangle\rangle\bigcirc\varphi$. This A-action is composed for each agent of the coalition of the action number attributed to $\langle\langle A\rangle\rangle\bigcirc\varphi$ in $\mathbb{L}$. Therefore if an action vector $\sigma_{\mathbb{A}}$ extends such an A-action, then φ is added to the prestate corresponding to $\sigma_{\mathbb{A}}$.

Selection of the proper negative successor formula $\neg\langle\langle A'\rangle\rangle\bigcirc\varphi$ is trickier and more technical: first, all the agents of the counter-coalition, that is agents belonging to $\mathbb{A} - A'$, must choose an action corresponding to a proper negative successor formula. This allows a given function co, defined below, to distribute the different negative successor components over each prestate resulting from action vectors corresponding to such criteria. For each successor formula $\neg\langle\langle A'\rangle\rangle\bigcirc\psi$, and for each A'-action $\sigma_{A'}$ possible from Δ, the agents of the counter-coalition can synchronise to extend $\sigma_{A'}$ in a dedicated action vector $\sigma_{\mathbb{A}}$ such that $\Delta \xrightarrow{\sigma_{\mathbb{A}}} \Gamma$ and ψ is the only negative component in Γ. Indeed, for each coalition A', there exists a player $b \in \mathbb{A} - A'$ that is able to enforce, no matter the choices of the agents in A', the value of co to the number corresponding to a given successor formula $\neg\langle\langle A'\rangle\rangle\bigcirc\psi \in \mathbb{L}$. Then all the other agents of the counter-coalition choose the first proper negative successor formula of the list $\mathbb{L}$. In this way, their choices correspond to add 0 in the computation of the function co, making unchanged the result decided by the agent b.

Rule Next Given a state Δ, do the following, where σ is a shorthand for $\sigma_{\mathbb{A}}$:

1. List all primitive successor formulae of Δ in such a way that all positive successor formulae precede all proper negative ones; let the result be the list

$$\mathbb{L} = [\langle\langle A_0\rangle\rangle\bigcirc\varphi_0, \ldots, \langle\langle A_{m-1}\rangle\rangle\bigcirc\varphi_{m-1}, \neg\langle\langle A'_0\rangle\rangle\bigcirc\psi_0, \ldots, \neg\langle\langle A'_{l-1}\rangle\rangle\bigcirc\psi_{l-1}]$$

   We recall that $A'_j \neq \mathbb{A}$ for all $0 \leq j \leq l-1$ by definition of primitive formulae. We also recall that in case of tight satisfiability $\mathbb{A} = \mathbb{A}_\theta$, and that in case of loose satisfiability $\mathbb{A} = \mathbb{A}_\theta \cup \{k+1\}$, assuming there are k agents in $\mathbb{A}_\theta$.
   Let $r_\Delta = m + l$; denote by $D(\Delta)$ the set $\{0, \ldots, r_\Delta - 1\}^{|\mathbb{A}|}$. Then, for every $\sigma \in D(\Delta)$, denote $N(\sigma) := \{i \mid \sigma_i \geqslant m\}$, where σ_i is the ith component of the tuple σ, and let $\mathsf{co}(\sigma) := [\Sigma_{i\in N(\sigma)}(\sigma_i - m)] \mod l$.
2. For each $\sigma \in D(\Delta)$ create a prestate:

$$\begin{aligned}\Gamma_\sigma = {} & \{\varphi_p \mid \langle\langle A_p\rangle\rangle\bigcirc\varphi_p \in \Delta \text{ and } \sigma_{\mathsf{a}} = p \text{ for all } \mathsf{a} \in A_p\} \\ & \cup \{\neg\psi_q \mid \neg\langle\langle A'_q\rangle\rangle\bigcirc\psi_q \in \Delta, \mathsf{co}(\sigma) = q \text{ and } \mathbb{A} - A'_q \subseteq N(\sigma)\}\end{aligned} \tag{3.4}$$

   If Γ_σ is empty, add $\top$ to it. Then connect Δ to Γ_σ with $\xrightarrow{\sigma}$.
   If, however, $\Gamma_\sigma = \Gamma$ for some prestate Γ that has already been added to the initial tableau, only connect Δ to Γ with $\xrightarrow{\sigma}$.

We repeat iteratively the rule **SR** and the rule **Next** until no new state or prestate can be added to the structure. The so obtained structure is called the *initial tableau for the formula* θ, and is denoted by $\mathcal{T}_0^\theta$.

Example 3.3 (Continuation of Example 3.2)**.** For both states Δ_1 and Δ_2, the list of successor formulae is the following:

$$\mathbb{L} = [\langle\langle \overset{0}{H}\rangle\rangle\bigcirc l,\ \langle\langle\emptyset\rangle\rangle\neg\langle\langle \overset{1}{H},\overset{1}{B}\rangle\rangle\Diamond(p\wedge r),\ \neg\langle\langle \overset{0}{H}\rangle\rangle\bigcirc p,\ \neg\langle\langle \overset{1}{B}\rangle\rangle\bigcirc r]$$
$$\mathbb{L} = [\langle\langle \underset{2}{\overset{0}{H}}\rangle\rangle\bigcirc l,\ \langle\langle\emptyset\rangle\rangle\neg\langle\langle \overset{1}{H},\overset{1}{B}\rangle\rangle\Diamond(p\wedge r),\ \neg\langle\langle \underset{2}{\overset{0}{H}}\rangle\rangle\bigcirc p,\ \neg\langle\langle \underset{3}{\overset{1}{B}}\rangle\rangle\bigcirc r]$$

Note that the numbers on the upper line correspond to positions among proper negative successor formulae, and the numbers on the lower line correspond to positions among all successor formulae.

So, $m = 2$, $l = 2$, $r_{\Delta_1} = r_{\Delta_2} = 4$, and

σ	$N(\sigma)$	$\mathsf{co}(\sigma)$	Γ_σ	σ	$N(\sigma)$	$\mathsf{co}(\sigma)$	Γ_σ
0,0	$\emptyset$	0	l, $\neg\langle\langle H,B\rangle\rangle\Diamond(p\wedge r)$	2,0	$\{H\}$	0	$\neg\langle\langle H,B\rangle\rangle\Diamond(p\wedge r)$
0,1	$\emptyset$	0	l, $\neg\langle\langle H,B\rangle\rangle\Diamond(p\wedge r)$	2,1	$\{H\}$	0	$\neg\langle\langle H,B\rangle\rangle\Diamond(p\wedge r)$
0,2	$\{B\}$	0	l, $\neg p$, $\neg\langle\langle H,B\rangle\rangle\Diamond(p\wedge r)$	2,2	$\{H,B\}$	0	$\neg p$, $\neg\langle\langle H,B\rangle\rangle\Diamond(p\wedge r)$
0,3	$\{B\}$	1	l, $\neg\langle\langle H,B\rangle\rangle\Diamond(p\wedge r)$	2,3	$\{H,B\}$	1	$\neg r$, $\neg\langle\langle H,B\rangle\rangle\Diamond(p\wedge r)$
1,0	$\emptyset$	0	$\neg\langle\langle H,B\rangle\rangle\Diamond(p\wedge r)$	3,0	$\{H\}$	1	$\neg r$, $\neg\langle\langle H,B\rangle\rangle\Diamond(p\wedge r)$
1,1	$\emptyset$	0	$\neg\langle\langle H,B\rangle\rangle\Diamond(p\wedge r)$	3,1	$\{H\}$	1	$\neg r$, $\neg\langle\langle H,B\rangle\rangle\Diamond(p\wedge r)$
1,2	$\{B\}$	0	$\neg p$, $\neg\langle\langle H,B\rangle\rangle\Diamond(p\wedge r)$	3,2	$\{H,B\}$	1	$\neg r$, $\neg\langle\langle H,B\rangle\rangle\Diamond(p\wedge r)$
1,3	$\{B\}$	1	$\neg\langle\langle H,B\rangle\rangle\Diamond(p\wedge r)$	3,3	$\{H,B\}$	0	$\neg p$, $\neg\langle\langle H,B\rangle\rangle\Diamond(p\wedge r)$

which lead to five successor prestates for both Δ_1 and Δ_2:

$\Gamma_1 : \{l,\ \neg\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r)\}$ $\quad$ $\Gamma_2 : \{l,\ \neg p,\ \neg\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r)\}$

$\Gamma_3 : \{\neg\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r)\}$ $\quad$ $\Gamma_4 : \{\neg p,\ \neg\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r)\}$

$\Gamma_5 : \{\neg r,\ \neg\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r)\}$

By applying two more times the rules SR and Next, we obtain the initial tableau of Figure 3.4.

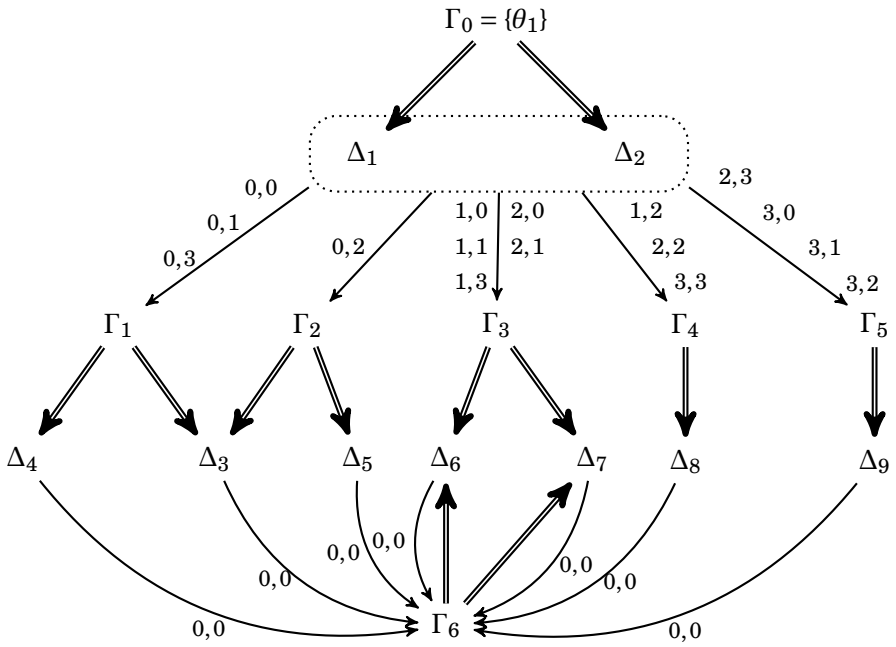


Figure 3.4: Initial tableau for $\theta_1 = \langle\langle H\rangle\rangle\bigcirc l \wedge \neg\langle\langle H,B\rangle\rangle\Diamond(p \wedge r) \wedge \neg\langle\langle H\rangle\rangle\bigcirc p \wedge \neg\langle\langle B\rangle\rangle\bigcirc r$

with Δ_1 and Δ_2 as in Figure 3.2 and:

$\Delta_3 : \{l,\ \neg\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r),\ \neg(p \wedge r),\ \neg p,\ \neg\langle\langle 1,2\rangle\rangle\bigcirc\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r), \langle\langle\emptyset\rangle\rangle\neg\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r)\}$

$\Delta_4 : \{l,\ \neg\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r),\ \neg(p \wedge r),\ \neg r,\ \neg\langle\langle 1,2\rangle\rangle\bigcirc\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r), \langle\langle\emptyset\rangle\rangle\neg\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r)\}$

$\Delta_5 : \{l,\ \neg p,\ \neg\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r),\ \neg(p \wedge r),\ \neg r,\ \neg\langle\langle 1,2\rangle\rangle\bigcirc\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r), \langle\langle\emptyset\rangle\rangle\neg\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r)\}$

$\Delta_6 : \{\neg\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r),\ \neg(p \wedge r),\ \neg p,\ \neg\langle\langle 1,2\rangle\rangle\bigcirc\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r), \langle\langle\emptyset\rangle\rangle\neg\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r)\}$

$\Delta_7 : \{\neg\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r),\ \neg(p \wedge r),\ \neg r,\ \neg\langle\langle 1,2\rangle\rangle\bigcirc\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r), \langle\langle\emptyset\rangle\rangle\neg\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r)\}$

$\Delta_8 : \{\neg p,\ \neg\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r),\ \neg(p \wedge r),\ \neg r,\ \neg\langle\langle 1,2\rangle\rangle\bigcirc\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r), \langle\langle\emptyset\rangle\rangle\neg\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r)\}$

$\Delta_9 : \{\neg r,\ \neg\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r),\ \neg(p \wedge r),\ \neg p,\ \neg\langle\langle 1,2\rangle\rangle\bigcirc\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r), \langle\langle\emptyset\rangle\rangle\neg\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r)\}$

$\Gamma_6 : \{\neg\langle\langle 1,2\rangle\rangle\Diamond(p \wedge r)\}$

3.3 Elimination Phase

As we have already observed, $\mathscr{T}_0^\theta$ is a finite graph. The *elimination phase* applies to $\mathscr{T}_0^\theta$ and also works step-by-step. In order to go through one step to another we apply by turns two elimination rules, called **ER1** and **ER2**, until no more nodes can be eliminated. The rule **ER1** detects and deletes nodes with missing successors, while the rule **ER2** detects and deletes states that do not realize all their eventualities. At each step, we obtain a new intermediate tableau, denoted by $\mathscr{T}_n^\theta$. We denote by S_n^θ the set of nodes (states and prestates) of the intermediate tableau $\mathscr{T}_n^\theta$.

Remark 3.1. Contrary to the tableau-based decision procedure in [31], we present a version where prestates are eliminated with the rule **ER1** only when necessary. This does not have any effect on the result of the procedure, nor any relevant modification in the soundness and completeness proofs, but it makes implementation quicker and easier.

Rule ER1 Let $\Xi \in S_n^\theta$ be a node (prestate or state).

- In the case where Ξ is a prestate: if all nodes Δ with $\Xi \Longrightarrow \Delta$ have been eliminated at earlier stages, then obtain $\mathscr{T}_{n+1}^\theta$ by eliminating Ξ from $\mathscr{T}_n^\theta$.
- In the case where Ξ is a state: if, for some $\sigma \in D(\Xi)$, the node Γ with $\Xi \xrightarrow{\sigma} \Gamma$ has been eliminated at earlier stage, then obtain $\mathscr{T}_{n+1}^\theta$ by eliminating Ξ from $\mathscr{T}_n^\theta$.

Before stating the rule **ER2**, we first define what is an eventuality and how to check that eventualities are realized.

Realization of eventualities In *vanilla*-ATL, eventualities are formulae of the form $\langle\langle A\rangle\rangle\varphi \mathsf{U} \psi$ or of the form $\neg\langle\langle A\rangle\rangle\Box\varphi$. With this kind of formulae, one can express properties that must occur eventually, even if we don't know when. The danger when we construct a tableau, is to postpone the moment when the property holds forever. Therefore, if an eventuality is not immediately realized, we need to check in the future whether it is indeed realized or not. In order to obtain the set of successor prestates (and therefore the associated successor states) involved in the realization of a given eventuality, we introduce the following notation:

Notation 3.1. Let $\Delta \in S_n^\theta$ and let $\mathbb{L} = [\langle\langle A_0\rangle\rangle\bigcirc\varphi_0, \ldots, \langle\langle A_{m-1}\rangle\rangle\bigcirc\varphi_{m-1}, \neg\langle\langle A'_0\rangle\rangle\bigcirc\theta_0, \ldots, \neg\langle\langle A'_{l-1}\rangle\rangle\bigcirc\theta_{l-1}]$ be the list of all primitive successor formulae of Δ, induced as part of application of **(Next)**.
$Succ(\Delta, \langle\langle A_p\rangle\rangle\bigcirc\varphi_p) := \{\Gamma \mid \Delta \xrightarrow{\sigma} \Gamma, \sigma_{\mathsf{a}} = p \text{ for every } \mathsf{a} \in A_p\}$
$Succ(\Delta, \neg\langle\langle A'_q\rangle\rangle\bigcirc\theta_q) := \{\Gamma \mid \Delta \xrightarrow{\sigma} \Gamma, \mathsf{co}(\sigma) = q \text{ and } \mathbb{A} - A'_q \subseteq N(\sigma)\}$

Definition 3.2 (Realization of eventuality of the form $\langle\langle A\rangle\rangle\varphi \mathsf{U}\psi$)**.** The eventuality $\langle\langle A\rangle\rangle\varphi \mathsf{U}\psi$ is realized at Δ in $\mathscr{T}_n^\theta$ when:

1. If $\{\psi, \langle\langle A\rangle\rangle\varphi \mathsf{U}\psi\} \subseteq S_n^\theta$, then $\langle\langle A\rangle\rangle\varphi \mathsf{U}\psi$ is realized at Δ in $\mathscr{T}_n^\theta$; or
2. If $\{\varphi, \langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\varphi \mathsf{U}\psi, \langle\langle A\rangle\rangle\varphi \mathsf{U}\psi\} \subseteq \Delta$ and for every $\Gamma \in Succ(\Delta, \langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\varphi \mathsf{U}\psi)$, there exists $\Delta' \in S_n^\theta$ such that

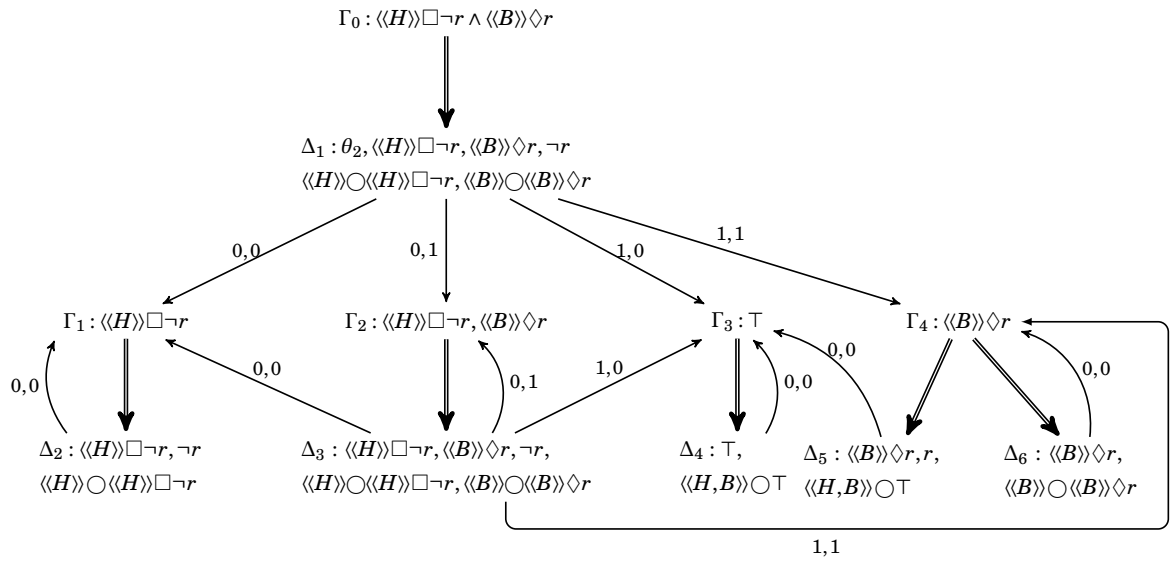


Figure 3.5: Initial tableau for $\theta_2 = \langle\langle H\rangle\rangle\Box\neg r \wedge \langle\langle B\rangle\rangle\Diamond r$

- $\Gamma \Longrightarrow \Delta'$ and,
- $\langle\langle A\rangle\rangle\varphi \mathsf{U} \psi$ is realized at Δ' in $\mathscr{T}_n^\theta$,

Definition 3.3 (Realization of eventuality of the form $\neg\langle\langle A\rangle\rangle\Box\varphi$)**.** The eventuality $\neg\langle\langle A\rangle\rangle\Box\varphi$ is realized at Δ in $\mathscr{T}_n^\theta$ when:

1. If $\{\neg\varphi, \neg\langle\langle A\rangle\rangle\Box\varphi\} \subseteq S_n^\theta$, then $\neg\langle\langle A\rangle\rangle\Box\varphi$ is realized at Δ in $\mathscr{T}_n^\theta$; or
2. If $\{\neg\langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\Box\varphi, \neg\langle\langle A\rangle\rangle\Box\varphi\} \subseteq \Delta$ and for every $\Gamma \in Succ(\Delta, \neg\langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\Box\varphi)$, there exists $\Delta' \in S_n^\theta$ such that
   - $\Gamma \Longrightarrow \Delta'$ and,
   - $\neg\langle\langle A\rangle\rangle\Box\varphi$ is realized at Δ' in $\mathscr{T}_n^\theta$,

Rule ER2 If $\Delta \in S_n^\theta$ is a state and contains an eventuality that is not realized at $\Delta \in \mathscr{T}_n^\theta$, then obtain $\mathscr{T}_{n+1}^\theta$ by removing Δ from S_n^θ.

Example 3.4 (Elimination phase for θ_2)**.** The initial tableau for the formula θ_2 (Formula 3.2) is given in Fig. 3.5. The state Δ_1 of the initial tableau for θ_2 contains one eventuality, namely $\langle\langle B\rangle\rangle\Diamond r$. The state Δ_1 does not contain the proposition r, therefore, the eventuality $\langle\langle B\rangle\rangle\Diamond r$ is not immediately realized in Δ. The set of successors of Δ_1 for this eventuality is $Succ(\Delta_1, \langle\langle B\rangle\rangle\bigcirc \langle\langle B\rangle\rangle\Diamond r) = \{\Gamma_2, \Gamma_4\}$.

Let us study Γ_2. The successor of Γ_2 is Δ_3, which does not contain the proposition r either. This means that the eventuality $\langle\langle B\rangle\rangle\Diamond r$ is not immediately realized in Δ_3. Moreover, $Succ(\Delta_3, \langle\langle B\rangle\rangle\bigcirc \langle\langle B\rangle\rangle\Diamond r) = \{\Gamma_2, \Gamma_4\}$ and Δ_3 is the only successor of Γ_2, so we conclude that the eventuality $\langle\langle B\rangle\rangle\Diamond r$ cannot be realized from Δ_1 or Δ_3. Therefore, according to rule **ER2**, we eliminate these two states, which implies, according to rule **ER1**, the elimination of Γ_2 and Γ_0.

At the end of the elimination phase, we obtain the *final tableau* for θ, denoted by $\mathscr{T}^\theta$. The set of nodes (prestates and states) in $\mathscr{T}^\theta$ is denoted by S^θ. It is declared *open* if the initial node belongs to S^θ, and *closed* otherwise . The procedure for deciding satisfiability of θ returns "No" if $\mathscr{T}^\theta$ is closed, "Yes" otherwise.

Example 3.5 (Continuation of Examples 3.4 and 3.3)**.** The initial node Γ_0 of the tableau for θ_2 has been eliminated during the elimination phase. Therefore, the tableau is closed and the formula θ_2 is unsatisfiable.

On the other hand, no node can be eliminated from the initial tableau for θ_1, so the tableau is open and θ_1 is satisfiable.

3.4 Conclusion

In our opinion, this tableau-based decision procedure is relatively intuitive. Indeed it follows the semantic of ATL formulae, and it is possible to easily represent the resulting tableau for small formulae. This tableau-based decision procedure for ATL is sound, complete and runs in EXPTIME[31]. In next chapter, we will see that it is possible to extend this procedure for deciding satisfiability of ATL^+ and ATL^* without distorting the underlying basic structure. These new procedures are also sound and complete, but runs in 2EXPTIME, which is the optimal complexity.

Part II

Deciding ATL$^+$ and ATL* Satisfiability by Tableaux

CHAPTER 4

TABLEAU-BASED DECISION PROCEDURES FOR ATL$^+$ AND ATL*

In Chapter 3 we have described the tableau-based decision procedure for ATL introduced in [31]. However, with the syntax of ATL, one cannot express properties where several objectives have to be achieved by a same strategy, or fairness constraints, for instance.

The extensions ATL$^+$ and ATL* of ATL allow one to express such properties, but in return, bring additional difficulties.

First, as seen in Subsection 2.2.4, memoryless strategies and perfect-recall strategies are not equivalent, regarding the class of satisfiable formulae. Here, we present a tableau procedure using perfect-recall strategies. In the conclusion of this chapter, we discuss why it is difficult, not to say impossible, to build a tableau using memoryless strategies.

Also, the possibility to use Boolean combination and nesting of temporal operators complicates the decomposition of formulae; fixed-point equivalences cannot be used directly to decompose formulae containing temporal properties, as done for the ATL tableau procedure. This is because the strategic quantifiers $\langle\langle A\rangle\rangle$ and $[\![A]\!]$ cannot distribute over Boolean connectors. For example, $\langle\langle A\rangle\rangle(\Box\Phi_1 \vee \Box\Phi_2) \not\equiv \langle\langle A\rangle\rangle\Box\Phi_1 \vee \langle\langle A\rangle\rangle\Box\Phi_2$.

Last, it is not clear whether a formula is an eventuality or not. For instance, should we consider the formula $\langle\langle A\rangle\rangle(\Box\varphi \vee \Diamond\psi)$ as an eventuality? Once having decided which formulae are eventualities, it then remains the problem of determining when an eventuality is realized.

Nevertheless, it is possible to deal with all these difficulties inherent to ATL$^+$ and ATL*, and construct a tableau for formulae of these extensions without modifying the general structure of the tableau-based decision procedure for ATL. Therefore, we present our tableau procedures for ATL$^+$ and ATL* by highlighting the differences with respect to the procedure presented in Chapter 3. These differences are outlined in Table 4.1 and treated in detail in the rest of this chapter. We end by analysing the complexity of these procedures. We have figured out the two

tableau procedures incrementally by first solving problems in ATL$^+$ [13], and then dealing with problems only occurring in ATL* [20].

In order to illustrate our new procedures, let us first modify the example formula θ_1 into the following ATL$^+$ formula:

$$\theta_1^+ = \langle\langle H\rangle\rangle \bigcirc \text{logged} \wedge \neg\langle\langle H,B\rangle\rangle(\Diamond \text{ticket\_printed} \wedge \Diamond \text{ticket\_reimbursed}) \wedge \neg\langle\langle H\rangle\rangle(\bigcirc \text{ticket\_printed} \vee \bigcirc \text{ticket\_reimbursed})$$

which means that Hugo can log at next step, that Hugo and Bob are not sure to have the possibility to print and reimburse a ticket, and that Hugo is not sure to be able to print or reimburse a ticket at next step. We abbreviate and put into negation normal form θ_1^+, and we obtain:

$$\theta_1^+ = \langle\langle H\rangle\rangle \bigcirc l \wedge [\![H,B]\!](\Box\neg p \vee \Box\neg r) \wedge [\![H]\!](\bigcirc\neg p \wedge \bigcirc\neg r) \tag{4.1}$$

We also take as an additional example the ATL$^+$ formula θ_3^+:

$$\theta_3^+ = \langle\langle H\rangle\rangle(((\neg \text{ticket\_printed} \wedge \neg \text{ticket\_reimbursed}) \mathsf{U} \text{logged}) \wedge \Diamond \text{ticket\_reimbursed}) \wedge \langle\langle B\rangle\rangle\Box\neg \text{logged}$$

which means that Hugo has a strategy to not be able to print or reimburse a ticket until being logged and eventually reimburse a ticket, and Bob has a strategy to never be logged. This unsatisfiable formula is abbreviated as

$$\theta_3^+ = \langle\langle H\rangle\rangle(((\neg p \wedge \neg r)\mathsf{U} l) \wedge \Diamond r) \wedge \langle\langle B\rangle\rangle\Box\neg l \tag{4.2}$$

For ATL*, we take as example the formula θ_3^*:

$$\theta_3^* = \langle\langle H\rangle\rangle(\neg \text{logged} \mathsf{U}(\text{logged} \wedge \Diamond\Box \text{ticket\_unavailable})) \wedge \langle\langle B\rangle\rangle\Box\neg \text{ticket\_unavailable}$$

which means that Hugo has a strategy to be unlogged until being logged and eventually get to a point where the ticket is unavailable for ever, and Bob has a strategy to always have the ticket available. This unsatisfiable formula is abbreviated as

$$\theta_3^* = \langle\langle H\rangle\rangle(\neg l \mathsf{U}(l \wedge \Diamond\Box u)) \wedge \langle\langle B\rangle\rangle\Box\neg u \tag{4.3}$$

We also use the following formula, extracted and adapted from [57]:

$$\theta_4^* = [\![1]\!]\Box((p \wedge \bigcirc\neg p) \vee (\neg p \wedge \bigcirc p)) \wedge [\![1]\!]\Box(\neg p \vee \neg q) \wedge [\![1]\!]\Box(\neg p \vee \neg r) \wedge [\![1]\!]\Box(\neg q \vee \neg r) \wedge [\![1]\!]\Box(\Diamond q \wedge \Diamond r) \wedge q \tag{4.4}$$

This formula shows interesting properties linked to the simultaneous nesting and Boolean combination of temporal operators, that have an impact on the elimination phase.

The proofs concerning our procedure for deciding the satisfiability of ATL$^+$ formulae can be found in [14, 15]. In appendix B, we give the proofs of our tableau procedure for ATL* formulae.

Step	ATL$^+$	ATL*
Decomposition	treatment of Boolean combination of temporal operators: new kind of formulae, namely γ-formulae, to be decomposed	treatment of Boolean combination and nesting of temporal operators: new kind of formulae, namely γ-formulae, to be decomposed
Saturation	take into account γ-formulae	
rule **SR**	no modifications	
rule **Next**	take into account negation normal form	
rule **ER1**	No modifications	
Realization of eventualities	treatment of γ-formulae as eventualities: new function (Realized) to compute immediate realization of eventualities + link between eventualities	treatment of γ-formulae as eventualities: new function (WF) to compute immediate realization of eventualities + link between eventualities
rule **ER2**	no modifications	

Table 4.1: Modification for the adaptation of the tableau-based decision procedure for ATL to the extensions ATL$^+$ and ATL*

4.1 New Kind of Formulae = New Decomposition

In section 3.2.1, we have seen that we can partition ATL formulae into primitive formulae, α-formulae and β-formulae. In ATL$^+$ and ATL*, we encounter formulae of the form $\langle\langle A\rangle\rangle\Phi$ or $[\![A]\!]\Phi$ where Φ can be as complicated as we want, with the limitation that they do not have $\bigcirc$ as main operator, and for ATL$^+$ they do not contain nesting of temporal operators. Therefore, we are not able to describe all the possibilities of decomposition of non-primitive formulae as we did with Table 3.1. We need a specific way to decompose this new kind of formulae, that we call *γ-formulae*. Nevertheless, we still keep the notion of *primitive formulae*, which corresponds to the formulae $\top$, $\bot$, p, $\neg p$, $\langle\langle A\rangle\rangle\bigcirc\varphi$ and $[\![A]\!]\bigcirc\varphi$, where p is a proposition and φ is a state formulae, as well as the notion of α-formulae, which corresponds to the formulae of the from $\varphi_1 \wedge \varphi_2$, and β-formulae, which corresponds to the formulae of the form $\varphi_1 \vee \varphi_2$, where φ_1 and φ_2 are state formulae. We then define γ-formulae as non-primitive formulae different from α- and β-formulae, that is formulae of the form $\langle\langle A\rangle\rangle\Phi$ and $[\![A]\!]\Phi$ where $\Phi \neq \bigcirc\varphi$.

Let us see how to decompose γ-formulae. To understand how it works, we need to return to the source of this type of tableaux for temporal logic, that is the tableaux for LTL described in [66]. We must remember that, for LTL, decomposition of temporal operators aims at having on one side "a requirement on the current state" and on the other side "a requirement on the rest of the sequence". Since ATL and its extensions are branching logics, we cannot keep talking about

sequences, but this kind of decomposition can be adapted to strategies. In our case, properties are evaluated on branches resulting from the execution of a given strategy. Thus our objective is to decompose γ-formulae in order to obtain the same separation between the current state and the *rest of the branches*. In other words, for each possibility embedded in the path formula Φ associated to a γ-formula of the form $\langle\langle A\rangle\rangle\Phi$ or $[\![A]\!]\Phi$, we want to obtain a pair $\langle\psi,\Psi\rangle$ where ψ is a state formula that represents a requirement on the current state, and Ψ is a path formula that represents a requirement on the rest of the branches. In this purpose, we propose two new functions $\mathsf{dec}^+ : \mathsf{ATL}^+_\mathsf{p} \to \mathscr{P}(\mathsf{ATL}^+_\mathsf{s} \times \mathsf{ATL}^+_\mathsf{p})$ and $\mathsf{dec}^* : \mathsf{ATL}^*_\mathsf{p} \to \mathscr{P}(\mathsf{ATL}^*_\mathsf{s} \times \mathsf{ATL}^*_\mathsf{p} \times \mathscr{P}(\mathsf{ATL}^*_\mathsf{p}))$, where $\mathsf{ATL}^+_\mathsf{p}$ is a set of ATL$^+$ path formulae, $\mathsf{ATL}^+_\mathsf{s}$ is a set of ATL$^+$ state formulae, $\mathsf{ATL}^*_\mathsf{p}$ is a set of ATL* path formulae, and $\mathsf{ATL}^*_\mathsf{s}$ is a set of ATL* state formulae. The definition of the function dec^* is slightly different from the one of dec^+. This difference will be explained in Section 4.1.2.

4.1.1 Decomposition Function for ATL$^+$ γ-Formulae

We define the function dec^+ recursively on the structure of the path formula Φ as follows:

- $\mathsf{dec}^+(\varphi) = \{\langle\varphi,\top\rangle\}$ and $\mathsf{dec}^+(\bigcirc\varphi) = \{\langle\top,\varphi\rangle\}$ for any ATL$^+$ state formula φ.
- $\mathsf{dec}^+(\Box\varphi) = \{\langle\varphi,\Box\varphi\rangle\}$
- $\mathsf{dec}^+(\varphi \mathsf{U} \psi) = \{\langle\varphi,\varphi \mathsf{U} \psi\rangle, \langle\psi,\top\rangle\}$
- $\mathsf{dec}^+(\Phi_1 \wedge \Phi_2) = \mathsf{dec}^+(\Phi_1) \otimes^+ \mathsf{dec}^+(\Phi_2)$, where
  $\mathscr{S}_1 \otimes^+ \mathscr{S}_2 := \{\langle\psi_i \wedge \psi_j, \Psi_i \wedge \Psi_j\rangle \mid \langle\psi_i,\Psi_i\rangle \in \mathscr{S}_1, \langle\psi_j,\Psi_j\rangle \in \mathscr{S}_2\}$
- $\mathsf{dec}^+(\Phi_1 \vee \Phi_2) = \mathsf{dec}^+(\Phi_1) \cup \mathsf{dec}^+(\Phi_2) \cup (\mathsf{dec}^+(\Phi_1) \oplus^+ \mathsf{dec}^+(\Phi_2))$, where
  $\mathscr{S}_1 \oplus^+ \mathscr{S}_2 := \{\langle\psi_i \wedge \psi_j, \Psi_i \vee \Psi_j\rangle \mid \langle\psi_i,\Psi_i\rangle \in \mathscr{S}_1, \langle\psi_j,\Psi_j\rangle \in \mathscr{S}_2, \Psi_i \neq \top, \Psi_j \neq \top\}$

where the operators $\otimes^+$ and $\oplus^+$ are associative, up to logical equivalence.

The first three items of the definition are directly derived from decomposition and fixed-point equivalences for LTL. The novelty comes from the treatment of Boolean connectors in path formulae. The conjunctive case is clear: every path that satisfies $\Phi_1 \wedge \Phi_2$ combines a type of path that satisfies Φ_1 with a type of path that satisfies Φ_2. To understand the disjunctive case, we recall that the construction of the tableau is step-by-step. Therefore, for a given prestate under construction, when we have a formula of the form $\langle\langle A\rangle\rangle(\Phi_1 \vee \Phi_2)$, where, for instance $\Phi_1 = \Box\varphi_1$ and $\Phi_2 = \Box\varphi_2$, we do not know in advance which of $\Box\varphi_1$ or $\Box\varphi_2$ would be completed for each possible path; so it is important to keep both possibilities at the current state, if possible. This idea is expressed by the use of $\mathsf{dec}^+(\Phi_1) \oplus^+ \mathsf{dec}^+(\Phi_2)$ in the above union, where we keep both disjuncts true at the present state and delay the choice. This is why, in $\oplus^+$ definition, the state formulae ψ_i and ψ_j are connected by $\wedge$ but the path formulae Ψ_i and Ψ_j are connected by $\vee$. Moreover, the $\oplus^+$ operation avoids the construction of a pair $\langle\psi_i \wedge \psi_j, \Psi_i \vee \Psi_j\rangle$ where either Ψ_i or Ψ_j is $\top$, because that case is already included in $\mathsf{dec}^+(\Phi_1)$ or in $\mathsf{dec}^+(\Phi_2)$. The three cases for paths that satisfy the disjunction $\Phi_1 \vee \Phi_2$ can be illustrated by the picture in Figure 4.1.

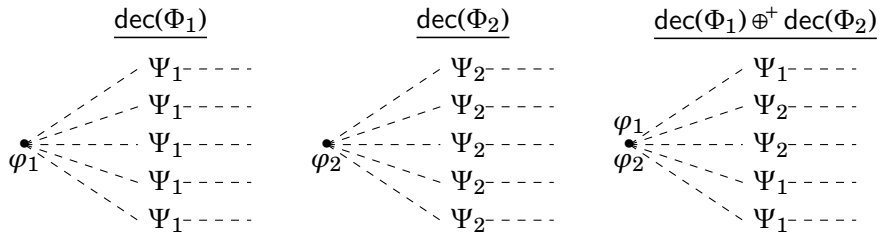


Figure 4.1: The three cases for disjunctive path objectives in a γ-formula

Example 4.1 (function dec^+ applied to $[\![H,B]\!](\Box\neg p \vee \Box\neg r)$)**.** For this example, the path formula to be studied is $\Phi = \Box\neg p \vee \Box\neg r$, therefore

$$\mathsf{dec}^+(\Phi) = \mathsf{dec}^+(\Box\neg p) \cup \mathsf{dec}^+(\Box\neg r) \cup (\mathsf{dec}^+(\Box\neg p) \oplus^+ \mathsf{dec}^+(\Box\neg r))$$
$$\mathsf{dec}^+(\Phi) = \{\langle\neg p, \Box\neg p\rangle\} \cup \{\langle\neg r, \Box\neg r\rangle\} \cup (\{\langle\neg p, \Box\neg p\rangle\} \oplus^+ \{\langle\neg r, \Box\neg r\rangle\})$$
$$\mathsf{dec}^+(\Phi) = \{\langle\neg p, \Box\neg p\rangle\} \cup \{\langle\neg r, \Box\neg r\rangle\} \cup \{\langle\neg p \wedge \neg r, \Box\neg p \vee \Box\neg r\rangle\}$$
$$\mathsf{dec}^+(\Phi) = \{\langle\neg p, \Box\neg p\rangle, \langle\neg r, \Box\neg r\rangle, \langle\neg p \wedge \neg r, \Box\neg p \vee \Box\neg r\rangle\}$$

4.1.2 Decomposition Function for ATL* γ-Formulae

The decomposition function dec^* is similar to the decomposition function dec^+, but we also need to go one step further into the recursion to deal with nesting of temporal operators. Therefore we need to analyse sub path formulae coming after temporal operators. This is why the two cases $\Box\Phi$ and $\Phi\mathsf{U}\Psi$ are now recursive steps in the definition of dec^*.

In temporal logics, e.g. LTL, the operator U is considered as an eventuality operator, that is an operator that promises to verify a given formula at some instant/state. When we write $\lambda \models \varphi_1 \mathsf{U} \varphi_2$, where φ_1 and φ_2 are state formulae, we mean that there is a state λ_i of the computation λ where φ_2 holds and φ_1 holds for all the states of λ preceding λ_i. So, once the property φ_2 is verified, we do not need to take care of φ_1, φ_2 and $\varphi_1 \mathsf{U} \varphi_2$ any more. We say that $\varphi_1 \mathsf{U} \varphi_2$ is *realized*. However, if φ_1 and φ_2 are path formulae, e.g. $\Box\Phi_1$ and $\Box\Phi_2$ respectively, there is a state λ_i from which Φ_2 must hold forever — we say that $\Box\Phi_2$ is "initiated" at λ_i, in the sense that we start to make $\Box\Phi_2$ true at λ_i —, and for every computation $\lambda_{\geq j}$, where $j < i$, $\Box\Phi_1$ must hold. So Φ_1 has to be true forever, that is even after $\Box\Phi_2$ had been initiated. This explains the fact that at a state s the path formula $\varphi_1 \mathsf{U} \varphi_2$ may become $\varphi_1 \mathsf{U} \varphi_2 \wedge \varphi_1$ when φ_1 is a path formula and we postpone φ_2. Note that φ_1 is then also initiated at s. We now face the problem of memorizing the fact that a path formula Φ is initiated. Indeed, path formulae cannot be stored directly in a state that, as for ATL, is a set of state formulae. In order to deal with this problem, during the decomposition of γ-formulae, we add a new set of path formulae linked to a γ-component and the current state.

In order to take into account this third element, the operators $\otimes^+$ and $\oplus^+$ are modified into the two operators $\otimes^*$ and $\oplus^*$, respectively, and whose definition is as follows:

- $\mathscr{S}_1 \otimes^* \mathscr{S}_2 := \{\langle \psi_i \dot{\wedge} \psi_j, \Psi_i \dot{\wedge} \Psi_j, S_i \cup S_j\rangle \mid \langle \psi_i, \Psi_i, S_i\rangle \in \mathscr{S}_1, \langle \psi_j, \Psi_j, S_j\rangle \in \mathscr{S}_2\}$
- $\mathscr{S}_1 \oplus^* \mathscr{S}_2 := \{\langle \psi_i \dot{\wedge} \psi_j, \Psi_i \dot{\vee} \Psi_j, S_i \cup S_j\rangle \mid \langle \psi_i, \Psi_i, S_i\rangle \in \mathscr{S}_1,\ \langle \psi_j, \Psi_j, S_j\rangle \in \mathscr{S}_2, \Psi_i \neq \top, \Psi_j \neq \top\}$

where $\otimes^*$ and $\oplus^*$ are associative, up to logical equivalence.

The definition of the function dec* is recursive on the structure of the path formula Φ:

- $\mathsf{dec}^*(\varphi) = \{\langle \varphi, \top, \emptyset\rangle\}$ for any ATL* state formula φ
- $\mathsf{dec}^*(\bigcirc\Phi_1) = \{\langle \top, \Phi_1, \emptyset\rangle\}$ for any path formula Φ_1
- $\mathsf{dec}^*(\Box\Phi_1) = \{\langle \top, \Box\Phi_1, \{\Phi_1\}\rangle\} \otimes^* \mathsf{dec}^*(\Phi_1)$
- $\mathsf{dec}^*(\Phi_1 \mathsf{U} \Phi_2) = (\{\langle \top, \Phi_1 \mathsf{U} \Phi_2, \{\Phi_1\}\rangle\} \otimes^* \mathsf{dec}^*(\Phi_1)) \cup (\{\langle \top, \top, \{\Phi_2\}\rangle\} \otimes^* \mathsf{dec}^*(\Phi_2))$
- $\mathsf{dec}^*(\Phi_1 \wedge \Phi_2) = \mathsf{dec}^*(\Phi_1) \otimes^* \mathsf{dec}^*(\Phi_2)$
- $\mathsf{dec}^*(\Phi_1 \vee \Phi_2) = \mathsf{dec}^*(\Phi_1) \cup \mathsf{dec}^*(\Phi_2) \cup (\mathsf{dec}^*(\Phi_1) \oplus^* \mathsf{dec}^*(\Phi_2))$

It is worth noticing that the function dec* subsumes the function dec$^+$.

Remark 4.1. The operators $\dot{\wedge}$ and $\dot{\vee}$ correspond respectively to the operators $\wedge$ and $\vee$ where the associativity, commutativity, idempotence and identity element properties are embedded in the syntax. The aim of both $\dot{\wedge}$ and $\dot{\vee}$ is to automatically transform resultant formulae in conjunctive normal form without redundancy, and therefore ensures the termination of our tableau-based decision procedure. For instance, when applying the function dec* on $\Box\Diamond\Phi \wedge \Diamond\Phi$ we may obtain a path formula $\Box\Diamond\Phi \wedge \Diamond\Phi \wedge \Diamond\Phi$ and applying again the function dec* on the so obtained path formula may return $\Box\Diamond\Phi \wedge \Diamond\Phi \wedge \Diamond\Phi \wedge \Diamond\Phi$, and so on forever. Without caution, the closure of the initial formula might be infinite, and therefore also the corresponding tableau. Moreover when the formula is complicated with $\wedge$ and $\vee$ embedded in temporal operators, we may not be able to define which parts of the path formula are identical. We avoid these unwanted behaviours with our use of $\dot{\wedge}$ and $\dot{\vee}$ and the transformation of any new path formula in conjunctive normal form without redundancies.

Example 4.2 (function dec* applied to $\langle\langle H\rangle\rangle(\neg l \mathsf{U}(l \wedge \Diamond\Box u))$)**.** For this example, the path formula to be studied is $\Phi = \neg l \mathsf{U}(l \wedge \Diamond\Box u)$ therefore

$$\mathsf{dec}^*(\Phi) = (\{\langle \top, \neg l \mathsf{U}(l \wedge \Diamond\Box u), \{\neg l\}\rangle\} \otimes^* \mathsf{dec}^*(\neg l)) \cup (\{\langle \top, \top, \{l \wedge \Diamond\Box u\}\rangle\} \otimes^* \mathsf{dec}^*(l \wedge \Diamond\Box u))$$

$$\mathsf{dec}^*(\Phi) = (\{\langle \top, \neg l \mathsf{U}(l \wedge \Diamond\Box u), \{\neg l\}\rangle\} \otimes^* \{\langle \neg l, \top, \emptyset\rangle\}) \cup (\{\langle \top, \top, \{l \wedge \Diamond\Box u\}\rangle\} \otimes^* \mathsf{dec}^*(l) \otimes^* \mathsf{dec}^*(\Diamond\Box u))$$

$$\mathsf{dec}^*(\Phi) = \{\langle \neg l, \neg l \mathsf{U}(l \wedge \Diamond\Box u), \{\neg l\}\rangle\} \cup (\{\langle \top, \top, \{l \wedge \Diamond\Box u\}\rangle\} \otimes^* \{\langle l, \top, \emptyset\rangle\} \otimes^* (\{\langle \top, \Diamond\Box u, \{\top\}\rangle\} \otimes^* \mathsf{dec}^*(\top)) \cup (\{\langle \top, \top, \{\Box u\}\rangle\} \otimes^* \mathsf{dec}^*(\Box u)))$$

$$\mathsf{dec}^*(\Phi) = \{\langle \neg l, \neg l \mathsf{U}(l \wedge \Diamond\Box u), \{\neg l\}\rangle\} \cup (\{\langle l, \top, \{l \wedge \Diamond\Box u\}\rangle\} \otimes^* (\{\langle \top, \Diamond\Box u, \{\top\}\rangle\} \cup \{\langle u, \Box u, \{\Box u, u\}\rangle\}))$$

$$\mathsf{dec}^*(\Phi) = \{\langle \neg l, \neg l \mathsf{U}(l \wedge \Diamond\Box u), \{\neg l\}\rangle\} \cup (\{\langle l, \top, \{l \wedge \Diamond\Box u\}\rangle\} \otimes^* \{\langle \top, \Diamond\Box u, \{\top\}\rangle, \langle u, \Box u, \{\Box u, u\}\rangle\})$$

$$\mathsf{dec}^*(\Phi) = \{\langle \neg l, \neg l \mathsf{U}(l \wedge \Diamond\Box u), \{\neg l\}\rangle\} \cup \{\langle l, \Diamond\Box u, \{l \wedge \Diamond\Box u\}\rangle, \langle l \wedge u, \Box u, \{l \wedge \Diamond\Box u, \Box u, u\}\rangle\}$$

$$\mathsf{dec}^*(\Phi) = \{\langle \neg l, \neg l \mathsf{U}(l \wedge \Diamond\Box u), \{\neg l\}\rangle, \langle l, \Diamond\Box u, \{l \wedge \Diamond\Box u\}\rangle, \langle l \wedge u, \Box u, \{l \wedge \Diamond\Box u, \Box u, u\}\rangle\}$$

4.1.3 Decomposition of ATL^+ and ATL^* Formulae

Let $\zeta = \langle\langle A\rangle\rangle\Phi$ or $\zeta = [\![A]\!]\Phi$ be a γ-formula to be decomposed. All pairs $\langle\psi,\Psi\rangle \in \mathsf{dec}^+(\Phi)$ are converted to a *γ-component* $\gamma(\psi,\Psi)$ of the form:

$$\gamma(\psi,\Psi) = \psi \text{ if } \Psi = \top$$
$$\gamma(\psi,\Psi) = \psi \wedge \langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\Psi \text{ if } \zeta = \langle\langle A\rangle\rangle\Phi$$
$$\gamma(\psi,\Psi) = \psi \wedge [\![A]\!]\bigcirc[\![A]\!]\Psi \text{ if } \zeta = [\![A]\!]\Phi$$

in the case of ATL^+;

or all triples $\langle\psi,\Psi,S\rangle \in \mathsf{dec}^*(\Phi)$ are converted to a γ-set $\gamma_s(\psi,\Psi,S) = S$ and a *γ-component* $\gamma_c(\psi,\Psi,S)$ as follows:

$$\gamma_c(\psi,\Psi,S) = \psi \text{ if } \Psi = \top$$
$$\gamma_c(\psi,\Psi,S) = \psi \wedge \langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\Psi \text{ if } \zeta = \langle\langle A\rangle\rangle\Phi,$$
$$\gamma_c(\psi,\Psi,S) = \psi \wedge [\![A]\!]\bigcirc[\![A]\!]\Psi \text{ if } \zeta = [\![A]\!]\Phi$$

in the case of ATL^*.

Note that the sets $\gamma_s(\psi,\Psi,S)$ will be used during the elimination phase in order to determine whether eventualities are realized or not (see Section 4.4)

Example 4.3 (Continuation of Example 4.1 and 4.2)**.** The γ-components of $[\![H,B]\!](\Box\neg p \vee \Box\neg r)$ are

$$\gamma(\neg p, \Box\neg p) = \neg p \wedge [\![H,B]\!]\bigcirc[\![H,B]\!]\Box\neg p$$
$$\gamma(\neg r, \Box\neg r) = \neg r \wedge [\![H,B]\!]\bigcirc[\![H,B]\!]\Box\neg r$$
$$\gamma(\neg p \wedge \neg r, \Box\neg p \vee \Box\neg r) = \neg p \wedge \neg r \wedge [\![H,B]\!]\bigcirc[\![H,B]\!](\Box\neg p \vee \Box\neg r)$$

The γ-components and γ-sets of $[\![H]\!](\neg l \mathsf{U}(l \wedge \Diamond\Box u)$ are

$$\gamma_c(\neg l, \neg l\mathsf{U}(l \wedge \Diamond\Box u), \{\neg l\}) = \neg l\langle\langle H\rangle\rangle\bigcirc\langle\langle H\rangle\rangle(\neg l\mathsf{U}(l \wedge \Diamond\Box u)) \text{ and}$$
$$\gamma_s(\neg l, \neg l\mathsf{U}(l \wedge \Diamond\Box u), \{\neg l\}) = \{\neg l\}$$
$$\gamma_c(l, \Diamond\Box u, \{l \wedge \Diamond\Box u\}) = l \wedge \langle\langle H\rangle\rangle\bigcirc\langle\langle H\rangle\rangle\Diamond\Box u \text{ and}$$
$$\gamma_s(l, \Diamond\Box u, \{l \wedge \Diamond\Box u\}) = \{l \wedge \Diamond\Box u\}$$
$$\gamma_c(l \wedge u, \Box u, \{l \wedge \Diamond\Box u, \Box u, u\}) = l \wedge u \wedge \langle\langle H\rangle\rangle\bigcirc\langle\langle H\rangle\rangle\Box u \text{ and}$$
$$\gamma_s(l \wedge u, \Box u, \{l \wedge \Diamond\Box u, \Box u, u\}) = \{l \wedge \Diamond\Box u, \Box u, u\}$$

Closure The closure $cl(\varphi)$ of an ATL^+ or ATL^* state formula φ is the least set of ATL^* formulae such that $\varphi, \top, \bot \in cl(\varphi)$, and $cl(\varphi)$ is closed under taking successor, α-, β- and γ-components of φ. For any set of state formulae Γ we define

$$\mathsf{cl}(\Gamma) = \bigcup\{\mathsf{cl}(\psi) \mid \psi \in \Gamma\} \tag{4.5}$$

For both ATL$^+$ and ATL*, each formula in the initial tableau for the formula θ belongs to $\mathsf{cl}(\theta)$ and $|\mathsf{cl}(\theta)|$ is finite (see Section 4.5). This ensures the termination of the construction procedure.

We prove with lemma 4.1 that each disjunction of $\gamma_c(\psi,\Psi,S)$ obtained from the function $\mathsf{dec}^*(\Phi)$ is equivalent to the analysed formula $\langle\langle A\rangle\rangle\Phi$ (or $[\![A]\!]\Phi$). This key property is item 3 of the lemma (the first two items being just auxiliary claims), and it is the core distinction between the proposed calculus for ATL$^+$ in this thesis and the tableau calculus for ATL.

We only give the following lemma with its proof for ATL*, the case of ATL$^+$ being very similar.

Lemma 4.1. *For any* ATL* *γ-formula $\zeta = \langle\langle A\rangle\rangle\Phi$ or $\zeta = [\![A]\!]\Phi$, the following properties hold:*

1. $\Phi \equiv \bigvee\{\psi \wedge \bigcirc\Psi \mid \langle\psi,\Psi,S\rangle \in \mathsf{dec}^*(\Phi)\}$
2. $\langle\langle A\rangle\rangle\Phi \equiv \bigvee\{\langle\langle A\rangle\rangle(\psi \wedge \bigcirc\Psi) \mid \langle\psi,\Psi,S\rangle \in \mathsf{dec}^*(\Phi)\}$*, and*
   $[\![A]\!]\Phi \equiv \bigvee\{[\![A]\!](\psi \wedge \bigcirc\Psi) \mid \langle\psi,\Psi,S\rangle \in \mathsf{dec}^*(\Phi)\}$
3. $\langle\langle A\rangle\rangle\Phi \equiv \bigvee\{\gamma_c(\psi,\Psi,S) \mid \langle\psi,\Psi,S\rangle \in \mathsf{dec}^*(\Phi)\}$

Proof.

Claim 1) We will prove the claim by induction on the path formula Φ. It is equivalent to the following property $P(\Phi)$:

> For every CGM $\mathcal{M}$ and a play λ in it, $\mathcal{M},\lambda \models \Phi$ iff there exists $\langle\psi,\Psi,S\rangle \in \mathsf{dec}^*(\Phi)$ such that $\mathcal{M},\lambda_0 \models \psi$ and $\mathcal{M},\lambda_{\geq 1} \models \Psi$.

The base cases is $\Phi = \varphi$: the property $P(\Phi)$ follows immediately from the definition of dec^*.

For the inductive steps, there are five cases to consider:

Case 1 **[$\Phi = \Phi_1 \wedge \Phi_2$]** We have that:
$\mathcal{M},\lambda \models \Phi$ iff $\mathcal{M},\lambda \models \Phi_1$ and $\mathcal{M},\lambda \models \Phi_2$ iff by the inductive hypothesis on Φ_1 and Φ_2
(i) there exists $\langle\psi_1,\Psi_1,S_1\rangle \in \mathsf{dec}^*(\Phi_1)$ such that $\mathcal{M},\lambda_0 \models \psi_1$ and $\mathcal{M},\lambda_{\geq 1} \models \Psi_1$
(ii) there exists $\langle\psi_2,\Psi_2,S_2\rangle \in \mathsf{dec}^*(\Phi_2)$ such that $\mathcal{M},\lambda_0 \models \psi_2$ and $\mathcal{M},\lambda_{\geq 1} \models \Psi_2$
These two are the case iff
$\mathcal{M},\lambda_0 \models \psi_1 \wedge \psi_2$ and $\mathcal{M},\lambda_{\geq 1} \models \Psi_1 \wedge \Psi_2$
iff $\mathcal{M},\lambda_0 \models \psi$ and $\mathcal{M},\lambda_{\geq 1} \models \Psi$ where $\psi = \psi_1 \wedge \psi_2$, $\Psi = \Psi_1 \wedge \Psi_2$ and $\langle\psi,\Psi,S\rangle \in \mathsf{dec}^*(\Phi)$ with $S = S_1 \cup S_2$.

This completes the proof of $P(\Phi)$ for $\Phi = \Phi_1 \wedge \Phi_2$.

Case 2 **[$\Phi = \Phi_1 \vee \Phi_2$]** We have that $\mathcal{M},\lambda \models \Phi$ iff $\mathcal{M},\lambda \models \Phi_1$ or $\mathcal{M},\lambda \models \Phi_2$.
By inductive hypothesis for Φ_1 and Φ_2 and from the fact that $\mathsf{dec}^*(\Phi_1) \cup \mathsf{dec}^*(\Phi_2) \subseteq \mathsf{dec}^*(\Phi)$, we obtain the direction from left to right in property $P(\Phi)$.

For the converse direction, we only need to consider the case that does not follow directly from the inductive hypothesis for Φ_1 and Φ_2, viz. when there exists $\langle\psi,\Psi,S\rangle \in (\mathsf{dec}^*(\Phi_1) \oplus^* \mathsf{dec}^*(\Phi_2))$ such that $\mathcal{M},\lambda_0 \models \psi$ and $\mathcal{M},\lambda_{\geq 1} \models \Psi$. In this case, $\psi = \psi_1 \wedge \psi_2$ and $\Psi = \Psi_1 \vee \Psi_2$ for some $\langle\psi_1,\Psi_1,S_1\rangle \in \mathsf{dec}^*(\Phi_1)$ and $\langle\psi_2,\Psi_2,S_2\rangle \in \mathsf{dec}^*(\Phi_2)$ such that $\Psi_1 \neq \top$ and $\Psi_2 \neq \top$.

Suppose $\mathcal{M},\lambda_{\geq 1} \models \Psi_1$. Since we also have $\mathcal{M},\lambda_0 \models \psi_1$, by inductive hypothesis for Φ_1, it follows that $\mathcal{M},\lambda \models \Phi_1$, hence $\mathcal{M},\lambda \models \Phi$.
Likewise, when $\mathcal{M},\lambda_{\geq 1} \models \Psi_2$.

Case 3 **[$\Phi = \bigcirc\Phi'$]** The property $P(\Phi)$ follows immediately from the definition of dec^*.

Case 4 **[$\Phi = \Box\Phi_1$]**

- *Left to right*
  If $\mathcal{M},\lambda \models \Box\Phi_1$ then using the well-known fixed-point LTL equivalences, we have that $\mathcal{M},\lambda \models \Phi_1$ and $\mathcal{M},\lambda_{\geq 1} \models \Box\Phi_1$. By the inductive hypothesis for Φ_1, we have that $\mathcal{M},\lambda \models \Phi_1$ only if there is $\langle \psi_1,\Psi_1,S_1\rangle \in \mathsf{dec}^*(\Phi_1)$ such that $\mathcal{M},\lambda_0 \models \psi_1$, $\mathcal{M},\lambda_{\geq 1} \models \Psi_1$.
  Therefore $\mathcal{M},\lambda_0 \models \psi_1$, $\mathcal{M},\lambda_{\geq 1} \models \Psi_1 \wedge \Box\Phi_1$, that is $\mathcal{M},\lambda_0 \models \psi$, $\mathcal{M},\lambda_{\geq 1} \models \Psi$ where $\psi = \psi_1 = \psi_1 \wedge \top$, $\Psi = \Psi_1 \wedge \Box\Phi_1$. As Φ_1 is initiated at λ_0, we can set that $S = S_1 \cup \{\Phi_1\}$. The so obtained triple $\langle \psi,\Psi,S\rangle$ belongs to $\mathsf{dec}^*(\Phi)$ by definition of $\mathsf{dec}(\Box\Phi)$ and $\otimes^*$.
- *Right to left*
  If there exists $\langle \psi,\Psi,S\rangle \in \mathsf{dec}^*(\Box\Phi_1)$ such that $\mathcal{M},\lambda_0 \models \psi$ and $\mathcal{M},\lambda_{\geq 1} \models \Psi$, then by construction of dec^*, there exists $\langle \psi_1,\Psi_1,S_1\rangle \in \mathsf{dec}^*(\Phi_1)$ such that $\langle \psi,\Psi,S\rangle = \langle \psi_1,\Psi_1 \wedge \Box\Phi_1,S_1 \cup \{\Phi_1\}\rangle$. So $\mathcal{M},\lambda_0 \models \psi_1$, $\mathcal{M},\lambda_{\geq 1} \models \Psi_1 \wedge \Box\Phi_1$, that is $\mathcal{M},\lambda_{\geq 1} \models \Psi_1$ and $\mathcal{M},\lambda_{\geq 1} \models \Box\Phi_1$. So, by inductive hypothesis, $\mathcal{M},\lambda \models \Phi_1$. Thus, $\mathcal{M},\lambda \models \Phi_1$ and $\mathcal{M},\lambda_{\geq 1} \models \Box\Phi_1$, that is $\mathcal{M},\lambda \models \Box\Phi_1$.

Case 5 **[$\Phi = \Phi_1 \mathsf{U} \Phi_2$]**

- *Left to right*
  If $\mathcal{M},\lambda \models \Phi$ then
  (i) $\mathcal{M},\lambda \models \Phi_1$ and $\mathcal{M},\lambda_{\geq 1} \models \Phi_1 \mathsf{U} \Phi_2$, or
  (ii) $\mathcal{M},\lambda \models \Phi_2$
  For the case **(i)**, by the inductive hypothesis for Φ_1, we have that
  there exists $\langle \psi_1,\Psi_1,S_1\rangle \in \mathsf{dec}^*(\Phi_1)$ such that $\mathcal{M},\lambda_0 \models \psi_1$ and $\mathcal{M},\lambda_{\geq 1} \models \Psi_1$. Therefore $\mathcal{M},\lambda_0 \models \psi_1$ and $\mathcal{M},\lambda_{\geq 1} \models \Psi_1 \wedge \Phi_1 \mathsf{U} \Phi_2$, that is $\mathcal{M},\lambda_0 \models \psi$ and $\mathcal{M},\lambda_{\geq 1} \models \Psi$ where $\psi = \psi_1 = \psi_1 \wedge \top$, $\Psi = \Psi_1 \wedge \Phi_1 \mathsf{U} \Phi_2$. As Φ_1 is initiated at λ_0, we can set that $S = S_1 \cup \{\Phi_1\}$. The so obtained triple $\langle \psi,\Psi,S\rangle$ belongs to $\mathsf{dec}^*(\Phi)$ by definition of $\mathsf{dec}^*(\Box\Phi)$ and $\otimes^*$.
  For the case **(ii)**, by the inductive hypothesis for Φ_2, we have that
  there exists $\langle \psi_2,\Psi_2,S_2\rangle \in \mathsf{dec}^*(\Phi_2)$ such that $\mathcal{M},\lambda_0 \models \psi_2$ and $\mathcal{M},\lambda_{\geq 1} \models \Psi_2$. Therefore $\mathcal{M},\lambda_0 \models \psi_2$ and $\mathcal{M},\lambda_{\geq 1} \models \Psi_2$, that is $\mathcal{M},\lambda_0 \models \psi$ and $\mathcal{M},\lambda_{\geq 1} \models \Psi$ where $\psi = \psi_2 = \psi_2 \wedge \top$, $\Psi = \Psi_2$. As Φ_2 is initiated at λ_0, we can set that $S = S_2 \cup \{\Phi_2\}$. The so obtained triple $\langle \psi,\Psi,S\rangle$ belongs to $\mathsf{dec}^*(\Phi)$ by definition of $\mathsf{dec}^*(\Box\Phi)$ and $\otimes^*$.
- *Right to left*
  If there exists $\langle \psi,\Psi,S\rangle \in \mathsf{dec}^*(\Phi)$ such that $\mathcal{M},\lambda_0 \models \psi$ and $\mathcal{M},\lambda_{\geq 1} \models \Psi$ then by definition of the function dec^*, $\langle \psi,\Psi,S\rangle \in \mathsf{dec}^*(\Phi)$ is either
  (i) $\langle \psi_1,\Psi_1 \wedge \Phi_1 \mathsf{U} \Phi_2,S \cup \{\Phi_1\}\rangle$ where $\langle \psi_1,\Psi_1,S_1\rangle \in \mathsf{dec}^*(\Phi_1)$, or
  (ii) $\langle \psi_2,\Psi_2,S_2 \cup \{\Phi_2\}\rangle$ where $\langle \psi_2,\Psi_2,S_2\rangle \in \mathsf{dec}^*(\Phi_2)$.
  The first case **(i)** means that $\mathcal{M},\lambda_0 \models \psi_1$ and $\mathcal{M},\lambda_{\geq 1} \models \Psi_1 \wedge \Phi_1 \mathsf{U} \Phi_2$. Therefore $\mathcal{M},\lambda_0 \models \psi_1$

and $\mathscr{M}, \lambda_{\geq 1} \models \Psi_1$. So by the inductive hypothesis on Φ_1, we have $\mathscr{M}, \lambda \models \Phi_1$. Thus, since we also have $\mathscr{M}, \lambda_{\geq 1} \models \Phi_1 \mathsf{U} \Phi_2$, $\mathscr{M}, \lambda \models \Phi_1 \mathsf{U} \Phi_2$.
The second case **(ii)** means that $\mathscr{M}, \lambda_0 \models \psi_2$ and $\mathscr{M}, \lambda_{\geq 1} \models \Psi_2$. By the inductive hypothesis on Φ_2, $\mathscr{M}, \lambda \models \Phi_2$. Thus $\mathscr{M}, \lambda \models \Phi_1 \mathsf{U} \Phi_2$.

Claim 2) We will consider the case of $\Theta = \langle\langle A \rangle\rangle \Phi$; the case of $[\![A]\!]\Phi$ is analogous.

- The implication from right to left of the claimed equivalence follows from Claim 1.a) and the monotonicity of $\langle\langle A \rangle\rangle$ (in the sense that if $\Psi \models \Phi$ then $\langle\langle A \rangle\rangle \Psi \models \langle\langle A \rangle\rangle \Phi$).
- To prove the converse direction, we take any CGM $\mathscr{M}$ and state s in it such that $\mathscr{M}, s \models \langle\langle A \rangle\rangle \Phi$. We also take and fix any collective strategy F_A of A such that $\mathscr{M}, \lambda \models \Phi$ for every play λ starting at s and consistent with F_A. We denote that set of such plays by $\mathsf{Out}(s, F_A)$. What we need to prove is equivalent to the following property $P(\Phi)$ by induction on Φ:

  > For all plays $\lambda \in \mathsf{Out}(s, F_A)$ satisfying Φ, there exists one $\langle \psi, \Psi, S \rangle \in \mathsf{dec}^*(\Phi)$ such that $\mathscr{M}, \lambda \models \psi \wedge \bigcirc \Psi$.

  We then recall that every ATL* path formula Ξ is a positive Boolean combination of sub-formulae of the types φ, $\bigcirc \Phi'_1$, $\Box \Phi'_1$ and $\Phi'_1 \mathsf{U} \Phi'_2$, where φ is a state formula and Φ'_1, Φ'_2 are path formulae.
  Let the set of these sub-formulae be $S(\Xi)$. Now, we introduce some ad hoc notation for special sets of formulae in $S(\Xi)$.
  - $L(\Xi)$ is the set of all state formulae in $S(\Xi)$;
  - $N(\Xi) := \{\Phi'_1 \mid \bigcirc \Phi'_1 \in S(\Xi)\}$;
  - $B(\Xi) := \{\Phi'_1 \mid \Box \Phi'_1 \in S(\Xi)\}$;
  - $U(\Xi) := \{\Phi'_1 \mathsf{U} \Phi'_2 \mid \Phi'_1 \mathsf{U} \Phi'_2 \in S(\Xi)\}$;
  - $U_1(\Xi) := \{\Phi'_1 \mid \Phi'_1 \mathsf{U} \Phi'_2 \in S(\Xi)\}$;
  - $U_2(\Xi) := \{\Phi'_2 \mid \Phi'_1 \mathsf{U} \Phi'_2 \in S(\Xi)\}$.

  Without loss of generality we can assume that Φ is in a D.N.F over the set of formulae in $S(\Phi)$, i.e. $\Phi = \Phi_1 \vee \cdots \vee \Phi_m$, where each Φ_i is a conjunction of formulae from $S(\Phi)$.
  For every play $\lambda \in \mathsf{Out}(s, F_A)$, there is a set $\{\Phi_1, \ldots, \Phi_n\}$ such that $\mathscr{M}, \lambda \models \{\Phi_1, \ldots, \Phi_n\}$ for some $n \leqslant m$.
  Let Φ_i be one of them. We will associate with it a pair $\langle \psi_i, \Psi_i, S_i \rangle \in \mathsf{dec}^*(\Phi)$ as follows:
  First note that all formulae from $L(\Phi_i)$ are true at s and all formulae of $B(\Phi_i)$ are true on λ. Further, let $E_i(s, F_A)$ be the subset of those formulae from $U_2(\Phi_i)$ which are true on all paths $\lambda \in \mathsf{Out}(s, F_A)$ satisfying Φ_i.
  Thus, for every play $\lambda \in \mathsf{Out}(s, F_A)$ satisfying Φ_i the following holds:
  (i) $\mathscr{M}, \lambda \models \varphi$ for every $\varphi \in L(\Phi_i)$
  (ii) $\mathscr{M}, \lambda \models \bigcirc \Phi'_1$ for each $\bigcirc \Phi'_1 \in S(\Phi_i)$
  (iii) $\mathscr{M}, \lambda \models \psi' \wedge \bigcirc (\Psi' \wedge \Box \Phi'_1)$ for each $\Box \Phi'_1 \in S(\Phi_i)$ and some appropriate $\langle \psi', \Psi', S' \rangle \in \mathsf{dec}^*(\Phi'_1)$

(iv) $\mathscr{M},\lambda \models \psi' \wedge \bigcirc\Psi'$ for each $\Phi'_2 \in E_i(s,F_A)$ and some appropriate $\langle \psi',\Psi',S'\rangle \in \mathsf{dec}^*(\Phi'_2)$

(v) $\mathscr{M},\lambda \models \Phi'_1 \wedge \bigcirc(\Psi' \wedge \Phi'_1 \mathsf{U} \Phi'_2)$ for each $\Phi'_2 \in U_2(\Phi_i) - E_i(s,F_A)$ and some appropriate $\langle \psi',\Psi',S'\rangle \in \mathsf{dec}^*(\Phi'_1)$

Properties **(iii)-(v)** come respectively from

a) $\mathscr{M},\lambda \models \Phi'_1 \wedge \bigcirc\Box\Phi'_1$ for each $\Box\Phi'_1 \in S(\Phi_i)$, so $\mathscr{M},\lambda \models \Phi'_1$ and $\mathscr{M},\lambda \models \bigcirc\Box\Phi'_1$. By applying the inductive hypothesis on Φ'_1 we have that $\mathscr{M},\lambda \models \psi' \wedge \bigcirc\Psi'$ for some $\langle \psi',\Psi',S'\rangle \in \mathsf{dec}^*(\Phi'_1)$, thus $\mathscr{M},\lambda \models \psi' \wedge \bigcirc(\Psi' \wedge \Box\Phi'_1)$

b) $\mathscr{M},\lambda \models \Phi'_2$ for each $\Phi'_2 \in E_i(s,F_A)$. By applying the inductive hypothesis on Φ'_2 we have that $\mathscr{M},\lambda \models \psi' \wedge \bigcirc\Psi'$ for some $\langle \psi',\Psi',S'\rangle \in \mathsf{dec}^*(\Phi'_2)$.

c) $\mathscr{M},\lambda \models \Phi'_1 \wedge \bigcirc\Phi'_1 \mathsf{U} \Phi'_2$ for each $\Phi'_2 \in U_2(\Phi_i) - E_i(s,F_A)$, so $\mathscr{M},\lambda \models \Phi'_1$ and $\mathscr{M},\lambda \models \bigcirc\Phi'_1 \mathsf{U} \Phi'_2$. By applying the inductive hypothesis on Φ'_1 we have that $\mathscr{M},\lambda \models \psi' \wedge \bigcirc\Psi'$ for some $\langle \psi',\Psi',S'\rangle \in \mathsf{dec}^*(\Phi'_1)$, thus $\mathscr{M},\lambda \models \Phi'_1 \wedge \bigcirc(\Psi' \wedge \Phi'_1 \mathsf{U} \Phi'_2)$

Now, suppose $\Phi_i = \Psi_{i1} \wedge \cdots \wedge \Psi_{ik}$ for some $\Psi_{i1} \wedge \cdots \wedge \Psi_{ij} \in S(\Phi)$. Then $\mathsf{dec}^*(\Phi) = \mathsf{dec}^*(\Psi_{i1}) \otimes^* \cdots \otimes^* \mathsf{dec}^*(\Psi_{ik})$. (Recall that the operators $\otimes^*$ and $\oplus^*$ are associative, up to logical equivalence, so there is no need to put parentheses.) As seen with properties (i)-(v), every sub-formula Ψ_{ij} of the form $\Box\Phi'_1$ or $\Phi'_1 \mathsf{U} \Phi'_2$ is associated with a triple $\langle \psi',\Psi',S'\rangle \in \mathsf{dec}^*(\Phi'_1)$ or $\in \mathsf{dec}^*(\Phi'_2)$. Let $G(\Xi)$ be the union of all S' associated to each $\Psi_{ij} \in \Psi_{i1} \wedge \cdots \wedge \Psi_{ik} = \Phi_i$.

Thus, for every $\langle \psi_i,\Psi_i,S_i\rangle \in \mathsf{dec}^*(\Phi_i)$, S_i is the union of all formulae from $L(\Xi)$, $B(\Xi)$, $G(\Xi)$ and, for every conjunct of Φ_i of the type $\Phi'_1 \mathsf{U} \Phi'_2$, at least one of the respective formulae coming for $U_1(\Phi_i)$ and $U_2(\Phi_2)$.

Now, we select $\langle \psi_i,\Psi_i,S_i\rangle \in \mathsf{dec}^*(\Phi_i)$ to be the one where the conjuncts taken from $U_2(\Phi_i)$ are exactly those in $E_i(s)$.

Then, we claim that for every play $\lambda \in \mathsf{Out}(s,F_A)$ satisfying Φ_i, it is the case that $\mathscr{M},\lambda \models \psi_i \wedge \bigcirc\Psi_i$. Indeed, this follows from the list of properties **(i)-(v)** above and from the definition of $\mathsf{dec}^*(\Psi_{i1}) \otimes^* \cdots \otimes^* \mathsf{dec}^*(\Psi_{ik})$. Note further, that if Ψ_i above is $\top$, then $\mathscr{M},\lambda \models \psi_i \wedge \bigcirc\Psi_i$ for all paths λ starting at s and following the strategy F_A, so we can assume without affecting what follows that no Ψ_i above is $\top$.

After having selected such a pair $\langle \psi_i,\Psi_i,S_i\rangle \in \mathsf{dec}^*(\Phi_i)$ for each $\Phi_i \in \{\Phi_1,\ldots,\Phi_n\}$, we use these n pairs (or, those of them for which $\Psi_i \neq \top$) to construct the pair $\langle \psi,\Psi,S\rangle \in \mathsf{dec}^*(\Phi_1) \oplus^* \cdots \oplus^* \mathsf{dec}^*(\Phi_n)$ such that $\psi = \psi_1 \wedge \cdots \wedge \psi_n$, $\Psi = \Psi_1 \vee \cdots \vee \Psi_n$ and $S = S_1 \cup \cdots \cup S_n$.

Finally we claim by virtue of the construction, $\mathscr{M},\lambda \models \psi \wedge \bigcirc\Psi$ for every play $\lambda \in \mathsf{Out}(s,F_A)$ satisfying Φ. Therefore, the strategy F_A is a witness of the truth of $\mathscr{M},s \models \langle\langle A\rangle\rangle(\psi \wedge \bigcirc\Psi)$, hence $\mathscr{M},s \models \bigvee\{\langle\langle A\rangle\rangle(\psi \wedge \bigcirc\Psi) \mid \langle \psi,\Psi,S\rangle \in \mathsf{dec}^*(\Phi)\}$.

This completes the proof of the implication from left to right of Claim 2.

Claim 3 This claim follows easily from Claim 2 respectively by noting that:

- $\langle\langle A\rangle\rangle(\psi \wedge \bigcirc\Psi) \equiv \psi \wedge \langle\langle A\rangle\rangle \bigcirc \Psi \equiv \psi \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle \Psi$, because ψ is a state formula. Note that the second equivalence is due to the fact that the semantics of $\langle\langle\rangle\rangle$ is based on perfect-recall strategies, that can be composed. More precisely, it essentially assumes that any strategy

at s ensuring that every successor satisfies $\langle\langle A\rangle\rangle\Psi$ can be decomposed with the family of strategies, one for every such successor s' witnessing the truth of $\langle\langle A\rangle\rangle\Psi$ on all plays starting at s', into a perfect-recall strategy that guarantees the truth of $\bigcirc\Psi$ on all plays starting at s. (This, in general, cannot be done if only positional strategies are considered, as those applied at the different successors of s may interfere with each other).

- Likewise, $[\![A]\!](\psi \wedge \bigcirc\Psi) \equiv \psi \wedge [\![A]\!]\bigcirc\Psi \equiv \psi \wedge [\![A]\!]\bigcirc[\![A]\!]\Psi$.

■

4.2 Saturation of Prestates

Decomposition of γ-formulae for ATL$^+$ and ATL* gives γ-components that we need to take into account during the saturation of prestates. Therefore, we give a new definition of *full saturated sets of formulae*.

Definition 4.1 (full saturated sets of ATL$^+$ (resp. ATL*) formulae)**.** Let Γ, Δ be sets of ATL$^+$ (resp. ATL*) formulae and $\Gamma \subseteq \Delta \subseteq \mathsf{cl}(\Gamma)$.

1. Δ is *patently inconsistent* if it contains $\bot$ or a pair of formulae φ and $\neg\varphi$.
2. Δ is a *full saturated set of* Γ if it is not patently inconsistent and satisfies the following closure conditions:
   - if $\varphi \wedge \psi \in \Delta$ then $\varphi \in \Delta$ and $\psi \in \Delta$;
   - if $\varphi \vee \psi \in \Delta$ then $\varphi \in \Delta$ or $\psi \in \Delta$;
   - if $\varphi \in \Delta$ is a γ-formula, then at least one γ-component of φ is in Δ and exactly one of these γ-components, say $\gamma(\psi,\Psi)$ (resp. $\gamma_c(\psi,\Psi,S)$), in Δ, denoted $\gamma_l(\varphi,\Delta)$, is designated as *the γ-component in Δ linked to the γ-formula φ*, as explained below. In the case of ATL*, we also denote by $\gamma_{sl}(\varphi,\Delta)$ the set of path formulae $\gamma_s(\psi,\Psi,S)$, which is linked to the γ-component $\gamma_l(\varphi,\Delta)$.

For ATL* (resp. ATL$^+$), the family of all full saturated sets of a set Γ is denoted $\mathbf{FS}^*(\Gamma)$ (resp. $\mathbf{FS}^+(\Gamma)$).

Proposition 4.1. *For any finite set of* ATL$^+$ *or* ATL* *state formulae* Γ*:*

$$\bigwedge\Gamma \equiv \bigvee\{\bigwedge\Delta \mid \Delta \in \mathbf{FS}^*(\Gamma)\}.$$

Proof. Lemma 4.1 implies that every step of extension of a set of ATL* state formulae applied to a family of sets $\mathscr{F}$ preserves the formula $\bigvee\{\bigwedge\Delta \mid \Delta \in \mathscr{F}\}$ up to logical equivalence. At the beginning, that formula is $\bigwedge\Gamma$. ■

Example 4.4. The saturation of the prestate $\Gamma_0 = \{\theta_1^+\}$ gives three successor states

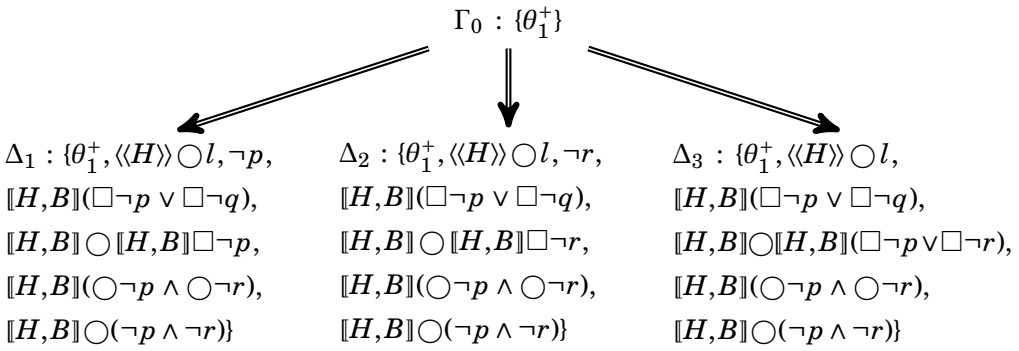


Figure 4.2: Successor states of $\Gamma_0 = \{\theta_1^+\}$

Moreover:

$$\gamma_l([\![H,B]\!](\Box\neg p \vee \Box\neg r), \Delta_1) = \gamma(\neg p, \Box\neg p) = \neg p \wedge [\![H,B]\!]\bigcirc[\![H,B]\!]\Box\neg p$$
$$\gamma_l([\![H,B]\!](\Box\neg p \vee \Box\neg r), \Delta_2) = \gamma(\neg r, \Box\neg r) = \neg r \wedge [\![H,B]\!]\bigcirc[\![H,B]\!]\Box\neg r$$
$$\gamma_l([\![H,B]\!](\Box\neg p \vee \Box\neg r), \Delta_3) = \gamma(\neg p \wedge \neg r, \Box\neg p \vee \Box\neg r) = \neg p \wedge \neg r \wedge [\![H,B]\!]\bigcirc[\![H,B]\!](\Box\neg p \vee \Box\neg r)$$

Example 4.5 (Saturation of the prestate $\Gamma_0 = \{\theta_3^*\}$)**.** The saturation of the prestate $\Gamma_0 = \{\theta_3^*\}$ gives two successor states

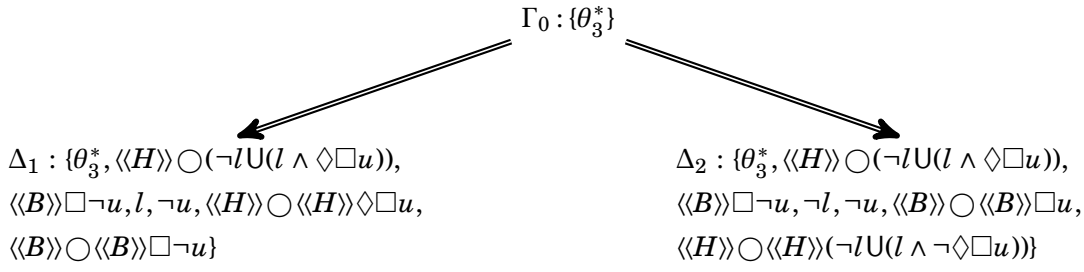


Figure 4.3: Successor states of $\Gamma_0 = \{\theta_3^*\}$

Moreover:

$$\gamma_l(\langle\langle H\rangle\rangle(\neg l \mathsf{U}(l \wedge \Diamond\Box u)), \Delta_1) = \gamma_c(l, \Diamond\Box u, \{l \wedge \Diamond\Box u\}) = l \wedge \langle\langle 1\rangle\rangle \bigcirc \langle\langle 1\rangle\rangle\Diamond\Box u, \text{ and}$$
$$\gamma_{sl}(\langle\langle H\rangle\rangle(\neg l \mathsf{U}(l \wedge \Diamond\Box u)), \Delta_1) = \gamma_s(l, \Diamond\Box u, \{l \wedge \Diamond\Box u\}) = \{l \wedge \Diamond\Box u\}$$
$$\gamma_l(\langle\langle H\rangle\rangle(\neg l \mathsf{U}(l \wedge \Diamond\Box u)), \Delta_2) = \gamma_c(\neg l, \neg l \mathsf{U}(l \wedge \Diamond\Box u), \{\neg l\}) = \neg l \wedge \langle\langle 1\rangle\rangle \bigcirc \langle\langle 1\rangle\rangle(\neg l \mathsf{U}(l \wedge \Diamond\Box u)), \text{ and}$$
$$\gamma_{sl}(\langle\langle H\rangle\rangle(\neg l \mathsf{U}(l \wedge \Diamond\Box u)), \Delta_2) = \gamma_s(\neg l, \neg l \mathsf{U}(l \wedge \Diamond\Box u), \{\neg l\}) = \{\neg l\}$$

4.3 Rule Next

In order to take into account the negation normal form of the ATL$^+$ and ATL* syntax, we slightly modify the rule **Next**. Besides taking into account negation normal form, this new rule **Next** no longer considers formulae of the form $[\![\mathbb{A}]\!]\bigcirc\varphi$ as actions available to agents. Indeed, for these particular successor formulae, φ has to be present in all successor prestates and therefore the "program" or "vote" of the agents does not really enter in consideration. However, a special case is to be considered when *all* successor formulae in $\mathbb{L}$ are of the form $[\![\mathbb{A}]\!]\bigcirc\varphi$. Indeed, in this case, we apply the rule **Next** with one dummy action, ensuring that at least one action is available to agents, this is why we have $r_\Delta = \max\{m+l, 1\}$, below.

Given a state Δ, do the following, where σ is a shorthand for $\sigma_{\mathbb{A}}$:

1. List all primitive successor formulae of Δ in such a way that all successor formulae of the form $\langle\langle A\rangle\rangle\bigcirc\varphi$ precede all formulae of the form $[\![A']\!]\bigcirc\varphi$, where $A' \neq \mathbb{A}$ and put at the end of the list, all formulae of the form $[\![\mathbb{A}]\!]\bigcirc\varphi$; let the result be the list

$$\begin{aligned}\mathbb{L} = \big[&\langle\langle A_0\rangle\rangle\bigcirc\varphi_0, \ldots, \langle\langle A_{m-1}\rangle\rangle\bigcirc\varphi_{m-1},\ [\![A'_0]\!]\bigcirc\psi_0, \ldots, [\![A'_{l-1}]\!]\bigcirc\psi_{l-1},\\ &[\![\mathbb{A}]\!]\bigcirc\mu_0, \ldots, [\![\mathbb{A}]\!]\bigcirc\mu_{n-1}\big]\end{aligned} \tag{4.6}$$

   Let $r_\Delta = \max\{m+l, 1\}$.
   We denote by $d(\Delta)$ the set $\{0, \ldots, r_\Delta - 1\}$ and by $D(\Delta)$ the set $\{0, \ldots, r_\Delta - 1\}^{|\mathbb{A}|}$.
   Then, for every $\sigma \in D(\Delta)$, denote $N(\sigma) := \{i \mid \sigma_i \geqslant m\}$, where σ_i is the ith component of the tuple σ, and let $\mathsf{co}(\sigma) := \big[\sum_{i\in N(\sigma)}(\sigma_i - m)\big] \mod l$.
2. For each $\sigma \in D(\Delta)$ create a prestate:

$$\begin{aligned}\Gamma_\sigma = &\{\varphi_p \mid \langle\langle A_p\rangle\rangle\bigcirc\varphi_p \in \Delta \text{ and } \sigma_{\mathsf{a}} = p \text{ for all } \mathsf{a} \in A_p\}\\ &\cup \{\psi_q \mid [\![A'_q]\!]\bigcirc\psi_q \in \Delta, \mathsf{co}(\sigma) = q \text{ and } \mathbb{A} - A'_q \subseteq N(\sigma)\}\\ &\cup \{\mu_r \mid [\![\mathbb{A}]\!]\bigcirc\mu_r \in \Delta\}\end{aligned} \tag{4.7}$$

   If Γ_σ is empty, add $\top$ to it. Then connect Δ to Γ_σ with $\xrightarrow{\sigma}$.
   If, however, $\Gamma_\sigma = \Gamma$ for some prestate Γ that has already been added to the initial tableau, only connect Δ to Γ with $\xrightarrow{\sigma}$.

Example 4.6 (Continuation of Example 4.4)**.** For the state Δ_1 of the tableau for θ_1^+, the list of successor formulae is the following:

$$\mathbb{L} = [\langle\langle H\rangle\rangle^{0}_{0}\bigcirc l,\ [\![H]\!]\bigcirc[\![H]\!]^{0}_{1}(\neg p \wedge \neg r),\ [\![H,B]\!]\Box\neg p]$$

So, $m = 1$, $l = 1$, $r_{\Delta_1} = 2$, and

σ	$N(\sigma)$	$\mathsf{co}(\sigma)$	Γ_σ
$0,0$	$\emptyset$	0	l, $[\![H,B]\!]\Box\neg p$
$0,1$	$\{B\}$	0	l, $[\![H]\!](\neg p \wedge \neg r)$, $[\![H,B]\!]\Box\neg p$
$1,0$	$\{H\}$	0	$[\![H,B]\!]\Box\neg p$
$1,1$	$\{H,B\}$	0	$[\![H]\!](\neg p \wedge \neg r)$, $[\![H,B]\!]\Box\neg p$

4.4 Realization of Eventualities

In the context of ATL$^+$ and ATL*, we consider as *potential eventualities* all γ-formulae containing at least one eventuality operator (U or $\Diamond$). We recall that a γ-formula is of the form $\langle\langle A\rangle\rangle\Phi$ or $[\![A]\!]\Phi$ where $\Phi \neq \bigcirc\varphi$. When constructing a tableau step-by-step as we do in our approach, it is possible to postpone forever promises encapsulated in eventuality operators as far as we keep promising to satisfy them. In order to check realization of potential eventualities, we first introduce a Boolean-valued function called Realized for ATL$^+$, and a slightly different function called WF for ATL*, which returns a path formula. Then, we define what is a descendant potential eventuality, which is next used to decide whether a potential eventuality is realized or not.

4.4.1 Realization of Eventualities for ATL$^+$

In the case of ATL$^+$, the function Realized takes as arguments two elements: an ATL$^+$ path formula Φ and a set Θ of ATL$^+$ state formulae. This function allows us to check the immediate realization of a potential eventuality of the form $\langle\langle A\rangle\rangle\Phi$ and $[\![A]\!]\Phi$ (where Φ is the first argument of Realized) at a given state labelled by Θ (where Θ is the second argument of Realized). The definition of Realized for ATL$^+$ is given by recursion on the structure of Φ as follows:

- $\mathsf{Realized}(\varphi,\Theta) = true$ iff $\varphi \in \Theta$
- $\mathsf{Realized}(\Phi \wedge \Psi,\Theta) = \mathsf{Realized}(\Phi,\Theta) \wedge \mathsf{Realized}(\Psi,\Theta)$
- $\mathsf{Realized}(\Phi \vee \Psi,\Theta) = \mathsf{Realized}(\Phi,\Theta) \vee \mathsf{Realized}(\Psi,\Theta)$
- $\mathsf{Realized}(\bigcirc\varphi,\Theta) = true$
- $\mathsf{Realized}(\Box\varphi,\Theta) = true$
- $\mathsf{Realized}(\varphi \mathsf{U} \psi,\Theta) = true$ iff $\psi \in \Theta$

Example 4.7. The initial tableau $\mathscr{T}_0^\theta$ of θ_3^+ is given in Figure 4.4.

Applications of the function Realized with the eventuality $\langle\langle H\rangle\rangle(((\neg p \wedge \neg r)\mathsf{U} l) \wedge \Diamond r)$ give the following results:

$\mathsf{Realized}(((\neg p \wedge \neg r)\mathsf{U} l) \wedge \Diamond r, \Delta_1) = false$ since $l \notin \Delta_1$ (and $r \notin \Delta_1$)

$\mathsf{Realized}(((\neg p \wedge \neg r)\mathsf{U} l) \wedge \Diamond r, \Delta_5) = false$ since $l \notin \Delta_5$

$\mathsf{Realized}(((\neg p \wedge \neg r)\mathsf{U} l) \wedge \Diamond r, \Delta_6) = true$ since $l \in \Delta_6$ and $r \in \Delta_6$

$\mathsf{Realized}(((\neg p \wedge \neg r)\mathsf{U} l) \wedge \Diamond r, \Delta_7) = false$ since $l \notin \Delta_7$ (and $r \notin \Delta_7$)

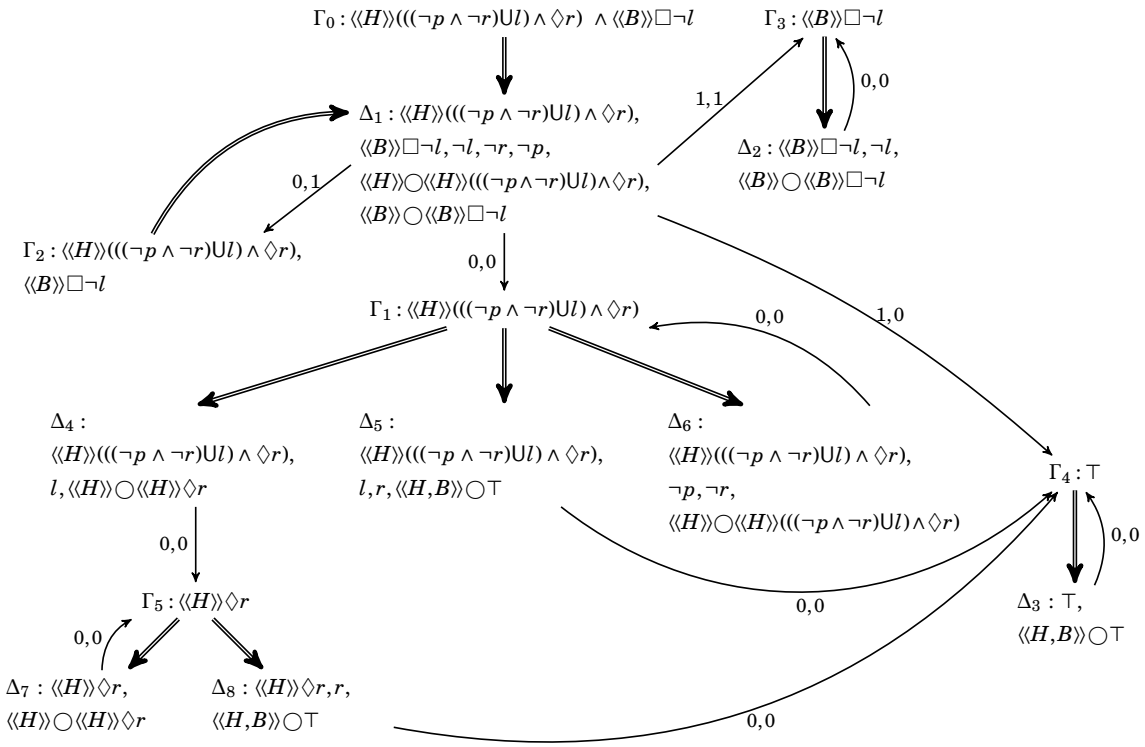


Figure 4.4: Initial tableau $\mathscr{T}_0^\theta$ of $\theta_3^+ = \langle\langle H\rangle\rangle(((\neg p \wedge \neg r)\mathsf{U} l) \wedge \Diamond r) \wedge \langle\langle B\rangle\rangle\Box\neg l$

We will see later, with Definition 4.3, that if the function Realized declares that, at a given state, a formula does not immediately realize all its eventualities, then we look if it is the case in its successor states. But, because of the way γ-formulae are decomposed, an eventuality may change its appearance from one state to another. Therefore, we define the notion of *Descendant potential eventuality* in order to define a parent/child link between potential eventualities and keep track of not yet realized eventualities, and finally check whether the potential eventualities are realized at a given moment.

Definition 4.2 (Descendant potential eventualities)**.** Let Δ be a state and let $\xi \in \Delta$ be a potential eventuality of the form $\langle\langle A\rangle\rangle\Phi$ or $[\![A]\!]\Phi$. Suppose the γ-component $\gamma_l(\xi, \Delta)$ in Δ linked to ξ is, respectively, of the form $\psi \wedge \langle\langle A\rangle\rangle\bigcirc\langle\langle A\rangle\rangle\Psi$ or $\psi \wedge [\![A]\!]\bigcirc[\![A]\!]\Psi$. Then the *successor potential eventuality* of ξ w.r.t. $\gamma_l(\xi, \Delta)$ is the γ-formula $\langle\langle A\rangle\rangle\Psi$ (resp. $[\![A]\!]\Psi$) and it will be denoted by ξ_Δ^1. The notion of *descendant potential eventuality of ξ of degree d*, for $d > 1$, is defined inductively as follows:

- any successor eventuality of ξ (w.r.t. some γ-component of ξ) is a descendant eventuality of ξ of degree 1;
- any successor eventuality of a descendant eventuality ξ^n of ξ of degree n is a descendant eventuality of ξ of degree $n+1$.

We will also consider ξ to be a descendant eventuality of itself of degree 0.

In order to give the definition of *Realization of potential eventualities*, we first adapt the Notation 3.1.

Notation 4.1. Let $\mathbb{L} = [\langle\langle A_0\rangle\rangle\bigcirc\varphi_0, \ldots, \langle\langle A_{m-1}\rangle\rangle\bigcirc\varphi_{m-1}, [\![A'_0]\!]\bigcirc\psi_0, \ldots, [\![A'_{l-1}]\!]\bigcirc\psi_{l-1}, [\![\mathbb{A}]\!]\bigcirc\psi_0, \ldots, [\![\mathbb{A}]\!]\bigcirc\psi_{n-1}]$ be the list of all primitive successor formulae of $\Delta \in S_0^\theta$, induced as part of application of **(Next)**.

1. $Succ(\Delta, \langle\langle A_p\rangle\rangle\bigcirc\varphi_p) := \{\Gamma \mid \Delta \xrightarrow{\sigma} \Gamma, \sigma_{\mathsf{a}} = p \text{ for every } \mathsf{a} \in A_p\}$
2. $Succ(\Delta, [\![A'_q]\!]\bigcirc\psi_q) := \{\Gamma \mid \Delta \xrightarrow{\sigma} \Gamma, \mathsf{co}(\sigma) = q \text{ and } \mathbb{A} - A'_q \subseteq N(\sigma)\}$
3. $Succ(\Delta, [\![\mathbb{A}]\!]\bigcirc\psi_r) := \{\Gamma \mid \Delta \xrightarrow{\sigma} \Gamma\}$

Definition 4.3 (Realization of potential eventualities for ATL^+)**.** Let a state $\Delta \in S_n^\theta$ be a state and $\xi \in \Delta$ be a potential eventuality of the form $\langle\langle A\rangle\rangle\Phi$ or $[\![A]\!]\Phi$. Then:

1. If $\mathsf{Realized}(\Phi, \Delta) = true$ then ξ is realized at Δ in $\mathscr{T}_n^\theta$.
2. Else, let ξ^1_Δ be the successor potential eventuality of ξ w.r.t. $\gamma_l(\xi, \Delta)$. If for every $\Gamma \in Succ(\Delta, \langle\langle A\rangle\rangle\bigcirc\xi^1_\Delta)$ (resp. $\Gamma \in Succ(\Delta, [\![A]\!]\bigcirc\xi^1_\Delta)$), there exists $\Delta' \in \mathscr{T}_n^\theta$ with $\Gamma \Longrightarrow \Delta'$ and ξ^1_Δ is realized at Δ' in $\mathscr{T}_n^\theta$, then ξ is realized at Δ in $\mathscr{T}_n^\theta$.

Example 4.8 (Continuation of example 4.7)**.** Let us consider the two eventualities $\xi = \langle\langle H\rangle\rangle(((\neg p \wedge \neg r)\mathsf{U} l) \wedge \Diamond r)$ and $\zeta = \langle\langle H\rangle\rangle\Diamond r$. We have that

$$\begin{aligned}
&\gamma(\xi, \Delta_1) = \neg p \wedge \neg r \wedge \langle\langle H\rangle\rangle\bigcirc\langle\langle H\rangle\rangle(((\neg p \wedge \neg r)\mathsf{U} l) \wedge \Diamond r) && \text{so } \xi^1_{\Delta_1} = \langle\langle H\rangle\rangle(((\neg p \wedge \neg r)\mathsf{U} l) \wedge \Diamond r)\\
&\gamma(\xi, \Delta_4) = l \wedge \langle\langle H\rangle\rangle\bigcirc\langle\langle H\rangle\rangle\Diamond r && \text{so } \xi^1_{\Delta_4} = \langle\langle H\rangle\rangle\Diamond r\\
&\gamma(\xi, \Delta_6) = \neg l \wedge \neg r \wedge \langle\langle H\rangle\rangle\bigcirc\langle\langle H\rangle\rangle(((\neg p \wedge \neg r)\mathsf{U} l) \wedge \Diamond r) && \text{so } \xi^1_{\Delta_6} = \langle\langle H\rangle\rangle(((\neg p \wedge \neg r)\mathsf{U} l) \wedge \Diamond r)\\
&\gamma(\zeta, \Delta_8) = \langle\langle H\rangle\rangle\bigcirc\langle\langle H\rangle\rangle\Diamond\Box u && \text{so } \zeta^1_{\Delta_8} = \langle\langle H\rangle\rangle\Diamond\Box u
\end{aligned}$$

and

$$\begin{aligned}
&Succ(\Delta_1, \langle\langle H\rangle\rangle\bigcirc\langle\langle H\rangle\rangle(((\neg p \wedge \neg r)\mathsf{U} l) \wedge \Diamond r)) = \{\Gamma_1, \Gamma_2\}\\
&Succ(\Delta_4, \langle\langle H\rangle\rangle\bigcirc\langle\langle H\rangle\rangle\Diamond r) = \{\Gamma_5\}\\
&Succ(\Delta_6, \langle\langle H\rangle\rangle\bigcirc\langle\langle H\rangle\rangle(((\neg p \wedge \neg r)\mathsf{U} l) \wedge \Diamond r)) = \{\Gamma_1\}\\
&Succ(\Delta_8, \langle\langle H\rangle\rangle\bigcirc\langle\langle H\rangle\rangle\Diamond r) = \{\Gamma_5\}
\end{aligned}$$

From these facts, we can deduce that ξ is not realized in Δ_1. Indeed, one of the successor of Δ_1 is Γ_2, whose only successor is Δ_1. So Δ_1 is eliminated by Rule **ER2**. Then, Γ_0 is eliminated by Rule **ER1**, since Δ_1 was the only successor of Γ_0. Note that the rules **ER1** and **ER2** are kept unchanged comparing to the procedure for ATL (see Table 4.1). Therefore, we can conclude that the tableau for θ_3^+ is closed and that θ_3^+ is unsatisfiable.

4.4.2 Realization of Eventualities for ATL*

In the case of ATL*, a promise is a path formula, and we consider that it is immediately satisfied (or realized) once it is initiated at the current state. This corresponds to an engagement to keep it true if necessary, for any computation starting at that state and consistent with the considered strategy. It might happen that even if an ATL* formula is satisfiable, we will not be able to find a node where an adaptation to ATL* of the function Realized returns true. This is because promises, nested in an operator $\Box$ for instance, and contained in potential eventualities can be regenerated faster than they can all be realized. This can be illustrated by Example 4.9. Therefore, instead of deciding if a given potential formula $\xi = \langle\langle A\rangle\rangle\Phi$ is immediately realized at a given state Δ, we define the function WF, that reduces the path formula Φ so to keep only the elements that make Φ not immediately realized at Δ. If ξ is immediately realized, then the function WF applied to Φ will return the path formula $\top$. To check that a potential eventualities is realized, we check that it is possible to reduce Φ to $\top$ in some nodes that contain its descendant potential eventuality, see Definitions 4.4 and 4.5. Indeed, we want to look forward along the computation consistent with the strategy for ξ that the objectives inside Φ are realized. Note that the definition of descendant potential eventualities (Def. 4.2) stays unchanged for ATL*.

Therefore we define the function $\mathsf{WF} : \mathsf{ATL}^*_\mathsf{p} \times \mathscr{P}(\mathsf{ATL}^*_\mathsf{s}) \times \mathscr{P}(\mathsf{ATL}^*_\mathsf{p}) \to \mathsf{ATL}^*_\mathsf{p}$ as follows. The first argument of the function WF is the path formula to study, the second argument is a set of state formulae Θ, and the third argument is a set of initiated path formulae. This third argument is exactly what is added with respect to ATL$^+$ treatment, and corresponds in practice to the set $S = \gamma_{sl}(\Phi,\Theta)$ obtained during the decomposition of Φ and the full expansion of Θ. This last set S is computed in Subsection 4.1.3 and corresponds to the set of path formulae initiated in the current state Θ. The definition of WF for ATL* is given by recursion on the structure of Φ as follows:

- $\mathsf{WF}(\varphi,\Theta,S) = \begin{cases} \top & \text{if } \varphi \in \Theta \\ \varphi & \text{else} \end{cases}$
- $\mathsf{WF}(\Phi_1 \wedge \Phi_2,\Theta,S) = \mathsf{WF}(\Phi_1,\Theta,S) \wedge \mathsf{WF}(\Phi_2,\Theta,S)$
- $\mathsf{WF}(\Phi_1 \vee \Phi_2,\Theta,S) = \mathsf{WF}(\Phi_1,\Theta,S) \vee \mathsf{WF}(\Phi_2,\Theta,S)$
- $\mathsf{WF}(\bigcirc\Phi_1,\Theta,S) = \top$
- $\mathsf{WF}(\Box\Phi_1,\Theta,S) = \top$
- $\mathsf{WF}(\Phi_2 \mathsf{U} \Phi_1,\Theta,S) = \begin{cases} \top & \text{if } \Phi_1 \in \Theta \cup S \\ \Phi_1 \mathsf{U} \Phi_2 & \text{else} \end{cases}$

At the end, we can reduce the result of the function WF by applying the two equivalences $\Phi \wedge \top \equiv \top \wedge \Phi \equiv \Phi$ and $\Phi \vee \top \equiv \top \vee \Phi \equiv \top$.

Remark 4.2. In the last item, we use the set $\Theta \cup S$ for the case when Φ_1 is a state formula that is already in the set Θ because of the behaviour of another coalition of agents.

Example 4.9. The initial tableau $\mathscr{T}_0^\theta$ of θ_4^* is given in Figure 4.5. We first define the two sets $\Xi_1 = \{[\![1]\!]\Box(\neg p \vee \neg q), [\![1]\!]\Box(\neg p \vee \neg r), [\![1]\!]\Box(\neg q \vee \neg r)\}$ and $\Xi_2 = \{[\![1]\!]\Box(\neg p \vee \neg q), [\![1]\!]\Box(\neg p \vee \neg r), [\![1]\!]\Box(\neg q \vee$

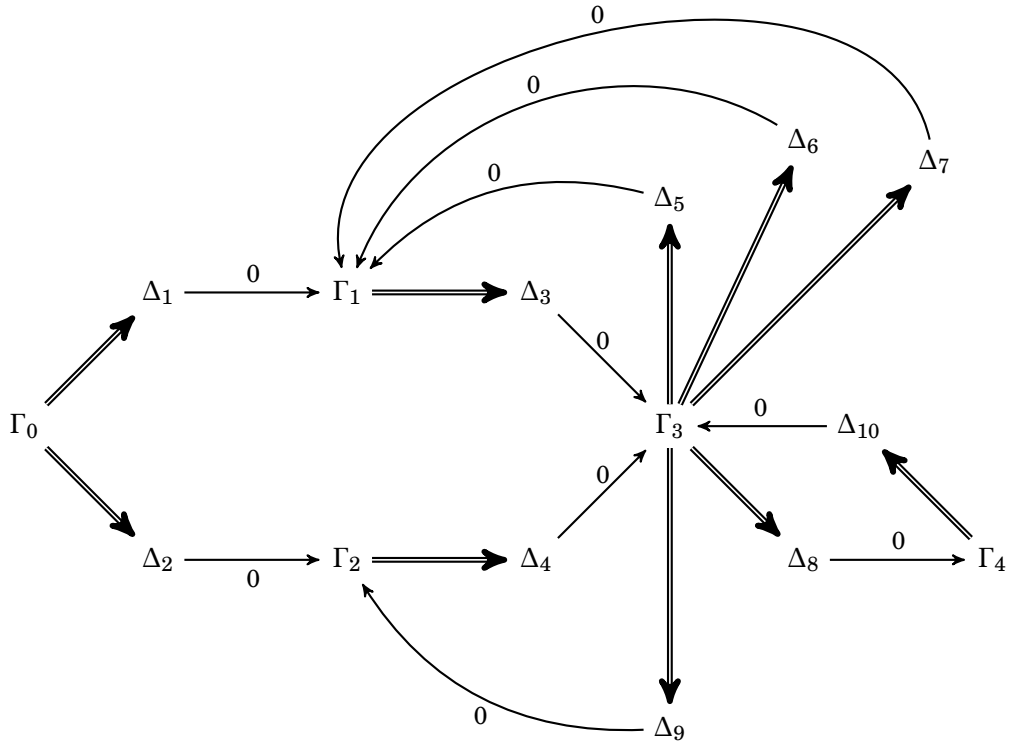


Figure 4.5: Initial tableau $\mathscr{T}_0^\theta$ of $\theta_4^* = [\![1]\!]\Box((p \wedge \bigcirc\neg p) \vee (\neg p \wedge \bigcirc p)) \wedge [\![1]\!]\Box(\neg p \vee \neg q) \wedge [\![1]\!]\Box(\neg p \vee \neg r) \wedge [\![1]\!]\Box(\neg q \vee \neg r) \wedge [\![1]\!]\Box(\Diamond q \wedge \Diamond r) \wedge q$

$\neg r), [\![1]\!]\bigcirc[\![1]\!]\Box(\neg p \vee \neg q), [\![1]\!]\bigcirc[\![1]\!]\Box(\neg p \vee \neg r), [\![1]\!]\bigcirc[\![1]\!]\Box(\neg q \vee \neg r)\}$. Also, in order to save space, we denote by Φ_1 the path formula $\Box((p \wedge \bigcirc\neg p) \vee (\neg p \wedge \bigcirc p))$, and by Φ_2 the path formula $\Box(\Diamond q \wedge \Diamond r)$. Therefore we label prestates and states in $\mathscr{T}_0^\theta$ of θ_4^* as follows:

$$\Gamma_0 = \{\theta_4^*\}$$

$$\Gamma_1 = \Xi_1 \cup \{[\![1]\!](p \wedge \Phi_1), [\![1]\!](\Phi_2 \wedge \Diamond q \wedge \Diamond r)\} \qquad \Gamma_2 = \Xi_1 \cup \{[\![1]\!](p \wedge \Phi_1), [\![1]\!](\Phi_2 \wedge \Diamond r)\}$$

$$\Gamma_3 = \Xi_1 \cup \{[\![1]\!](\neg p \wedge \Phi_1), [\![1]\!](\Phi_2 \wedge \Diamond q \wedge \Diamond r)\} \qquad \Gamma_4 = \Xi_1 \cup \{[\![1]\!](p \wedge \Phi_1), [\![1]\!](\Phi_2 \wedge \Diamond q)\}$$

and

$$\Delta_1 = \Xi_2 \cup \{\theta_4^*, \neg p, q, \neg r, [\![1]\!]\Phi_1, [\![1]\!]\Phi_2, [\![1]\!]\bigcirc[\![1]\!](p \wedge \Phi_1), [\![1]\!]\bigcirc[\![1]\!](\Phi_2 \wedge \Diamond q \wedge \Diamond r)\}$$

$$\Delta_2 = \Xi_2 \cup \{\theta_4^*, \neg p, q, \neg r, [\![1]\!]\Phi_1, [\![1]\!]\Phi_2, [\![1]\!]\bigcirc[\![1]\!](p \wedge \Phi_1), [\![1]\!]\bigcirc[\![1]\!](\Phi_2 \wedge \Diamond r)\}$$

$$\Delta_3 = \Xi_2 \cup \{p, \neg q, \neg r, [\![1]\!](p \wedge \Phi_1), [\![1]\!](\Phi_2 \wedge \Diamond q \wedge \Diamond r), [\![1]\!]\bigcirc[\![1]\!](\neg p \wedge \Phi_1), [\![1]\!]\bigcirc[\![1]\!](\Phi_2 \wedge \Diamond q \wedge \Diamond r)\}$$

$$\Delta_4 = \Xi_2 \cup \{p, \neg q, \neg r, [\![1]\!](p \wedge \Phi_1), [\![1]\!](\Phi_2 \wedge \Diamond r), [\![1]\!]\bigcirc[\![1]\!](\neg p \wedge \Phi_1), [\![1]\!]\bigcirc[\![1]\!](\Phi_2 \wedge \Diamond q \wedge \Diamond r)\}$$

$$\Delta_5 = \Xi_2 \cup \{\neg p, \neg q, \neg r, [\![1]\!](\neg p \wedge \Phi_1), [\![1]\!](\Phi_2 \wedge \Diamond q \wedge \Diamond r), [\![1]\!]\bigcirc[\![1]\!](p \wedge \Phi_1), [\![1]\!]\bigcirc[\![1]\!](\Phi_2 \wedge \Diamond q \wedge \Diamond r)\}$$

$$\Delta_6 = \Xi_2 \cup \{\neg p, \neg q, [\![1]\!](\neg p \wedge \Phi_1), [\![1]\!](\Phi_2 \wedge \Diamond q \wedge \Diamond r), [\![1]\!]\bigcirc[\![1]\!](p \wedge \Phi_1), [\![1]\!]\bigcirc[\![1]\!](\Phi_2 \wedge \Diamond q \wedge \Diamond r)\}$$

$$\Delta_7 = \Xi_2 \cup \{\neg p, \neg r, [\![1]\!](\neg p \wedge \Phi_1), [\![1]\!](\Phi_2 \wedge \Diamond q \wedge \Diamond r), [\![1]\!]\bigcirc[\![1]\!](p \wedge \Phi_1), [\![1]\!]\bigcirc[\![1]\!](\Phi_2 \wedge \Diamond q \wedge \Diamond r)\}$$
$$\Delta_8 = \Xi_2 \cup \{\neg p, \neg q, r, [\![1]\!](\neg p \wedge \Phi_1), [\![1]\!](\Phi_2 \wedge \Diamond q \wedge \Diamond r), [\![1]\!]\bigcirc[\![1]\!](p \wedge \Phi_1), [\![1]\!]\bigcirc[\![1]\!](\Phi_2 \wedge \Diamond q)\}$$
$$\Delta_9 = \Xi_2 \cup \{\neg p, q, \neg r, [\![1]\!](\neg p \wedge \Phi_1), [\![1]\!](\Phi_2 \wedge \Diamond q \wedge \Diamond r), [\![1]\!]\bigcirc[\![1]\!](p \wedge \Phi_1), [\![1]\!]\bigcirc[\![1]\!](\Phi_2 \wedge \Diamond r)\}$$
$$\Delta_{10} = \Xi_2 \cup \{p, \neg q, \neg r, [\![1]\!](p \wedge \Phi_1), [\![1]\!](\Phi_2 \wedge \Diamond q), [\![1]\!]\bigcirc[\![1]\!](\neg p \wedge \Phi_1), [\![1]\!]\bigcirc[\![1]\!](\Phi_2 \wedge \Diamond q \wedge \Diamond r)\}$$

In the following, we denote by

$$\xi_1 = [\![1]\!]\Box(\Diamond q \wedge \Diamond r) \qquad \xi_2 = [\![1]\!]\Box(\Diamond q \wedge \Diamond r) \wedge \Diamond q$$
$$\xi_3 = [\![1]\!]\Box(\Diamond q \wedge \Diamond r) \wedge \Diamond r \qquad \xi_4 = [\![1]\!]\Box(\Diamond q \wedge \Diamond r) \wedge \Diamond q \wedge \Diamond r$$

So we have

$$\gamma_{sl}(\xi_1, \Delta_1) = \gamma_{sl}(\xi_4, \Delta_3) = \gamma_{sl}(\xi_3, \Delta_4) = \gamma_{sl}(\xi_2, \Delta_{10}) = \{\Diamond q \wedge \Diamond r\} = \gamma_s^1$$
$$\gamma_{sl}(\xi_1, \Delta_2) = \gamma_{sl}(\xi_4, \Delta_9) = \{\Diamond q \wedge \Diamond r, q\} = \gamma_s^2$$
$$\gamma_s(\xi_4, \Delta_8) = \{\Diamond q \wedge \Diamond r, r\} = \gamma_s^3$$

Let us see some applications of the function WF with the initial tableau $\mathscr{T}_0^\theta$.

$$\mathsf{WF}(\Box(\Diamond q \wedge \Diamond r), \Delta_1, \gamma_s^1) = \top$$
$$\mathsf{WF}(\Box(\Diamond q \wedge \Diamond r), \Delta_2, \gamma_s^2) = \top$$
$$\begin{aligned}\mathsf{WF}(\Box(\Diamond q \wedge \Diamond r) \wedge \Diamond q \wedge \Diamond r, \Delta_3, \gamma_s^1) &= \mathsf{WF}(\Box(\Diamond q \wedge \Diamond r), \Delta_3, \gamma_s^1) \wedge \mathsf{WF}(\Diamond q \wedge \Diamond r, \Delta_3, \gamma_s^1)\\ &= \top \wedge \mathsf{WF}(\Diamond q, \Delta_3, \gamma_s^1) \wedge \mathsf{WF}(\Diamond r, \Delta_3, \gamma_s^1)\\ &= \Diamond q \wedge \Diamond r \qquad \text{since } q \notin \gamma_s^1 \text{ and } r \notin \gamma_s^1\end{aligned}$$
$$\begin{aligned}\mathsf{WF}(\Box(\Diamond q \wedge \Diamond r) \wedge \wedge \Diamond r, \Delta_4, \gamma_s^1) &= \mathsf{WF}(\Box(\Diamond q \wedge \Diamond r), \Delta_4, \gamma_s^1) \wedge \mathsf{WF}(\Diamond r, \Delta_4, \gamma_s^1)\\ &= \top \wedge \Diamond r \qquad \text{since } r \notin \gamma_s^1\end{aligned}$$
$$\mathsf{WF}(\Box(\Diamond q \wedge \Diamond r) \wedge \Diamond q \wedge \Diamond r, \Delta_8, \gamma_s^3) = \Diamond q \qquad \text{since } q \in \gamma_s^1 \text{ and } r \notin \gamma_s^3$$
$$\mathsf{WF}(\Box(\Diamond q \wedge \Diamond r) \wedge \Diamond q \wedge \Diamond r, \Delta_9, \gamma_s^2) = \Diamond r \qquad \text{since } q \notin \gamma_s^1 \text{ and } r \in \gamma_s^3$$
$$\mathsf{WF}(\Box(\Diamond q \wedge \Diamond r) \wedge \Diamond q, \Delta_8, \gamma_s^3) = \Diamond q \qquad \text{since } q \in \gamma_s^1 \text{ and } r \notin \gamma_s^3$$

In this example, from the state Δ_3, there is no successor states that follows the linked potential eventualities of $\xi = \langle\langle H \rangle\rangle(\Box(\Diamond q \wedge \Diamond r) \wedge \Diamond q \wedge \Diamond r)$ where the function WF returns $\top$ as result, which would have correspond to the result of the function Realized being true. Therefore, applying definition 4.3 for ATL$^+$, would have lead to remove all the states of $\mathscr{T}_0^\theta$ and declares θ_4^* unsatisfiable, which is wrong as we will see with example 4.10.

Definition 4.4 (Path-realization)**.** Let $\Delta \in S_n^\theta$ be a state and $\xi \in \Delta$ be a potential eventuality of the form $\langle\langle A \rangle\rangle\Phi$ or $[\![A]\!]\Phi$. Let $S = \gamma_{sl}(\xi, \Delta)$. Let Ψ be a path formula[1].

1. If $\mathsf{WF}(\Psi, \Delta, S) = \top$ then Ψ is path-realized at Δ w.r.t ξ in $\mathscr{T}_n^\theta$.

[1]Differently from Definition 4.3, here Ψ is *any* path formula.

2. Else, let ξ^1 be the successor potential eventuality of ξ w.r.t $\gamma_{sl}(\xi,\Delta)$. If for every $\Gamma \in Succ(\Delta, \langle\langle A\rangle\rangle \bigcirc \xi^1_\Delta)$ (resp. $\Gamma \in Succ(\Delta, [\![A]\!] \bigcirc \xi^1_\Delta)$, there exists $\Delta' \in \mathscr{T}^\theta_n$ with $\Gamma \Rightarrow \Delta'$ and $\Psi' = \mathsf{WF}(\Psi, \Delta, S)$ is path-realized at Δ' w.r.t ξ^1_n at Δ in $\mathscr{T}^\theta_n$, then Ψ is path-realized at Δ w.r.t ξ in $\mathscr{T}^\theta_n$.

Definition 4.5 (Realization of potential eventualities for ATL^*)**.** Let $\Delta \in S^\theta_n$ be a state, and $\xi \in \Delta$ be a potential eventuality of the form $\langle\langle A\rangle\rangle\Phi$ or $[\![\langle\langle A\rangle\rangle]\!]\Phi$. Then ξ is realized at Δ in $\mathscr{T}^\theta_n$ if Φ is path-realized at Δ w.r.t ξ in $\mathscr{T}^\theta_n$.

Remark 4.3. In Definition 4.4, we use the descendant potential eventuality ξ_1 to follow the strategy that has been deployed during the construction phase to satisfy ξ. Therefore, it is possible to check if the objective Ψ is fully realized with regards to that strategy, even if Ψ is at that level independent of ξ (the link is made in Definition 4.5).

Example 4.10 (Continuation of Example 4.9)**.** Let us consider the eventuality $\xi = [\![1]\!](\Box(\Diamond q \wedge \Diamond r) \wedge \Diamond q \wedge \Diamond r)$ in the state Δ. As seen in Example 4.9, this eventuality is not immediately realized. First have that $\gamma(\xi, \Delta_3) = [\![1]\!] \bigcirc [\![1]\!](\Box(\Diamond q \wedge \Diamond r) \wedge \Diamond q \wedge \Diamond r$, so $\xi^1_{\Delta_3} = [\![1]\!](\Box(\Diamond q \wedge \Diamond r) \wedge \Diamond q \wedge \Diamond r)$ and $Succ(\Delta_3, [\![1]\!] \bigcirc [\![1]\!](\Box(\Diamond q \wedge \Diamond r) \wedge \Diamond q \wedge \Diamond r)) = \{\Gamma_3\}$. Then let us consider the successor Δ_8 of Γ_3 where we obtain $\mathsf{WF}(\Diamond q \wedge \Diamond r, \Delta_8, \gamma_{sl}(\xi^1_{\Delta_3}, \Delta_8)) = \Diamond q$.

Then, we repeat the same process with the only state successor of Δ_8, that is Δ_{10}, where $\mathsf{WF}(\Diamond q, \Delta_{10}, \gamma_{sl}(\xi^1_{\Delta_8}, \Delta_{10})) = \Diamond q$, and whose prestate successor is again Γ_3. So, here, we continue by visiting another successor of Γ_3, say Δ_9, and we compute $\mathsf{WF}(\Diamond q, \Delta_9, \gamma_{sl}(\xi^1_{\Delta_{10}}, \Delta_9))$ which returns $\top$. Therefore, we can conclude that ξ is realized at Δ_3 in $\mathscr{T}^\theta_0$.

In the same way, we can conclude that ξ is realized at Δ_8 and Δ_9 in $\mathscr{T}^\theta_0$, and that $\xi' = [\![1]\!](\Box(\Diamond q \wedge \Diamond r) \wedge \Diamond r)$ is realized at Δ_4 in $\mathscr{T}^\theta_0$ and that $\xi'' = [\![1]\!](\Box(\Diamond q \wedge \Diamond r) \wedge \Diamond q)$ is realized at Δ_{10} in $\mathscr{T}^\theta_0$

Thus, we can conclude that the tableau for θ^*_4 is open and that θ^*_4 is satisfiable.

4.5 Complexity

In the following, we denote by $|\psi|$ the length of the formula ψ, and by $\|\Gamma\|$ the cardinality of the set or list Γ.

4.5.1 Complexity of the Procedure for ATL^+

Lemma 4.2. *For any* ATL^+ *state formula* φ, $\|\mathsf{cl}(\varphi)\| < 2^{|\varphi|^2}$.

Proof. Every formula in $cl(\varphi)$ has length less than $2|\varphi|$,and is built from symbols in φ, so there can be at most $|\varphi|^{2|\varphi|} = 2^{2|\varphi|\log_2|\varphi|} < 2^{|\varphi|^2}$ such formulae. ■

The estimate above is rather crude, but $\|\mathsf{cl}(\varphi)\|$ *can* reach size exponential in $|\varphi|$. Indeed, consider the formulae $\phi_k = \langle\langle 1\rangle\rangle(p_1 \mathsf{U} q_1 \wedge (p_2 \mathsf{U} q_2 \wedge (\ldots \wedge p_k \mathsf{U} q_k)\ldots)$ for $k = 1, 2, \ldots$ and distinct

$p_1, q_1, \ldots, p_k, q_k, \ldots \in \mathbb{P}$. Then $|\phi_k| = O(k)$, while the number of different γ-components of ϕ_k is 2^k, hence $\|\mathsf{cl}(\phi_k)\| > 2^k$.

Theorem 4.1. *The tableau-based procedure for* ATL$^+$ *runs in at most* 2EXPTIME.

Proof. The argument generally follows the calculations computing the complexity of the tableau method for ATL in Section 4.7 of [31], with one essential difference: $\|\mathsf{cl}(\theta)\|$ for any ATL formula θ is linear in its length $|\theta|$, whereas $\|\mathsf{cl}(\theta)\|$ for an ATL$^+$ formula θ can be exponentially large in $|\theta|$, as shown after Lemma 4.2. This exponential blow-up, combined with the worst-case exponential in $\|\mathsf{cl}(\theta)\|$ number of states in the tableau, accounts for the 2EXPTIME worst-case complexity of the tableau method for ATL$^+$, which is the expected optimal lower bound. It is also an upper bound for the tableau method, because no further exponential blow-ups occur in the elimination phase. ■

4.5.2 Complexity of the Procedure for ATL*

If we apply the same reasoning for ATL* procedure, we obtain a 3EXPTIME complexity, which is suboptimal. But, in an ongoing work with Sven Schewe, we manage to obtain a 2EXPTIME complexity.

Theorem 4.2. *The tableau-based procedure for* ATL* *runs in at most* 2EXPTIME.

Proof. Let θ be the initial formula of a tableau. We now consider each occurrence of the block $\langle\langle \ldots \rangle\rangle$ or $[\![\ldots]\!]$ as a unique symbol.

First, we count the number of possible prestates in a graph, and then the maximum number of successor states that a prestate may have.

Number of prestates: With the exception of the initial prestate, we can consider that all prestates are sets that contain formulae of the form $\langle\langle A \rangle\rangle \Phi$ or $[\![A]\!]\Phi$. This choice is without loss of generality, because state formulae different from $\langle\langle A \rangle\rangle \Phi$ or $[\![A]\!]\Phi$ can easily be transformed into that form without modifying the satisfiability of the initial formula. It suffices to add the path quantifier linked to the successor formula from which they derive. Also note that in the path formula Φ the inner state formulae are not decomposed by $\mathsf{dec}^*(\Phi)$.

Let Ξ be the list of all the sub-expressions of θ that have as main operator a temporal operator, ordered by their position in the formula tree of θ. Also, let Λ be the list of occurrences of all path quantifiers ordered in the same way. The size of Ξ and the size of Λ are at most $|\theta|$. For example, if $\theta = \langle\langle 1 \rangle\rangle(\Box\Diamond p \vee \Diamond\langle\langle 2 \rangle\rangle(p \mathsf{U} q)) \wedge \langle\langle 1 \rangle\rangle\langle\langle 1 \rangle\rangle\Diamond p)$, then $\Xi = [\Box\Diamond p, \Diamond p, \Diamond\langle\langle 2 \rangle\rangle(p \mathsf{U} q), p \mathsf{U} q, \Diamond p]$ and $\Lambda = [\langle\langle 1 \rangle\rangle, \langle\langle 2 \rangle\rangle, \langle\langle 1 \rangle\rangle]$.

In the initial formula θ, all the elements of Ξ are in the scope of an element of Λ. Let $\Xi[i]$ be the sub-list of Ξ in the scope of a $\Lambda(i)$ (the ith element of Λ). Let $\mathsf{cnf}(\Xi(i))$ be the set of possible conjunctions of disjunctions of elements of $\Xi(i)$, without redundancy, as described in Remark 4.1.

The number of elements in the set $\mathsf{cnf}(\Xi(i))$ is at most $2^{2^{\|\Xi(i)\|}}$. Moreover, each prestate is composed of at most $\|\Lambda\|$ formulae of the form $\Lambda_i(\mathsf{cnf}(\Xi_i))$. Therefore, there are at most

$$\prod_{1\leq i\leq \|\Lambda\|} 2^{2^{\|\Xi_i\|}} = (2^{2^{\|\Xi\|}})^{\|\Lambda\|} = 2^{\|\Lambda\| 2^{\|\Xi_i\|}} < 2^{|\theta| 2^{|\theta|}} < 2^{2^{(|\theta|^2)}} \tag{4.8}$$

prestates.

Number of successor states for each prestate Γ**:** In each successor state, we have

- that each atomic proposition can be either present or absent from the state, which gives at most $n_1 = 2^{|\theta|}$ possibilities, since there are at most $|\theta|$ atomic propositions.
- that each state formula of the form $\langle\langle A\rangle\rangle\Phi$ or $[\![A]\!]\Phi$ in the prestate Γ can be either present or absent from the state, which gives at most $n_2 = 2^{|\theta|}$ possibilities, since there are at most $|\theta|$ such state formulae.
- that each successor formula is linked to a state formula of the form $\langle\langle A\rangle\rangle\Phi$ or $[\![A]\!]\Phi$ present in the state, and this successor formula is taken among $2^{2^{|\theta|}}$ possible successor formulae, which gives at most $n_3 = (2^{2^{|\theta|}})^{|\theta|} = (2^{|\theta| 2^{|\theta|}}) < 2^{2^{(|\theta|^2)}}$ possibilities.

Therefore, there are at most

$$n_1 \times n_2 \times n_3 = 2^{|\theta|} \times 2^{|\theta|} \times 2^{2^{(|\theta|^2)}} = 2^{2|\theta|} \times 2^{2^{(|\theta|^2)}} < 2^{2^{(|\theta|^2)}+2|\theta|} \tag{4.9}$$

successor states for each prestate.

Thus the size of the initial tableau (and therefore of the final tableau, which is smaller or equal to the initial tableau) is at most doubly exponential in the size of θ.

Elimination phase: Each node of the tableau can potentially contain at most $|\theta|$ eventualities. In the worst case, it will be needed to go through all edges of the tableau to check whether the eventuality is realized or not. There are at most $2^{2^{|\theta|}}$ nodes in the tableau (as seen just above) and therefore there are at most $(2^{2^{|\theta|}})^2 = 2^{2^{|\theta|+1}}$ edges in the tableau. So, to check realization of all the eventualities in the tableau is at most

$$|\theta| \times 2^{2^{|\theta|}} \times 2^{2^{|\theta|+1}} < 2^{2^{|\theta|}+2^{|\theta|+1}} < 2^{2^{|\theta|+2}}. \tag{4.10}$$

■

4.6 Conclusion

We manage to extend the tableau-based decision procedure for ATL to ATL^+ and ATL^* by mainly modifying the decomposition of non-primitive formulae and dealing with more complex eventualities. These procedures are optimal, since they run in 2EXPTIME, which is the complexity of the satisfiability problem for both ATL^+ and ATL^*. These procedures provide a solution to the satisfiability problem only for agents having perfect-recall strategies. Up to our knowledge, it is not known if this problem can be solved using memoryless strategies. In our opinion, it is at least

very difficult to find a solution, indeed by looking to example 2.4, it seems that we need to define several sets of formulae to be satisfied at the same state, but which are not active at the same moment. To be more precise, at state s_0, at instant t_0 we want to satisfy $\langle\langle a\rangle\rangle\Diamond(p \wedge \langle\langle a\rangle\rangle\Diamond q)$ – note that the first eventuality cannot immediately be realized–, whereas at the instant t_2, after having visited s_1 at t_1, we want to satisfy the formula $\langle\langle a\rangle\rangle\Diamond q$, and no more the formula $\langle\langle a\rangle\rangle\Diamond(p \wedge \langle\langle a\rangle\rangle\Diamond q)$ which has been realized in t_1. Therefore, the difficulty seems to be able to detect that two or more sets of formulae must be joined into the same state of the tableau.

In the next Chapter, we present our implementation of the tableau-based decision procedure for ATL*.

CHAPTER 5

IMPLEMENTATION

In order to test our tableau-based decision procedure for ATL*, we have developed a prototype in Ocaml. This prototype[1] is the extension of the one that we proposed in 2013 to decide satisfiability of ATL [19]. These prototypes are, up to our knowledge, the first available tools to decide satisfiability of ATL$^+$/ATL* and ATL formulae, respectively. We have called these prototypes TATL which stands for "Tableaux for ATL$^{(*)}$". Note that these prototypes test tight satisfiability of ATL and ATL*.

TATL is available as a command line application, and also as a web application. Web applications have the advantage to be directly usable without download and installation, and to be user-friendly. On the other side, the command line version allows one to benefit from the functionalities of Unix commands.

We first describe how one can use our prototype, then the different data structures we use, as well as some relevant algorithms. Finally, we make some comparison with the CTL* reasoner[2] developed by M. Reynolds [56].

5.1 The Application TATL

5.1.1 Web Application

Our prototype to decide satisfiability of formulae of the alternating-time temporal logic's family is very simple to use. It suffices to enter an ATL* formula in the editor (#1 on Figure 5.1), and click on the button "Launch" (#2). Different buttons have been added to help the user to input a formula (#3). Remember that ATL* formulae cover ATL and ATL$^+$ formulae. In this prototype,

[1] http://atila.ibisc.univ-evry.fr/tableau_ATL_star/

[2] http://www.csse.uwa.edu.au/~mark/research/Online/quicktab/

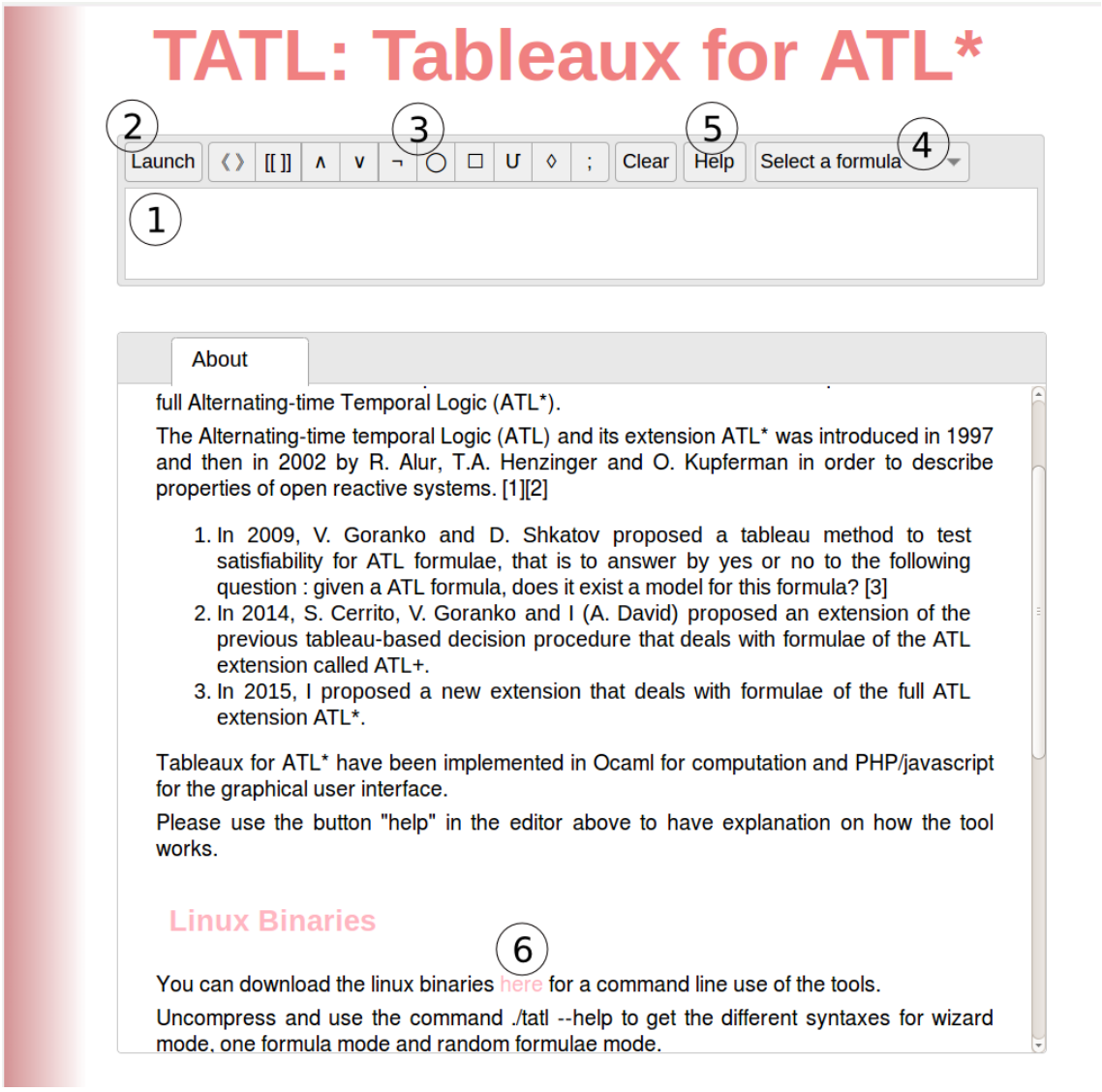


Figure 5.1: Home page of the application TATL

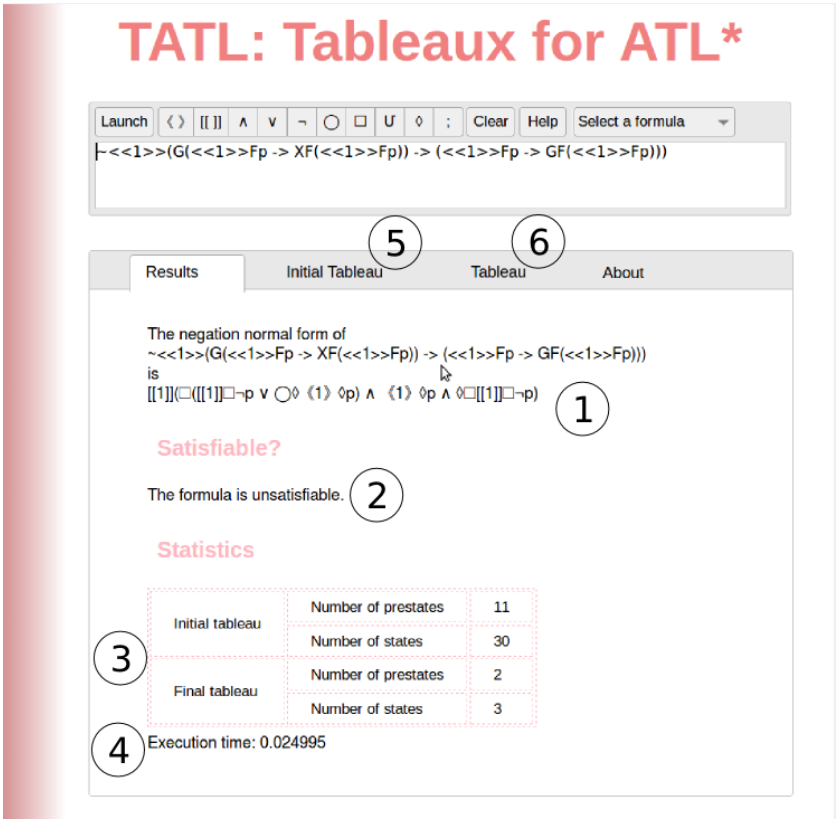


Figure 5.2: Result page of the application TATL

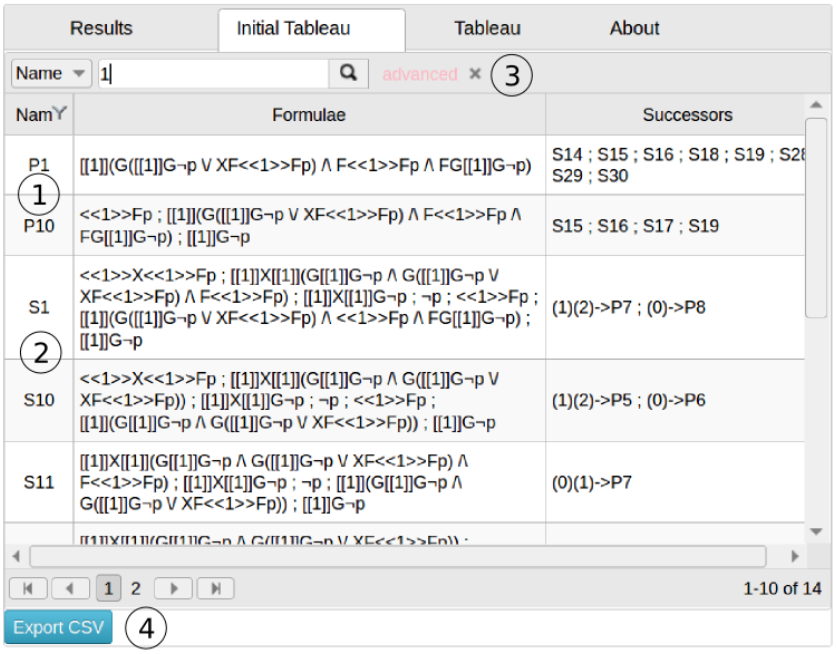


Figure 5.3: Initial tableau tabulation

we use the following convention for the syntax of formulae:

$$\text{State formula } S := p \mid \neg S \mid (S_1 /\backslash S_2) \mid (S_1 \backslash/ S_2) \mid (S_1 -> S_2) \mid (S_1 <-> S_2) \mid << A >> P \mid [[A]]P$$
$$\text{Path formula } P := S \mid (P_1 /\backslash P_2) \mid (P_1 \backslash/ P_2) \mid (P_1 -> P_2) \mid (P_1 <-> P_2) \mid XP \mid GP \mid FP \mid P_1 U P_2$$

It is also possible to enter several formulae at the same time by separating them with the character ";".

In order to give some example formulae to the user, we provide two lists of formulae that can be selected (#4 in Fig. 5.1). The first list contains ATL formulae that have been used to test the first version of TATL, and the second list contains the CTL* formulae proposed by M. Reynolds[3] that we have transformed into ATL* formulae. Once a formula is selected, it is written in the editor and the user has the possibility to modify it before clicking on the button "launch" to start the computation.

Once the computation is done, the so obtained result is displayed on the interface on three tabs: *Results*, *Initial Tableau* (#5) and *Tableau* (#6)(Figure 5.2).

The tab "Results" displays data about the computation:

- the negation normal form of the given formula (#1);
- the main result, that is whether the given formula is satisfiable or not (#2);
- the number of prestates displayed in the tab " Final Tableau" and the number of states displayed in the tab "Tableau"(#3);
- the execution time of the computation. Note that time for network traffic and time for displaying data on the interface are not taken into account in this execution time (#4).

The tabs "Initial Tableau" and "Tableau" are structured in the same way (Figure 5.3). In the middle part, we can find the tableau itself: each line corresponds to a prestate (#1) if its name begins with "P", or a state (#2) if its name begins with "S". Then each line is composed of the corresponding set of state formulae and of its successors. Successors of states are formatted as follows

$$\text{(action vector 1)}\ldots\text{(action vector } j\text{)} \rightarrow \text{name of the successor (prestate)}$$

It is also possible to filter the lines of the tableau by using the simple or advanced filter tool (#3, Fig. 5.3) and to extract data of the tableau into a csv file (#4).

Note that, at any time, the button "help" (#5 on Figure 5.1) can be clicked to obtain some explanation on how to use the web version of TATL.

5.1.2 Command Line Application

Binaries of the application can be downloaded from the web site (#6 on Figure 5.1). The two commands to use TATL are given in Listing 5.1. The syntax used to enter a formula is the same as

[3] http://www.csse.uwa.edu.au/~mark/research/Online/quicktab/quicktablong.pdf

the one for the web version. With the command line version, it is possible to choose the verbatim mode or not. In the verbatim mode (option `-v`), the application displays the initial tableau and the final tableau, in addition to the statistic data. The way to read the result is the same as for the web version.

Listing 5.1: commands for TATL

```
./tatl.native [-v]
./tatl.native -o [-v] [-f string]
```

5.2 General Organisation of the Application

For the web version, the connexion with the Ocaml program is done by using PHP and Ajax. The web interface is based on the jQuery framework jQwidgets[4].

The Ocaml program, which represents about 1900 lines, is divided in several modules, each one having its proper role. Figure 5.4 gives an overview of the general organisation of the application and the role of each module.

All data structures, as well as global functions and exception type declarations are included in the modules "Modules.ml", "Global.ml" and "Except.ml", respectively.

Formulae received by the Ocaml program are parsed into an Ocaml type formulae with the modules "Lexer_formula.mll" and "Parser_formula.mly", which are based on Ocamllex and Ocamlyacc[5], as well as the module "Transformation_frm.ml" to transform the formula into the ATL* grammar (Equation 2.4). The module "Pretty_printer.ml" transforms Ocaml type formulae, as well as states, the other way around.

The module "Vertex_states.ml" is the backbone of the application, indeed it is this module that organizes the construction of the tableau.

The modules "Construction.ml", "Decomposition.ml" and "Elimination.ml", contain respectively the code that corresponds to the construction rules, namely rule SR and rule Next, the code to decompose non-primitive formulae and the code that corresponds to the elimination rules, namely rule ER1 and rule ER2.

There are two main programs: "Tatl.ml" for the command line version and "One_shot.ml" for the web version of TATL.

5.3 Data Structures

The main data structure of our application is the directed graph that represents the tableau. To encode this structure, we use the *Ocamlgraph API*[6] and, in particular, the package "Imperative"

[4] http://www.jqwidgets.com/

[5] http://caml.inria.fr/pub/docs/manual-ocaml-4.00/manual026.html

[6] http://ocamlgraph.lri.fr/

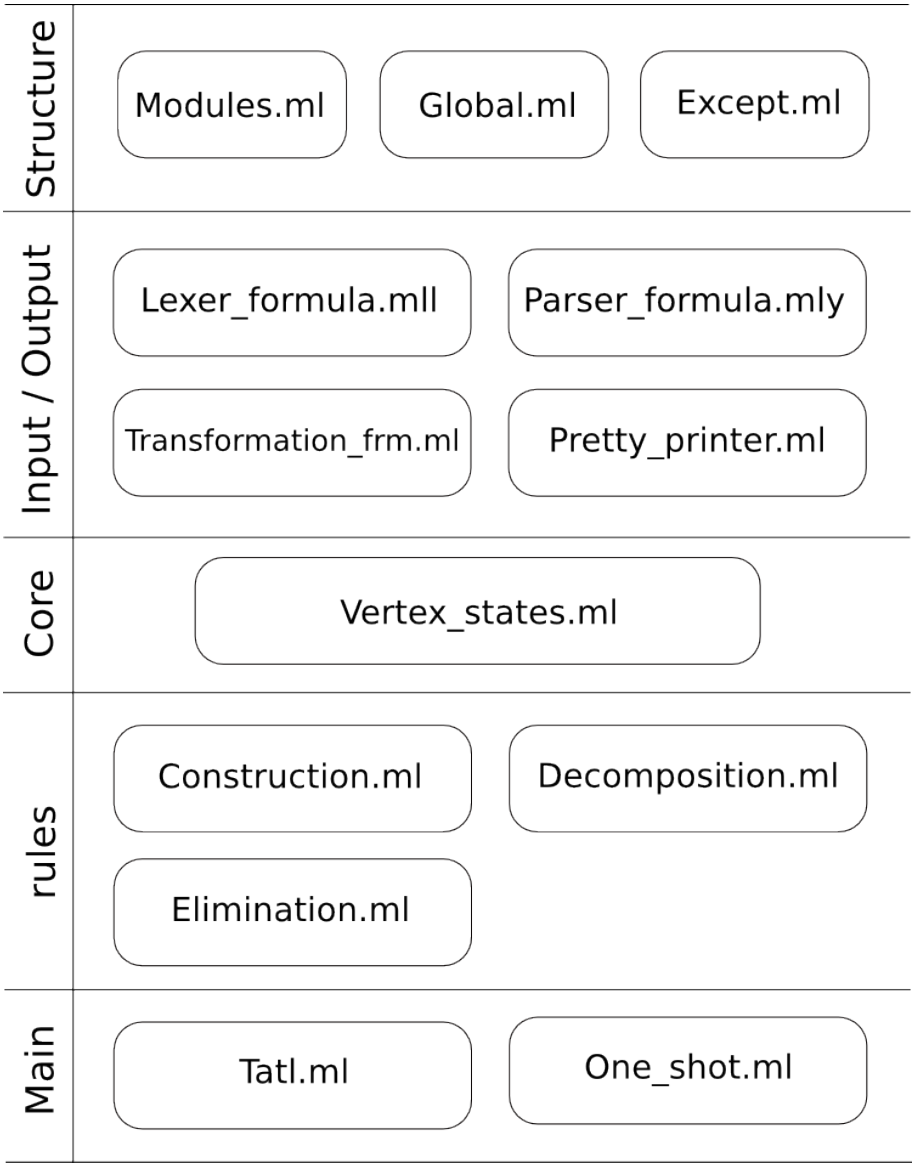


Figure 5.4: Structure of the code of the application

which allows us to easily make many modifications on the graph in order to add and remove vertices or edges. The graph is then composed of *vertices* (the nodes) and *edges*, each of them having, amongst other things, a *type* describing their specific data structure. Listing 5.2 gives the type for nodes and edges of our tableau.

Listing 5.2: type for nodes and vertex of ATL* tableau

```
1  type vertex =
2  {
3    name : string ;
4    category : categ ;
5    ens_frm : State_Formulae.t ;
6    event: (formula_tuple) list;
7    assoc_movecs : Movecs.t ;
8    lst_next_pos : (int * state_formula) list;
9    lst_next_neg : (int * state_formula) list;
10   lst_next_agents: state_formula list;
11   nb_pos : int;
12   nb_neg : int;
13 }
14 type edge = Movecs.t
```

Remark 5.1. We use several sets in our data structure. Our sets are built with the *Set.Make* functor as implementations of the *Set* module in Ocaml. Sets have the advantages of ordering data and avoiding duplicated elements automatically.

For each vertex, we define:

line 3 a name, which consists in a "S" if the vertex is a state and a "P" if the vertex is a prestate plus a number incremented each time we add a vertex.

line 4 a category: prestate or state.

line 5 the set of state formulae associated to the node.

line 6 the list of eventualities associated to the node. Each eventuality ξ at a given state Δ is represented as a triple composed of the γ-formula corresponding to ξ, a set of path formulae resulting from the γ-decomposition of ξ at Δ that is $\gamma_{sl}(\xi,\Delta)$, and the successor formula associated to ξ, that is the successor formula in $\gamma_l(\xi,\Delta)$.

line 7 the set of action vectors associated to the state. This set depends exclusively of the number of successor formulae in the state and the number of agents $\mathbb{A}$. During the elimination phase, it will be used for comparison with the set of action vectors associated to the outgoing edge of the state, and to check that no action vectors are missing.

lines 8, 9 & 10 a list of enforceable successor formulae, a list of proper unavoidable successor formulae with their position number in the list, as well as the list of successor formulae

whose coalition is the set of all agents. These position numbers correspond to the ordering defined in the rule Next, and will be used, as well as information on lines 11 and 12, to create the successor edges and vertices of the node.

lines 11 & 12 the number of positive successor formulae and the number of negative formulae.

Each edge is defined by a set of actions vectors (**line 14**).

It is worthwhile noticing that we also use several hashtables in order to easily refer to elements of the computation that have been done previously. For instance, we keep in memory the results of every decomposition function, so if one of these decompositions is again needed, we do not have to compute it a second time.

5.4 Relevant Algorithms: State and Prestate Elimination

In this section, we will give more details about the implementation of the phase of elimination. In our opinion, it is the less intuitive part of the implementation.

A source of increased execution time comes from the search of nodes to delete in the graph each time we need to apply the rule ER1 or ER2. Therefore, in the implementation, we have mixed the two rules in one algorithm. We check both the rules ER1 and ER2 for each node (Algorithm 1). Also when we delete a node, if some successor node becomes disconnected from the graph, we also remove this successor node (Algorithm 2). This allows us in the next steps to avoid checking nodes that are irrelevant to test satisfiability. In the case of elimination of a prestate, we also eliminate the predecessor states, since the condition that a state must have all its successors is not fulfilled. As it may take time between the moment we decide to delete a node and when we effectively delete it, we use a hashtable to store all the eliminated node. In this way, we can refer to this hashtable to know whether a node has to be considered removed or not.

In Algorithm 1 (line 13), we need to check whether eventualities are realized or not. This part of the implementation differs between ATL^+ and ATL^*. For ATL^+, we keep track of the evolution of the search for realization of a given eventuality with the three following status:

- **realized**: the eventuality is realized at that node.
- **in treatment**: we don't know whether the eventuality can be realized or not from that node, and we try to realize it.
- **not realized**: the eventuality cannot not be realized.

For a given eventuality, when we go through a node which is declared in treatment for the second time, that is the status of the node is *in treatment*, this means that this eventuality cannot be realized from this state. The state has therefore to be removed from the graph. This is operational only for ATL^+. For ATL^*, the treatment is more subtle. When a prestate has already been visited, we need to check whether it is possible to continue the play by choosing another successor state. In that purpose, we keep a list of successor states that have not yet been visited from this prestate for a given path.

Algorithm 1: state and prestate elimination

```
1  foreach vertex v in the tableau do
2      if v is a prestate then
3          if v has no successors then
4              remove v
5          end
6      else
           // v is a state
7          make a set with all the action vectors of the outgoing edges of v;
8          compare this set with the set of move actions in v;
9          if the two sets are different then
               // the state has at least one missing action vectors
10             remove v
11         else
12             get all the eventualities that are not immediately realized in v;
13             if one of these eventualities is not realized in successor states then
14                 remove the state
15             end
16         end
17     end
18 end
```

5.5 Test of the Implementation

As the main difference between ATL and ATL* comes from path formulae, we mainly focus our tests on that point. Therefore, we use and adapt the list of tests proposed by Reynolds for CTL*[7]. This allows us to check that our application gives the same results in term of satisfiability and that the running times we obtain for these examples are satisfactory. Moreover, other tests using formulae with non trivial coalitions have been done. The result of these tests are given in Table 5.1.

Others ATL* formulae have been tried, and it appears that the execution time blows up when $\Box$, $\Diamond$ and $\vee$ are combined and nested in the same formula. This seems normal, since it corresponds to cases where a lot of possible futures are generated by the function dec*. For example, we have not a reasonable time to obtain the result for the formula $\langle\langle 1\rangle\rangle(\Box\Diamond p \vee \Box\Diamond q \vee \Box\Diamond r)$.

[7] http://www.csse.uwa.edu.au/~mark/research/Online/quicktab/quicktablong.pdf

Algorithm 2: node removal

```
1  Function remove_state(v)
2      if v is not registered as deleted then
3          register that v is deleted; remove every incoming edge of v; foreach successor
           prestate w of v do
4              if w has only one predecessor state // (that is v)
5              then
6                  remove_prestate(w);
7              end
8          end
9          remove v;
10     end
11 Function remove_prestate(v)
12 if v is not registered as deleted then
13     register that v is deleted; foreach predecessor state u of v do
14         remove_state(u)
15     end
16     foreach successor prestate w of v do
17         if w has only one predecessor state // (that is v)
18         then
19             remove_state(w);
20         end
21     end
22     remove v;
23 end
```

	Sat?	ATL* time(ms)	CTL* time(ms)		Sat?	ATL* time(ms)	CTL* time(ms)
θ_1	yes	7	22	$\neg\theta_1$	no	0	58
θ_2	yes	0	7	$\neg\theta_2$	no	0	14
θ_3	yes	156	15	$\neg\theta_3$	no	7	31
θ_4	yes	7	1	$\neg\theta_4$	no	0	13
θ_5	yes	0	0	$\neg\theta_5$	no	0	3
θ_6	yes	0	1	$\neg\theta_6$	no	0	3
θ_7	yes	0	1	$\neg\theta_7$	no	5	2
θ_8	yes	0	0	$\neg\theta_8$	no	0	1
θ_9	yes	46	2	$\neg\theta_9$	no	6	82
θ_{10}	yes	7	3	$\neg\theta_{10}$	no	50	12
θ_{11}	yes	4660	6	$\neg\theta_{11}$	no	24	40003
θ_{12}	yes	0	2	$\neg\theta_{12}$	no	4	671
θ_{13}	yes	0	8	$\neg\theta_{13}$	no	39	2351
θ_{14}	yes	2	7	$\neg\theta_{14}$	no	39	40003
θ_{15}	yes	0	0	$\neg\theta_{15}$	yes	0	0
θ_{16}	yes	0	1	$\neg\theta_{16}$	yes	0	0
θ_{17}	yes	23	114	$\neg\theta_{17}$	yes	17	16
θ_{18}	yes	29	309	$\neg\theta_{18}$	yes	11	5
θ_{19}	no	7	343	$\neg\theta_{19}$	yes	0	3
θ_{20}	yes	31	68	$\neg\theta_{20}$	yes	3	29

$\theta_1 = \langle\langle 1\rangle\rangle(\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q))$

$\theta_2 = \langle\langle 1\rangle\rangle(\Box p \rightarrow (p \wedge \bigcirc p \wedge \bigcirc\Box p))$

$\theta_3 = \langle\langle 1\rangle\rangle((p \mathsf{U} q) \leftrightarrow (q \vee (p \wedge \bigcirc(p \mathsf{U} q))))$

$\theta_4 = \langle\langle 1\rangle\rangle((p \mathsf{U} q) \rightarrow \Diamond q)$

$\theta_5 = \langle\langle 1\rangle\rangle(p \rightarrow \langle\langle\rangle\rangle(\langle\langle 1\rangle\rangle p))$

$\theta_{10} = \langle\langle 1\rangle\rangle((\langle\langle\rangle\rangle(\Box(p \rightarrow (q \mathsf{U} r))) \wedge (q \mathsf{U} p)) \rightarrow (q \mathsf{U} r))$

$\theta_{11} = \langle\langle 1\rangle\rangle(\Box(\langle\langle 1\rangle\rangle\Diamond p \rightarrow \bigcirc\Diamond(\langle\langle 1\rangle\rangle\Diamond p)) \rightarrow (\langle\langle 1\rangle\rangle\Diamond p \rightarrow \Box\Diamond(\langle\langle 1\rangle\rangle\Diamond p)))$

$\theta_{12} = \langle\langle\rangle\rangle\Box(p \rightarrow \langle\langle 1\rangle\rangle\bigcirc p) \rightarrow (p \rightarrow \langle\langle 1\rangle\rangle\Box p)$

$\theta_6 = \langle\langle\rangle\rangle p \rightarrow \langle\langle\rangle\rangle(\langle\langle\rangle\rangle p)$

$\theta_7 = \langle\langle 1\rangle\rangle(\langle\langle\rangle\rangle\bigcirc p \rightarrow \bigcirc(\langle\langle\rangle\rangle p))$

$\theta_8 = \langle\langle 1\rangle\rangle(p \rightarrow \langle\langle\rangle\rangle p)$

$\theta_9 = \langle\langle 1\rangle\rangle(p \mathsf{U}(\langle\langle 1\rangle\rangle(p \mathsf{U} q))) \rightarrow \langle\langle 1\rangle\rangle(p \mathsf{U} q)$

$\theta_{13} = \langle\langle\rangle\rangle\Box(\langle\langle 1\rangle\rangle p \rightarrow \langle\langle 1\rangle\rangle\bigcirc((\langle\langle 1\rangle\rangle q)\mathsf{U}(\langle\langle 1\rangle\rangle p))) \rightarrow (\langle\langle 1\rangle\rangle p -> \langle\langle 1\rangle\rangle\Box((\langle\langle 1\rangle\rangle q)\mathsf{U}(\langle\langle 1\rangle\rangle p)))$

$\theta_{14} = (\langle\langle\rangle\rangle\Box(p \rightarrow \langle\langle 1\rangle\rangle\bigcirc r) \wedge \langle\langle\rangle\rangle\Box(r \rightarrow \langle\langle 1\rangle\rangle\bigcirc p)) \rightarrow (p \rightarrow \langle\langle 1\rangle\rangle\Box(\Diamond p \wedge \Diamond r))$

$\theta_{15} = p$

$\theta_{16} = \langle\langle 1\rangle\rangle(p \wedge \bigcirc p \wedge \Diamond\neg p)$

$\theta_{17} = \langle\langle\rangle\rangle\Box((p \wedge \bigcirc\neg p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge \bigcirc p \wedge q \wedge \neg r) \vee (\neg p \wedge \bigcirc p \wedge \neg q \wedge r)) \wedge \langle\langle 1\rangle\rangle(\Diamond q \wedge \Diamond r)$

$\theta_{18} = \langle\langle\rangle\rangle\Box(\langle\langle 1\rangle\rangle\bigcirc p \wedge \langle\langle 1\rangle\rangle\bigcirc\neg p) \wedge \langle\langle\rangle\rangle\Box(\Box p \vee ((\neg r)\mathsf{U}(r \wedge \neg p)))$

$\theta_{19} = \neg(\langle\langle\rangle\rangle\Diamond\langle\langle\rangle\rangle\Box q \rightarrow \langle\langle\rangle\rangle\Diamond\Box q)$

$\theta_{20} = \langle\langle\rangle\rangle\Box(p \leftrightarrow \bigcirc\neg p) \wedge \langle\langle\rangle\rangle\Box(p \rightarrow \neg q) \wedge \langle\langle\rangle\rangle\Box(p \rightarrow \neg r) \wedge \langle\langle\rangle\rangle\Box(q \rightarrow \neg r) \wedge \langle\langle\rangle\rangle\Box(\Diamond q \wedge \Diamond r) \wedge q$

Table 5.1: Comparison of TATL with the CTL* reasoner of M. Reynolds

CHAPTER 6

CONCLUSION & PERSPECTIVES

In this thesis, we have provided the first two tableau-based decision procedures for ATL^+ and for ATL^*. These procedures are sound, complete and optimal. They both run in 2EXPTIME, which corresponds to the complexity of the satisfiability problem for ATL^+ and ATL^*. We have also provided an implementation for ATL^* in Ocaml, which is available as a web application and a command line application. Up to our knowledge, it is the first running tool to decide satisfiability of ATL formulae, ATL^+ formulae and ATL^* formulae. This implementation is still a prototype and improvement can be done to decrease execution time of the procedure.

In the introduction, we have said that one of the goal of this thesis was to provide tools to design safe systems, and one way to do it was to directly generate a model from a given specification. Due to lack of time, this part of this thesis has just begun and in the following, we outline some ongoing works as perspectives.

6.1 Model Extraction

All the tableau-based decision procedures presented in this thesis (including the already existing one for ATL) are constructive. A method to extract a model from an open tableau of an input formula is given in each completeness proof (for ATL [31], for ATL^+ [15] and for ATL^* Appendix B.3). The methods of extraction are almost the same for the three versions of ATL. Nevertheless, these methods of extraction give very big models that are not easy to read by a human person in most cases. Moreover, this model could be used to model check additional properties, for instance inferred properties, and the smaller is the model, the better model checking works. The method extracts a Hintikka structure that can be easily transformed into a model. A Hintikka structure is a tree-like graph where nodes are labelled by states of the corresponding open tableau. A

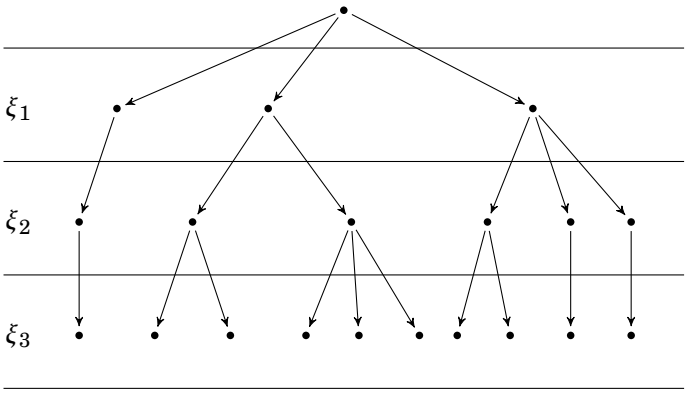


Figure 6.1: Construction of a Hintikka structure eventualities by eventualities

Hintikka structure for a given formula θ has properties (specific to each logic) that ensures to satisfy θ, and in particular, the realization of the eventualities contained in θ. So, in order to satisfy all the eventualities, we construct the Hintikka structure level by level, each level corresponding to an eventuality, as illustrated in Figure 6.1. In the case of ATL*, it may be necessary to treat the same eventuality several times, that is on several different levels.

Once we are sure that all eventualities are fulfilled, it is possible to close the so obtained tree-like structure by looping on the adequate nodes in order to obtain a graph. This graph is then transformed into a CGM by keeping in the label of each node only the propositions that are true at that node.

But, it often happens that several eventualities are realized simultaneously, and this should mean that it is not necessary to treat them on next levels.

Hintikka structures are constructed in this way to be sure that if an eventuality needs to realize two different objectives with two different branches of a prestate, then both branches can be selected one after the other. This is the case in Example 4.10, where in order to realize the eventuality $\xi = \langle\!\langle 1 \rangle\!\rangle \Box((p \wedge \bigcirc \neg p) \vee (\neg p \wedge \bigcirc p))$ we need to choose first the branch starting from Γ_3, and leading to a state Δ_8 where r is true, and then when we come back to Γ_3 choose the branch leading to Δ_9 where q is true (or vice-versa). But, we think that this situation can only happen with ATL* where Boolean combination *and* nesting of temporal operators are simultaneously allowed. For ATL and ATL$^+$, some work already done seems to support the conjecture that this situation cannot happen.

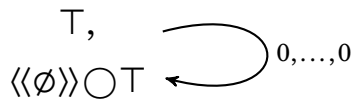
So to obtain smaller models, we plan to proceed differently with ATL* on one side, and with ATL and ATL$^+$ on the other side. Let us start with ATL*.

6.1.1 Smaller Models for ATL*

For ATL*, we have seen above that it is necessary to construct a Hintikka structure. We then need to find a way to reduce the number of nodes of the CGM obtained from the Hintikka structure.

Here the idea is to use coarsest partition refinement [37, 46] and bisimulation [1] that are already used to reduce transition system and Kripke structure for CTL* [10, 16] .

In that purpose, an ongoing work with two master students Eloïse Billa and Adrien Cotte,and Serenella Cerrito gives some results on how to adapt the Kannelakis & Smolka procedure or the Page & Tarjan procedure for CGM. Then, we will have to prove that a given model and its refined models are bisimilar. We hope that this will give us a solution that preserves satisfiability of all formulae in all the nodes of a CGM. But, what we really want is to preserve satisfiability of the initial formula. So, it seems that some heuristics can be found to reduce the numbers of nodes in a given Hintikka structure while preserving the truth of the initial formula, and therefore leading to a smaller model. For example, a lot of open tableaux contains the following state:



which intuitively means that, from this point, any proposition can be true (or false). Therefore this state can be removed and its ingoing edges can be redirected to their source state, as exemplify in Figure 6.2.

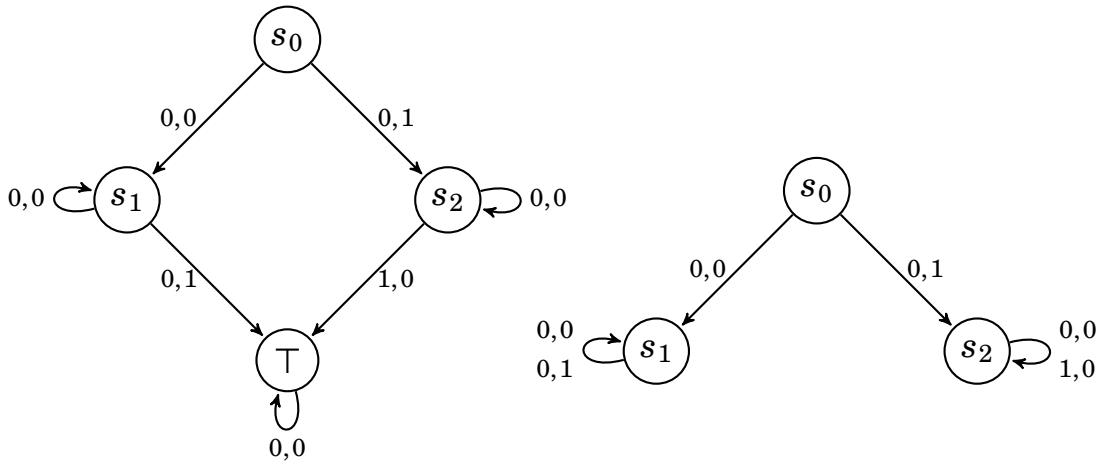


Figure 6.2: Suppression of the node ⊤ from a model. We can obtain the model on the right from the one on the left.

6.1.2 Smaller Models for ATL and ATL$^+$

Here the idea is to benefit from the fact that all eventualities can be realized by choosing only one option from each prestate. Therefore, it seems that by making the "good choice" at each non-deterministic fork starting from a prestate, we can directly extract a CGM for the open tableau without constructing the Hintikka structure. This will ensure us to obtain a CGM for a

given formula θ of at most the size of the open tableau for θ. This is a ongoing work with Fabio Papacchini from the university of Manchester.

Of course, after, methods to reduce the number of nodes can also be applied to reduce the model we have obtained, but in most cases, we will start with a smaller structure than the one extracted via Hintikka structures.

6.2 Comparison of Methods for Deciding Satisfiability of ATL* Formulae

Another ongoing work concerns the tableau-based decision procedure for ATL* itself. Our procedure is not the only one for deciding satisfiability of ATL* formulae. There also exists an automata-based decision procedure proposed by S. Schewe [61] to show that the complexity of the satisfiability problem for ATL* was 2EXPTIME-complete.

Therefore, we plan to make both a theoretical and a practical comparison between the tableau-based and the automata-based decision procedures for ATL*. In order to make a theoretical comparison of both tools, we will need to first implement the automata-based procedure, and then propose a benchmark for ATL*, if we want to be really informative. This will be a joint work with Sven Schewe from the University of Liverpool.

Appendices

APPENDIX A

ADDITIONAL DEFINITIONS FOR PROOFS

A.1 Actions and Outcomes

First, we recall that, for a given state s in a CGS, $\mathsf{act}_A(s)$ denotes the set of all A-actions that can be played by the coalition A at state s, i.e. $\mathsf{act}_A(s) = \prod_{\mathsf{a}\in A} \mathsf{act}_\mathsf{a}(s)$. Let $\sigma_A \in \mathsf{act}_A(s)$. We say that an action vector $\sigma_\mathbb{A}$ extends an A-action σ_A, denoted by $\sigma_\mathbb{A} \sqsupseteq \sigma_A$, if $\sigma_\mathbb{A}(\mathsf{a}) = \sigma_A(\mathsf{a})$ for every $\mathsf{a} \in A$.

We use $\mathsf{act^c}_A(s)$ to denote the set of all A-co-actions available at state s and σ^c_A for an element of this set. We also use $\mathsf{Out}(s, \sigma_A)$ to denote the set of all states s' for which there exists an action vector $\sigma_\mathbb{A} \in \mathsf{act}_\mathbb{A}(s)$ that extends σ_A and such that $\mathsf{out}(s, \sigma_\mathbb{A}) = s'$. We define in a same way $\sqsupseteq$ and $\mathsf{Out}(s, \sigma^c_A)$ for an A-co-action $\sigma^c_A \in \mathsf{act^c}_A(s)$.

A.2 Trees

In the following definitions and proofs, we make use of the notion of *tree*. In our context, we use this term as a synonym of "directed, connected, and acyclic graph, each node of which, except one, the root, has exactly one incoming edge". We note a tree as a pair $(R, \rightarrow)$, where R is the set of nodes and $\rightarrow$ is the parent-child relation (the edges).

Given sets X, Y, Z and mappings $c : X \rightarrow Y$ and $d : X \rightarrow Y \times Z$, we sometimes say that the set X is Y-coloured by c and that for any $x \in X$, the value $c(x)$ is the Y-colour of x under the colouring c. Moreover, we sometimes say and that the set X is Y-Z-coloured by d, that for any $x \in X$, the value $d(x)$ is the X-Y-colour of x under the colouring d.

Definition A.1. Let $\mathscr{R} = (R, \rightarrow)$ be a tree and X be a non-empty set. An X-colouring of $\mathscr{R}$ is a mapping $c : R \mapsto X$. When such mapping is fixed, we say that $\mathscr{R}$ is X-coloured. Moreover, let Y

be a non-empty set. An X-Y-colouring of $\mathscr{R}$ is a mapping $d : R \mapsto X \times Y$. When such mapping is fixed, we say that $\mathscr{R}$ is X-Y-coloured.

A.3 Additional Definitions for Tableaux

A.3.1 States and Prestates

We denote by prestates(Δ) the set of prestate successors of a state Δ, and by states(Γ) the set of state successors of a prestate Γ. We recall that $\mathscr{T}^\theta_n$, S^θ_n, $\mathscr{T}^\theta$ and S^θ correspond to the n-th intermediate tableau for the formula θ during the construction phase, the set of nodes of the intermediate tableau $\mathscr{T}^\theta_n$, the final tableau for the formula θ and the set of nodes of the final tableau $\mathscr{T}^\theta_n$, respectively.

A.3.2 Outcomes

Let $A \subseteq \mathbb{A}$ be a coalition and Δ be a state in a tableau. We denote by $D_A(\Delta)$ the set $d(\Delta)^{|A|}$ and by $D^c_A(\Delta)$ the set $d(\Delta)^{|\mathbb{A}-A|}$. See Section 4.3 for definition of $d(\Delta)$. Let $\sigma_A \in D_A(\Delta)$. We say that $\sigma_{\mathbb{A}} \in D(\Delta)$ extends σ_A, denoted by $\sigma_{\mathbb{A}} \sqsupseteq \sigma_A$, if $\sigma_{\mathbb{A}}(\mathsf{a}) = \sigma_A(\mathsf{a})$ for every $\mathsf{a} \in A$. We define in the same way $\sqsupseteq$ for $\sigma^c_A \in D^c_A(\Delta)$.

Definition A.2 (Outcome set of $\sigma_A \in \Delta$)**.** Let $\Delta \in S^\theta_n$ be a state and $\sigma_A \in D_A(\Delta)$. An outcome set of σ_A at Δ is a minimal set of states $X \subseteq S^\theta_n$ such that for every $\sigma_{\mathbb{A}} \sqsupseteq \sigma_A$ there exists exactly one state $\Delta' \in X$ such that $\Delta \xrightarrow{\sigma_{\mathbb{A}}} \Gamma \Longrightarrow \Delta'$.

Definition A.3 (Outcome set of σ^c_A at Δ)**.** Let $\Delta \in S^\theta_n$ be a state and $\sigma^c_A \in D^c_A(\Delta)$. An *outcome set of σ^c_A at Δ* is a minimal set of states X such that for every $\sigma_A \in D^c_A(\Delta)$, there exists exactly one $\Delta' \in X$ such that $\Delta \xrightarrow{\sigma^c_A(\sigma_A)} \Gamma \Longrightarrow \Delta'$.

Some notation. Let $\Delta \in S^\theta_n$ be a state.

1. Whenever we write $\langle\langle A_p\rangle\rangle\bigcirc\varphi_p \in \Theta$, we mean that $\langle\langle A_p\rangle\rangle\bigcirc\varphi_p$ is the p-th successor formula of the form $\langle\langle A\rangle\rangle\bigcirc\varphi$ according to the ordering of successor formulae induced by the application of rule **Next** to Δ.
   We use the notation $[\![A'_q]\!]\bigcirc\psi_q \in \Theta$ likewise.
2. Given $\langle\langle A_p\rangle\rangle\bigcirc\varphi_p \in \Theta$, we denote by $\sigma_{A_p}[\langle\langle A_p\rangle\rangle\bigcirc\varphi_p]$ the unique tuple σ_{A_p} enforcing φ_p such that $\sigma_{A_p}(\mathsf{a}) = p$ for every $\mathsf{a} \in A_p$.
3. Likewise, given a formula $[\![A'_q]\!]\bigcirc\psi \in \Theta$,
   - if $A'_q \neq \mathbb{A}$ then we denote by $\sigma^c_{A'_q}[[\![A'_q]\!]\bigcirc\psi]$ the unique A'_q-co-action $\sigma^c_{A'_q}$ enforcing ψ such that $\mathrm{co}(\sigma^c_{A'_q}(\sigma_{A'_q})) = q$ and $\mathbb{A} - A'_q \subseteq N(\sigma^c_{A'_q}(\sigma_{A'_q}))$;
   - if $A'_q = \mathbb{A}$ then we denote by $\sigma^c_{A'_q}[[\![A'_q]\!]\bigcirc\psi]$ the unique A'_q-co-action $\sigma^c_{A'_q}$ enforcing ψ, that is by definition the identity function.

A.3.3 Realization Witness Tree for Tableaux

Intuitively, a *realization witness tree* for a tableau is a tree that witnesses the satisfaction of a given potential eventuality ξ at a state and simulates a tree of runs in a tableau.

To define realization witness trees, we use the notion of descendant potential eventuality of degree d and its associate notation as seen in Definition 4.2. We recall that, given a potential eventuality $\xi = \langle\langle A\rangle\rangle\Phi$ ($[\![A]\!]\Phi$), by convention ξ itself is taken to be its (unique) descendant potential eventuality of degree 0 and that if ξ^i is a descendant eventuality of degree i of ξ then a γ-component of ξ^i will have the form $\psi \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle \Phi^{i+1}$ (respectively, $\psi \wedge [\![A]\!] \bigcirc [\![A]\!]\Phi^{i+1}$) and $\langle\langle A\rangle\rangle\Phi^{i+1}$ (respectively, $[\![A]\!]\Phi^{i+1}$) will be a descendant potential eventuality of ξ having degree $d = i+1$.

Definition A.4 (Realization Witness Trees for Tableaux)**.** A realization witness tree for a potential eventuality (a) $\xi = \langle\langle A\rangle\rangle\Phi$ or (b) $\xi = [\![A]\!]\Phi$ at state $\Delta \in S_n^\theta$ is a finite S_n^θ-$\mathsf{ATL}^*_\mathsf{p}$-coloured tree $\mathscr{R} = (R, \rightarrow)$ such that:

1. the root of $\mathscr{R}$ is coloured with Δ and Φ, and is of depth 0;
2. if an interior node w of depth i of $\mathscr{R}$ is coloured with Δ' and Φ', then there exists a successor ξ^i such that $\xi^i \in \Delta'$ and there exists ξ^{i+1} of ξ^i such that (a) $\langle\langle A\rangle\rangle \bigcirc \xi^{i+1} \in \Delta'$ or (b) $[\![A]\!] \bigcirc \xi^{i+1} \in \Delta'$;
3. for every interior node $w \in \mathscr{R}$ of depth i coloured with Δ' and Φ', the children of w are coloured bijectively with vertices from an outcome set of (a) $\sigma_A[\langle\langle A\rangle\rangle \bigcirc \xi^{i+1}]$ or (b) $\sigma^c_{A'_q}[[\![A'_q]\!] \bigcirc \psi]$, and by Φ'', where for each children w' of w so coloured by Δ'' and Φ'', $\xi^{i+1} = \langle\langle A\rangle\rangle\Phi^{i+1}$ or $\xi^{i+1} = [\![A]\!]\Phi^{i+1}$ respectively, and $\Phi'' = \mathsf{WF}(\Phi', \Delta'', \gamma_{sl}(\xi^{i+1}, \Delta''))$;
4. if a leaf of depth i of $\mathscr{R}$ is coloured with Δ' and Φ', then (a) $\xi^i = \langle\langle A\rangle\rangle\Phi_i \in \Delta'$ or (b) $\xi = [\![A]\!]\Phi_i \in \Delta'$ is such that $\mathsf{WF}(\Phi', \Delta', \gamma_{sl}(\xi_i, \Delta')) = \top$.

A.4 Hintikka Structures

We define the notion of *concurrent game Hintikka structure*, in short CGHS, in two steps. First, we define a structure, that we call *general Hintikka structure* without constraints on the labelling function. Then, after having defined the notion of *realization witness tree* for this structure, we give the full definition of CGHS. Moreover we give the definition of a CGHS for a given ATL^* formula θ and set that a CGM for θ can be extracted from this structure.

Definition A.5 (General Hintikka Structure)**.** A *general Hintikka structure* is a tuple $\mathscr{H} = (\mathbb{A}, \mathbb{S}, \{\mathsf{Act}_\mathsf{a}\}_{\mathsf{a}\in\mathbb{A}}, \{\mathsf{act}_\mathsf{a}\}_{\mathsf{a}\in\mathbb{A}}, \mathsf{out}, H)$ where $(\mathbb{A}, \mathbb{S}, \{\mathsf{Act}_\mathsf{a}\}_{\mathsf{a}\in\mathbb{A}}, \{\mathsf{act}_\mathsf{a}\}_{\mathsf{a}\in\mathbb{A}}, \mathsf{out})$ is a CGS and H is a labelling function $H : \mathbb{S} \mapsto \mathscr{P}(\Gamma)$, where Γ is a set of ATL^* formulae.

A.4.1 Realization Witness Tree for General Hintikka Structure

Here, we adapt the notation introduced for tableaux to general Hintikka structures. Consider a general Hintikka structure $\mathscr{H}$ and a state s such that $H(s) = \Theta$ and suppose that the elements of Θ are listed by the enumeration E defined in the rule Next where successor formulae of the form $\langle\langle A\rangle\rangle\bigcirc\varphi$ appear before successor formulae of the form $[\![A']\!]\bigcirc\varphi$, where $A' \neq \mathbb{A}$, and formulae of the form $[\![\mathbb{A}]\!]\bigcirc\varphi$ are at the end of the list.

1. Whenever we write $\langle\langle A_p\rangle\rangle\bigcirc\varphi_p \in \Theta$, we mean that $\langle\langle A_p\rangle\rangle\bigcirc\varphi_p$ is the p-th successor formula of the form $\langle\langle A\rangle\rangle\bigcirc\varphi$ according to the enumeration E.
   We use the notation $[\![A'_q]\!]\bigcirc\psi_q \in \Theta$ likewise.
2. Given $\langle\langle A_p\rangle\rangle\bigcirc\varphi_p \in \Theta$, we denote by $\sigma_{A_p}[\langle\langle A_p\rangle\rangle\bigcirc\varphi_p]$ the unique A_p-action σ_{A_p} enforcing φ_p such that $\sigma_{A_p}(\mathsf{a}) = p$ for every $\mathsf{a} \in A_p$
3. Likewise, given a formula $[\![A'_q]\!]\bigcirc\psi_q \in \Theta$,
   - if $A'_q \neq \mathbb{A}$ then we denote by $\sigma^c_{A'_q}[[\![A'_q]\!]\bigcirc\psi_q]$ the unique A'_q-co-action $\sigma^c_{A'_q}$ enforcing ψ_q such that $\mathsf{co}(\sigma^c_{A'_q}(\sigma_{A'_q})) = q$ and $\mathbb{A} - A'_q \subseteq N(\sigma^c_{A'_q}(\sigma_{A'_q}))$;
   - if $A'_q = \mathbb{A}$ then we denote by $\sigma^c_{A'_q}[[\![A'_q]\!]\bigcirc\psi_q]$ the unique A'_q-co-action $\sigma^c_{A'_q}$ enforcing ψ_q, that is, by definition, the identity function.

Realization witness trees for general Hintikka structure have the same objective as the ones for tableaux. However, the definition is slightly different since structures are different.

Definition A.6 (Realization Witness Trees for General Hintikka Structure)**.** Let s be a state of a general Hintikka structure. A realization witness tree for a potential eventuality (a) $\xi = \langle\langle A\rangle\rangle\Phi$ or (b) $\xi = [\![A]\!]\Phi$ at state $\Delta \in S^\theta_n$ is a finite $\mathbb{S}$-$\mathsf{ATL}^*_\mathsf{p}$-coloured tree $\mathscr{R} = (R, \rightarrow)$ such that:

1. the root of $\mathscr{R}$ is coloured with s and Φ, and is of depth 0;
2. if an interior node w of depth i of $\mathscr{R}$ is coloured with s' where $H(s') = \Theta$ and Φ', then there exists a successor ξ^i such that $\xi^i \in \Theta$ and there exists ξ^{i+1} of ξ^i such that the set (a) $\langle\langle A\rangle\rangle\bigcirc\xi^{i+1} \in \Theta$ or (b) $[\![A]\!]\bigcirc\xi^{i+1} \in \Theta$;
3. for every interior node $w \in \mathscr{R}$ of depth i coloured with s' and Φ', the children of w are coloured bijectively with vertices from (a) $\mathsf{Out}(s', \sigma_A[\langle\langle A\rangle\rangle\bigcirc\xi^{i+1}])$ or (b) the set $\mathsf{Out}(s', \sigma^c_{A'_q}[[\![A'_q]\!]\bigcirc\psi])$,and by Φ'', where for each children w' of w so coloured by Δ'' and Φ'', $\xi^{i+1} = \langle\langle A\rangle\rangle\Phi^{i+1}$ or $\xi^{i+1} = [\![A]\!]\Phi^{i+1}$ respectively, and $\Phi'' = \mathsf{WF}(\Phi', \Delta'', \gamma_{sl}(\xi^{i+1}, \Delta''))$;
4. if a leaf of depth i of $\mathscr{R}$ is coloured with s' where $H(s') = \Theta$ and Φ', then (a) $\xi^i = \langle\langle A\rangle\rangle\Phi_i \in \Theta$ or (b) $\xi = [\![A]\!]\Phi_i \in \Theta$ is such that $\mathsf{WF}(\Phi', \Theta, \gamma_{sl}(\xi_i, \Theta)) = \top$.

A.4.2 Concurrent Game Hintikka Structure

When defining *concurrent game Hintikka structure*, we aim at giving constraints on the labelling function H.

Definition A.7 (Concurrent Game Hintikka Structure)**.** A *concurrent game Hintikka structure* (for short, CGHS) is a general Hintikka structure $\mathscr{H} = (\mathbb{A}, \mathbb{S}, \{\mathsf{Act}_a\}_{a\in\mathbb{A}}, \{\mathsf{act}_a\}_{a\in\mathbb{A}}, \mathsf{out}, H)$ where the labelling function H satisfies the following constraints:

H1 If $\neg p \in H(s)$ then $p \notin H(s)$ for all $p \in \mathbb{P}$;

H2 If an α-formula belongs to $H(s)$, then its both α-components do;

H3 If a β-formula belongs to $H(s)$, then one of its β-components does;

H4 If a γ-formula belongs to $H(s)$, then one of its γ-components does;

H5 If $\langle\langle A\rangle\rangle \bigcirc \psi \in H(s)$, then there exists an A-action $\sigma_A \in \mathsf{act}_A s$ such that $\psi \in H(s')$ for all $s' \in \mathsf{Out}(s, \sigma_A)$. Likewise, if $[\![A]\!] \bigcirc \psi \in H(s)$, then there exists an A-co-action $\sigma^c_A \in \mathsf{act}^c{}_A(s)$ such that $\psi \in H(s')$ for all $s' \in \mathsf{Out}(s, \sigma^c_A)$.

H6 If a potential eventuality $\xi = \langle\langle A\rangle\rangle \Phi$ (resp. $\xi = [\![A]\!]\Phi$) belongs to $H(s)$, then there exists a realization witness tree, rooted at s in $\mathscr{H}$ for $\xi = \langle\langle A\rangle\rangle\Phi$ (resp. $\xi = [\![A]\!]\Phi$) at s.

Definition A.8. Let $\mathscr{H} = (\mathbb{A}, \mathbb{S}, \{\mathsf{Act}_a\}_{a\in\mathbb{A}}, \{\mathsf{act}_a\}_{a\in\mathbb{A}}, \mathsf{out}, H)$ be a CGHS and θ be an ATL^*-formula. We say that $\mathscr{H}$ is a concurrent game Hintikka structure for θ, if $\theta \in H(s)$ for some $s \in \mathbb{S}$.

The next theorem sets that from any CGHS for a given formula θ a CGM satisfying θ can be obtained.

Theorem A.1. *Let $\mathscr{H} = (\mathbb{A}, \mathbb{S}, \{\mathsf{Act}_a\}_{a\in\mathbb{A}}, \{\mathsf{act}_a\}_{a\in\mathbb{A}}, \mathsf{out}, H)$ be a CGHS for a given ATL^*-formula θ. Let further $\mathscr{M} = (\mathbb{A}, \mathbb{S}, \{\mathsf{Act}_a\}_{a\in\mathbb{A}}, \{\mathsf{act}_a\}_{a\in\mathbb{A}}, \mathsf{out}, \mathbb{P}, L)$ be the CGM obtained from $\mathscr{H}$ by setting, for every $s \in \mathbb{S}$, $L(s) = H(s) \cap \mathbb{P}$. Then, for every $s \in \mathbb{S}$ and every ATL^* formula φ, $\varphi \in H(s)$ implies $\mathscr{M}, s \models \varphi$. In particular, $\mathscr{M}$ satisfies θ.*

The proof of this theorem can be found in appendix B.1.

APPENDIX B

PROOFS

Here, we present proofs for our ATL* tableau-based decision procedure. The proofs for our ATL+ tableau-based decision procedure are relatively similar and can be found in [14].

B.1 Proof of Theorem A.1

Proof. Suppose $\varphi \in H(s)$. We will prove that $\mathcal{M}, s \models \varphi$ by induction on the structure of the state formula φ.

Base. If $\varphi = p$ or $\varphi = \neg p$, with $p \in \mathbb{P}$, it is immediate that $\mathcal{M}, s \models \varphi$, by definition of L and H1.

Inductive Step.

- φ is $\psi_1 \wedge \psi_2$. By H2 we get that $\psi_1 \in H(s)$ and $\psi_2 \in H(s)$. By inductive hypothesis $\mathcal{M}, s \models \psi_1$ and $\mathcal{M}, s \models \psi_2$. Therefore $\mathcal{M}, s \models \varphi$.
- φ is $\psi_1 \vee \psi_2$. By H3 we get that either $\psi_1 \in H(s)$ or $\psi_2 \in H(s)$. By inductive hypothesis either $\mathcal{M}, s \models \psi_1$ or $\mathcal{M}, s \models \psi_2$. Therefore $\mathcal{M}, s \models \varphi$.
- φ is $\langle\langle A \rangle\rangle \bigcirc \psi$ or $[\![A]\!] \bigcirc \psi$. An application of H5 and the inductive hypothesis to ψ imply that $\mathcal{M}, s \models \varphi$.
- φ is $\langle\langle A \rangle\rangle \Phi$ or φ is $[\![A]\!]\Phi$, where Φ is not of the form $\bigcirc\varphi$, that is φ is a γ-formula. Here we only present in detail the first case, the second one being similar. We need to prove the existence of a (perfect-recall) strategy F_A such that, for each branch λ in $\mathcal{M}$ stemming from s and consistent with that strategy, $\mathcal{M}, \lambda \models \Phi$. This will imply that $\mathcal{M}, s \models \varphi$. Below, we show how to construct the strategy F_A recursively on the Hintikka structure, proceeding step-by-step, possibly an infinite number of times, starting from s and $\xi = \langle\langle A \rangle\rangle \Phi \in H(s)$.

  The step for the construction of F_A: the step is applied to a given node $s' \in \mathcal{H}$ and a γ-formula $\xi' = \langle\langle A \rangle\rangle \Psi \in H(s')$. Since ξ' belongs to $H(s')$, then H6 guarantees the existence of

a realization witness tree T rooted at s' in $\mathscr{H}$ for ξ'. By construction, T provides a partial finite strategy F_{P_A}. Next, let us consider any path in T of the form $\lambda_{\leq n}$, where $\lambda_0 = s'$ and λ_n is a leaf. By construction of T, each node λ_i, for $0 \leq i \leq n$, is a node of $\mathscr{H}$ and each labelled edge of T is a labelled edge of $\mathscr{H}$. The descendant potential eventuality $\xi^n = \langle\langle A\rangle\rangle \Psi^n$ of ξ' belongs to the first part of the colour of λ_n by construction of T. Since λ_n is a node of $\mathscr{H}$ and $\xi^n \in H(\lambda_n)$, by H4 some γ-component χ of ξ^n belongs to $H(\lambda_n)$. This formula χ is either of the form ψ or of the form $\psi \wedge \langle\langle A\rangle\rangle \bigcirc \xi^{n+1}$ (the second case occurs, for instance, when ξ has the form $\langle\langle A\rangle\rangle \Box\Diamond\theta$).

- In the first case, any extension of the partial strategy F_{P_A} and any extension of $\lambda_{\leq n}$ to an infinite path will do.
- In the second case, we compute $\Psi^{n+1} = \mathsf{WF}(\Psi^n, H(\lambda_n), \gamma_{sl}(\xi^n, H(\lambda_n)))$.
  - If $\Psi^{n+1} \neq \top$, then we apply the step for the construction for F_A to λ_n and $\xi^n \in H(\lambda_n)$
  - else, we apply H2 to get $\psi \in H(\lambda_n)$ and $\langle\langle A\rangle\rangle \bigcirc \xi^{n+1} \in H(\lambda_n)$. By H5, there exists an A-action $\sigma_A \in \mathsf{act}_A(H(\lambda_n))$ such that $\xi^{n+1} \in H(s')$ for all $s' \in \mathsf{Out}(\lambda_n, \sigma_A)$. Playing this A-action σ_A after the partial strategy F_{P_A} gives us a new partial strategy F'_{P_A}. The set of successors of λ_n for T' is the set $\mathsf{Out}(\lambda_n, \sigma_A)$. For any $s' \in \mathsf{Out}(\lambda_n, \sigma_A)$, we apply the step for the construction of F_A to s'' and $\xi^{n+1} \in H(s'')$.

■

B.2 Soundness

Soundness of the tableau procedure with respect to unsatisfiability means that if a formula is satisfiable then its final tableau is open. To prove that, we essentially follow the same procedure as in the soundness proof for the tableau-based decision procedure for ATL in [31] and ATL$^+$ in [13].

The soundness proof establishes three main claims. First, we show that when a prestate Γ is satisfiable then at least one of the states in **states**(Γ) is satisfiable. Then, we prove that when a state Δ is satisfiable then all the prestates in **prestates**(Δ) are satisfiable. Finally, we show that no satisfiable nodes are eliminated during the elimination phase. Below, we take the input formula of the tableau procedure to be θ.

The first step of the proof consists in showing that SR is sound:

Lemma B.1. *Let Γ be a prestate of $\mathscr{T}_0^\theta$ and let $\mathscr{M}, s \models \Gamma$ for some CGM $\mathscr{M}$ and some $s \in \mathscr{M}$. Then, $\mathscr{M}, s \models \Delta$ holds for at least one $\Delta \in$ **states**(Γ).*

Proof. Straightforward from Proposition 4.1. ■

The aim of the next two lemmas is to show that the rule Next creates only satisfiable prestates from satisfiable states.

The following lemma states a semantic property, independent of the tableau construction.

Lemma B.2. *Let* $\Theta = \{\langle\langle A_1\rangle\rangle\bigcirc\varphi_1, \ldots, \langle\langle A_m\rangle\rangle\bigcirc\varphi_m, [\![A']\!]\bigcirc\psi, [\![\mathbb{A}]\!]\bigcirc\mu_1, \ldots, [\![\mathbb{A}]\!]\bigcirc\mu_n\}$ *be a set of formulae such that* $A_i \cap A_j = \emptyset$ *for every* $1 \leqslant i, j \leqslant m$, $i \neq j$ *and* $A_i \subseteq A'$ *for every* $1 \leqslant i \leqslant m$. *Let* $\mathcal{M}, s \models \Theta$ *for some CGM* $\mathcal{M}$ *and* $s \in \mathcal{M}$. *Let* $\sigma_{A_i} \in \mathsf{act}_{A_i}(s)$ *be an* A_i*-action witnessing the truth of* $\langle\langle A_i\rangle\rangle\bigcirc\varphi_i$ *at* s, *for each* $1 \leqslant i \leqslant m$, *and let, finally,* $\sigma^c_{A'} \in \mathsf{act^c}_{A'}(s)$ *be an* A'*-co-action witnessing the truth of* $[\![A']\!]\bigcirc\psi$ *at* s. *Then there exists* $s' \in \mathsf{Out}(s, \sigma_{A_1}) \cap \cdots \cap \mathsf{Out}(s, \sigma_{A_m}) \cap \mathsf{Out}(s, \sigma^c_{A'})$ *such that* $\mathcal{M}, s' \models \{\varphi_1, \ldots, \varphi_m, \psi, \mu_1, \ldots, \mu_n\}$.

Proof. Let $A = A_1 \cup \ldots \cup A_m$. Since $A_i \cap A_j = \emptyset$ for every $1 \leqslant i, j \leqslant m$, $i \neq j$, the actions $\sigma_{A_1}, \ldots, \sigma_{A_m}$ can be combined to get an A-action σ_A. This last can be arbitrarily extended to an A'-action $\sigma_{A'}$ because $A_i \subseteq A'$ for every $1 \leqslant i \leqslant m$. Finally, the so obtained $\sigma_{A'}$ can be completed by the A'-co-action $\sigma^c_{A'}$ since $A_i \subseteq A'$ for every $1 \leqslant i \leqslant m$. Moreover, for formulae of the form $[\![\mathbb{A}]\!]\bigcirc\mu_i$, $1 \leqslant i \leqslant n$, the $\mathbb{A}$-co-action is always the identity function. The resulting action vector $\sigma_{\mathbb{A}}$ leads from s to the desired s'. ■

The next lemma states that satisfiability propagates from states to their successor prestates created via rule **Next**.

Lemma B.3. *If* $\Delta \in \mathscr{T}^\theta_0$ *is a satisfiable state then all the prestates* Γ *obtained by applying the rule* ***Next*** *are satisfiable.*

Proof. Follows from Lemma B.2 and from the fact that rule **Next** ensures that every prestate Γ of Δ, that is every element of the finite set of prestates that are targets of $\xrightarrow{\sigma}$ edges outgoing from Δ, satisfies the following:

- if $\{\langle\langle A_i\rangle\rangle\bigcirc\varphi_i, \langle\langle A_j\rangle\rangle\bigcirc\varphi_j\} \subseteq \Delta$ and $\{\varphi_i, \varphi_j\} \subseteq \Gamma$, then $A_i \cap A_j = \emptyset$;
- Γ contains at most one formula of the form ψ such that $[\![A]\!]\bigcirc\psi \in \Delta$ where $A \neq \mathbb{A}$, since the number $\mathsf{co}(\sigma)$ is uniquely determined for every $\sigma \in D(\Delta)$;
- if $\{\langle\langle A_i\rangle\rangle\bigcirc\varphi_i, [\![A']\!]\bigcirc\psi\} \subseteq \Delta$ and $\{\varphi_i, \psi\} \subseteq \Gamma$, then $A_i \subseteq A'$.

■

Thus, the rule **SR** generates at least one satisfiable state from a satisfiable prestate and that the rule **Next** generates only satisfiable prestates from a satisfiable state. Hence, we can conclude that the construction phase of the tableau procedure is sound.

We now move to the elimination phase.

Lemma B.4. *Let* Θ *be a node in* $\mathscr{T}^\theta_n$. *If* Θ *is satisfiable then rule* ***ER1*** *cannot eliminate* Θ *from* $\mathscr{T}^\theta_n$.

Proof. By Lemma B.3, a satisfiable state Δ generates only satisfiable successor prestates, and, by Lemma B.1, a satisfiable prestate Γ generates at least one satisfiable state. Therefore, by definition of rule **ER1**, if a node Θ is satisfiable then it cannot be eliminated. ■

It remains to prove that a satisfiable state cannot be eliminated by rule **ER2**, either. We recall that rule **ER2** eliminates each state containing an eventuality that is not realized at that state. So we need to prove that if a state Δ is satisfiable, then every eventuality $\xi \in \Delta$ is realized at Δ at each step of the elimination phase.

Lemma B.5. *Let $\Delta \in S_n^\theta$ be a state and $\langle\langle A_p\rangle\rangle\bigcirc\varphi_p \in \Delta$ and let $\mathscr{M}, s \models \Delta$ for some CGM $\mathscr{M}$ and state $s \in \mathscr{M}$. Let, furthermore, $\sigma_{A_p} \in \mathsf{act}_{A_p}(s)$ be an A_p-action witnessing the truth of $\langle\langle A_p\rangle\rangle\bigcirc\varphi_p$ at s. Then, there exists in $\mathscr{T}_n^\theta$ an outcome set X of $\sigma_{A_p}[\langle\langle A_p\rangle\rangle\bigcirc\varphi_p]$ such that for each $\Delta' \in X$ there exists $s' \in \mathsf{Out}(s, \sigma_{A_p})$ such that $\mathscr{M}, s' \models \Delta'$.*

Proof. We consider the following set of prestates:

$$Y = \{\Gamma \in \mathbf{prestates}(\Delta) \mid \Delta \xrightarrow{\sigma_{\mathbb{A}}} \Gamma \text{ for some } \sigma_{\mathbb{A}} \sqsupseteq \sigma_{A_p}[\langle\langle A_p\rangle\rangle\bigcirc\varphi_p]\}$$

For every $\Gamma \in Y$, it follows immediately from the rule **Next** that Γ (which must contain φ_p) is either of the form

$\{\varphi_1, \ldots, \varphi_m, \psi, \mu_1, \ldots, \mu_n\}$, where $\{\langle\langle A_1\rangle\rangle\bigcirc\varphi_1, \ldots, \langle\langle A_m\rangle\rangle\bigcirc\varphi_m, [\![A']\!]\bigcirc\psi, [\![\mathbb{A}]\!]\bigcirc\mu_1, [\![\mathbb{A}]\!]\bigcirc\mu_n\} \subseteq \Delta$,

or of the form

$\{\varphi_1, \ldots, \varphi_m, \mu_1, \ldots, \mu_n\}$ where $\{\langle\langle A_1\rangle\rangle\bigcirc\varphi_1, \ldots, \langle\langle A_m\rangle\rangle\bigcirc\varphi_m, [\![\mathbb{A}]\!]\bigcirc\mu_1, [\![\mathbb{A}]\!]\bigcirc\mu_n\} \subseteq \Delta$.

Since $\mathscr{M}, s \models \Delta$, by Lemma B.2, there exists $s' \in \mathsf{Out}(s, \sigma_{A_p})$ with $\mathscr{M}, s' \models \Gamma$. Then Γ can be extended, in the tableau, to a fully expanded set Δ' containing at least one successor formula ($\langle\langle\mathbb{A}\rangle\rangle\bigcirc\top$, if nothing else) such that $\mathscr{M}, s' \models \Delta'$. This is done by choosing, for every β- or γ-formula to be processed in the procedure that computes the family of full expansions, a disjunct, resp. a γ-component, that is actually true in $\mathscr{M}$ at s' (if there are several such options, the choice is arbitrary) and adding it to the current set. ■

Corollary B.1. *Let $\Delta \in S_n^\theta$ be a state and $\langle\langle A_p\rangle\rangle\bigcirc\varphi_p \in \Delta$. Let $\mathscr{M}, s \models \Delta$ for some CGM $\mathscr{M}$ and state $s \in \mathscr{M}$. Let, furthermore, $\sigma_{A_p} \in \mathsf{act}_{A_p}(s)$ be an A_p-action witnessing the truth of $\langle\langle A_p\rangle\rangle\bigcirc\varphi_p$ at s and let $\chi \in cl(\theta)$ be a β-formula (resp. a γ-formula) and ψ be one of its β-components (resp. γ-components). Then there exists in $\mathscr{T}_n^\theta$ an outcome set X_ψ of $\sigma_{A_p}[\langle\langle A_p\rangle\rangle\bigcirc\varphi_p]$ such that for every $\Delta' \in X_\psi$ there exists $s' \in \mathsf{Out}(s, \sigma_{A_p})$ such that $\mathscr{M}, s' \models \Delta'$, and moreover, if $\mathscr{M}, s' \models \psi$, then $\psi \in \Delta'$.*

Proof. Construct X_ψ just like X was constructed in the proof of the preceding lemma, with a single modification: when dealing with the formula χ, instead of choosing arbitrarily between the different options for ψ, choose ψ which is true at s'. ■

Likewise, we obtain the following for successor formulae of the form $[\![A]\!]\bigcirc\varphi$:

Lemma B.6. *Let $\Delta \in S_n^\theta$ be a state, $[\![A'_q]\!]\bigcirc\psi_q \in \Delta$ and $\mathscr{M}, s \models \Delta$ for some CGM $\mathscr{M}$ and state $s \in \mathscr{M}$. Let, furthermore, $\sigma^c_{A'_q} \in \mathsf{act}^c{}_{A'_q}(s)$ be an A'_q-co-action witnessing the truth of $[\![A'_q]\!]\bigcirc\psi_q$ at s. Then, there exists in $\mathscr{T}_n^\theta$ an outcome set X of $\sigma^c_{A'_q}[\,[\![A'_q]\!]\bigcirc\psi_q]$ such that for each $\Delta' \in X$ there exists $s' \in \mathsf{Out}(s, \sigma^c_{A'_q})$ such that $\mathscr{M}, s' \models \Delta'$.*

The proof is analogous to the proof of Lemma B.5.

Corollary B.2. *Let $\Delta \in S_n^\theta$ be a state and and $[\![A'_q]\!]\bigcirc\psi_q \in \Delta$. Let $\mathscr{M}, s \models \Delta$ for some CGM $\mathscr{M}$ and state $s \in \mathscr{M}$. Let, furthermore, $\sigma^c_{A'_q} \in \mathsf{act}^{\mathsf{c}}{}_{A'_q}(s)$ be an A'_q-co-action witnessing the truth of $[\![A'_q]\!]\bigcirc\psi_q$ at s and let $\chi \in cl(\theta)$ be a β-formula (resp. a γ-formula), whose β_i-associate ($i \in \{1,2\}$) (resp. i-th γ-component ($i \geqslant 1$)) is χ_i. Then there exists in $\mathscr{T}_n^\theta$ an outcome set X_{χ_i} of $\sigma^c_{A'_q}[\,[\![A'_q]\!]\bigcirc\psi_q]$ such that for every $\Delta' \in X_{\chi_i}$ there exists $s' \in \mathsf{Out}(s, \sigma^c_{A'_q})$ such that $\mathscr{M}, s' \models \Delta'$, and moreover, if $\mathscr{M}, s' \models \chi_i$, then $\chi_i \in \Delta'$.*

Lemma B.7. *Let $\mathscr{R} = (R, \rightarrow)$ be a realization witness tree for a potential eventuality ξ at $\Delta \in S_n^\theta$. Then ξ is realized at Δ in $\mathscr{T}_n^\theta$.*

Proof. Straightforward from the definition of realization witness tree Section (Definition A.4) and the definitions 4.4 and 4.5. ■

We now prove the existence of a realization witness tree for any satisfiable state of a tableau containing a potential eventuality.

Lemma B.8. *Let $\xi \in \Delta$ be a potential eventuality and $\Delta \in S_n^\theta$ be satisfiable. Then there exists a realization witness tree $\mathscr{R} = (R, \rightarrow)$ for ξ at $\Delta \in S_n^\theta$. Moreover, every Δ', colouring a node of R, is satisfiable.*

Proof. We will only give the proof for potential eventualities of the type $\langle\langle A\rangle\rangle\Phi$. The case of potential eventualities of type $[\![A]\!]\Phi$ is similar.

When dealing with realization of potential eventualities, we have two cases:

1. $\mathsf{WF}(\Phi, \Delta, \gamma_{sl}(\xi, \Delta)) = \top$. This case is straightforward, the realization witness tree consists of only the root, coloured with Δ.
2. $\mathsf{WF}(\Phi, \Delta, \gamma_{sl}(\xi, \Delta)) \neq \top$.

   We start building the realization witness tree $\mathscr{R}$ with a simple tree whose root r is coloured with Δ and Φ. We construct the rest of the realization witness tree $\mathscr{R}$ step-by-step starting from Δ, ξ, Φ and r.

   Step applied from a given satisfiable state Δ_0, a given eventuality ξ_0, a given path formula Φ_0 and a node of $\mathscr{R}$:

   Let $\Phi'_0 = \mathsf{WF}(\Phi_0, \Delta_0, \gamma_{sl}(\xi_0, \Delta_0))$. There is a successor potential eventuality $\xi^1_{\Delta_0}$ of ξ_0 such that $\langle\langle A\rangle\rangle\bigcirc\xi^1_{\Delta_0} \in \Delta_0$.

   As Δ_0 is satisfiable, there exists a CGM $\mathscr{M}$ and a state $s \in \mathscr{M}$ such that $\mathscr{M}, s \models \Delta_0$, and in particular, $\mathscr{M}, s \models \langle\langle A\rangle\rangle\bigcirc\xi^1_{\Delta_0}$. Thus, there exists an A-action $\sigma_A \in \mathsf{act}_A(s)$ such that $\mathscr{M}, s' \models \xi^1_{\Delta_0}$ for all $s' \in \mathsf{Out}(s, \sigma_A)$, that is an A-action witnessing the truth of $\langle\langle A\rangle\rangle\bigcirc\xi^1_{\Delta_0}$ at s. We know that Δ_0 is satisfiable and that $\langle\langle A\rangle\rangle\bigcirc\xi^1_{\Delta_0}$ is a successor formula of the form $\langle\langle A\rangle\rangle\bigcirc\varphi$. Let p be the position of $\langle\langle A\rangle\rangle\bigcirc\xi^1_{\Delta_0}$ in the list built from the application of the rule **Next** on Δ_0. Note that $\xi_0 = \langle\langle A\rangle\rangle\Phi_0$ is a γ-formula $\in cl(\theta)$, where at least one of its γ-components,

obtained from a triple $\langle \psi, \Psi, S \rangle$, is such that $\mathsf{WF}(\Phi_0, \mathbf{FS}(\psi), S) = \top$, by the structure of the decomposition of γ-formula. Let χ be such a γ-component. So Corollary B.1 is applicable to Δ_0, and according to that corollary, there exists an outcome set X_χ of $\sigma_A[\langle\langle A \rangle\rangle \bigcirc \xi^1_{\Delta_0}]$ at Δ_0 such that, for every $\Delta' \in X_\chi$, there exists $s' \in \mathsf{Out}(s, \sigma)$ such that $\mathscr{M}, s' \models \Delta'$, and moreover, if $\mathscr{M}, s' \models \chi$, then $\chi \in \Delta'$. Thus the leaves of $\mathscr{R}$ are coloured bijectively with a node from X_χ and Φ'. The so obtained tree respects items 1 to 3 of Definition A.4; it is not necessarily that all leaves respect item 4 of this definition. For every such leaf l, we apply again the step from the corresponding Δ', ξ^1, Φ'_0 and l.
This step cannot be repeated ad libitum since $\mathscr{M}, s \models \Delta$, and in particular $\mathscr{M}, s \models \xi$.
Thus, the so constructed realization witness tree conforms to Definition A.4.

■

Lemma B.9. *Let Δ be a state in $\mathscr{T}^\theta_n$. If Δ is satisfiable then rule **ER2** cannot eliminate Δ from $\mathscr{T}^\theta_n$.*

Proof. Let $\Delta \in \mathscr{T}^\theta_n$ be a satisfiable state.
If Δ contains no eventuality, then rule ER2 is not applicable.
If Δ contains an eventuality ξ, then Lemma B.8 ensures that there exists a realization witness tree for Δ and, by Lemma B.7, we know that ξ is realized at Δ in $\mathscr{T}^\theta_n$. Therefore, ER2 cannot eliminate Δ for $\mathscr{T}^\theta_n$. ■

Theorem B.1 (Soundness)**.** *The tableau-based procedure for* ATL* *is sound with respect to unsatisfiability, that is if a formula, say θ, is satisfiable, then its final tableau $\mathscr{T}^\theta$ is open.*

Proof. Lemmas B.3–B.9 ensure that if a node is satisfiable, then it cannot be eliminated from $\mathscr{T}^\theta_n$ due to rule ER1 or rule ER2. Therefore the initial node of the tableau cannot be eliminated and therefore the final tableau $\mathscr{T}^\theta$ is open. ■

B.3 Completeness

In order to obtain a model from an open final tableau, we first extract from it *Hintikka structures* whose definition is given in subsection A.4. The proof of completeness for ATL* is very similar to the one for ATL$^+$, but we can notice that we will have to be sure that every formula with many often temporal operators, e.g. $\Box\Phi_1 \mathsf{U} \Phi_2$, is satisfied at the end of the construction of the Hintikka structure.

Completeness of the procedure means that the existence of an open tableau implies existence of a CGM. So, we start with an open tableau $\mathscr{T}^\theta$ for θ and we want to prove that θ is indeed satisfiable. The proof is constructive, as we will build from $\mathscr{T}^\theta$ a Hintikka structure $\mathscr{H}_\theta$ that can be turned into a model for θ. In order to construct that Hintikka structure, first we will extract special trees associated with potential eventualities, that can be seen as building modules to be

used to construct the entire structure. Eventually, we show that the so constructed structure is a Hintikka structure for θ.

First, we need to define *edge-labelling* of a tree.

Definition B.1 (Edge-labelling)**.** Let $\mathscr{W} = (W, \rightsquigarrow)$ be a tree and Y be a non-empty set. An *edge-labelling of $\mathscr{W}$ by Y* is a mapping l from the set of edges of $\mathscr{W}$ to the set of non-empty subsets of Y.

Definition B.2 (Tree conditions)**.** Given a tableau $\mathscr{T}^\theta$, a tree $\mathscr{W} = (W, \rightsquigarrow)$ is a *$\mathscr{T}^\theta$-tree* if the following conditions hold:

- $\mathscr{W}$ is S^θ-coloured, by some colouring mapping c.
- $\mathscr{W}$ is edge-labelled by $\bigcup_{(\Delta \in S^\theta)} \mathsf{act}_\Delta$, by some edge-labelling mapping l;
- $l(w \rightsquigarrow w') \subseteq \mathsf{act}_\Delta$ for every $w \in W$ with $c(w) = \Delta$;
- For every interior node $w \in W$ with $c(w) = \Delta$, every successor $\Gamma \in \mathscr{T}^\theta$ of Δ and every successor $\Delta' \in \mathscr{T}^\theta$ of Γ, there exists exactly one $w' \in W$ such that $l(w \rightsquigarrow w') = \{\sigma \mid \Delta \xrightarrow{\sigma} \Gamma\}$.

Definition B.3 (Rooted tree)**.** Let $\Delta \in S^\theta$. A $\mathscr{T}^\theta$-tree $\mathscr{W}$ is *rooted at* Δ if the root r of $\mathscr{W}$ is coloured with Δ.

For the purpose of our construction, we distinguish two kinds of $\mathscr{T}^\theta$-trees: *simple* or *realizing*. Their definitions are given below. Realizing $\mathscr{T}^\theta$-trees will deal especially with potential eventualities.

Definition B.4 (Simple tree)**.** A tree $\mathscr{W} = (W, \rightsquigarrow)$ is *simple* if it has no interior nodes except the root.

Simple $\mathscr{T}^\theta$-trees can be seen as one-step modules.

Definition B.5 (Realizing tree)**.** Let $\mathscr{W} = (W, \rightsquigarrow)$ be a $\mathscr{T}^\theta$-tree rooted at Δ and $\xi \in \Delta$ a potential eventuality. The tree $\mathscr{W}$ is a *realizing $\mathscr{T}^\theta$-tree for ξ*, denoted $\mathscr{W}_\xi$, if there exists a subtree $\mathscr{R}_\xi$ of $\mathscr{W}$ rooted at Δ such that $\mathscr{R}_\xi$ is a realization witness tree for ξ rooted at $\Delta \in \mathscr{T}^\theta$, where $\mathscr{T}^\theta$ is an open tableau for θ.

Lemma B.10. *Let $\Delta \in S^\theta$. Then, there exists a simple $\mathscr{T}^\theta$-tree rooted at Δ.*

Proof. We construct a simple $\mathscr{T}^\theta$-tree $\mathscr{W}$ rooted at Δ as follows. The root of $\mathscr{W}$ is a node r such that $c(r) = \Delta$. For every Γ, we select one successor state Δ' of Γ and add a successor t to $\mathscr{W}$ such that $c(t) = \Delta'$ and $l(r \rightsquigarrow t) = \{\sigma \mid \Delta \xrightarrow{\sigma} \Gamma\}$. ■

To show the existence of a realizing $\mathscr{T}^\theta$-tree for ξ at Δ, we first prove the existence of a realization witness tree $\mathscr{R}_\xi$ for ξ at Δ.

Lemma B.11. *Let $\mathscr{T}^\theta$ be an open tableau for θ and ξ be a potential eventuality realized at $\Delta \in \mathscr{T}^\theta$. Then, there exists a realization witness tree $\mathscr{R}_\xi$ for ξ at Δ in $\mathscr{T}^\theta$.*

Proof. Straightforward from Definition 4.4 and Definition 4.5. ■

Lemma B.12. *Let $\mathscr{T}^\theta$ be an open tableau for θ. Let $\xi \in \Delta \in S^\theta$ be a potential eventuality. Then, there exists a finite realizing $\mathscr{T}^\theta$-tree for ξ rooted at Δ.*

Proof. Since $\mathscr{T}^\theta$ is open, ξ is realized at Δ in $\mathscr{T}^\theta$. To construct the realizing $\mathscr{T}^\theta$-tree $\mathscr{W}_\xi$ for ξ rooted at Δ, we start from the realization witness tree $\mathscr{R}_\xi$, whose existence is given by Lemma B.11 and provisionally we take $\mathscr{W}_\xi$ to be $\mathscr{R}_\xi$. The problem with $\mathscr{R}_\xi$ is that for some $\sigma \in \mathsf{act}_(\Delta)$ at some node w of $\mathscr{R}_\xi$, there is no edge $w \rightsquigarrow w'$ such that $l(w \rightsquigarrow w') \ni \sigma$. Therefore, to extend $\mathscr{W}_\xi$ into a realizing $\mathscr{T}^\theta$-tree, for every such node w, we pick one of the successor states of $c(w)$ via σ, say Δ' and add a node w' to $\mathscr{W}_\xi$ such that $c(w') = \Delta'$ and $l(w \rightsquigarrow w') \ni \sigma$. ■

We now construct a final structure, denoted by $\mathfrak{F}$, from simple and realizing $\mathscr{T}^\theta$-trees. This construction is made step-by-step. At the end of the construction, we prove that $\mathfrak{F}$ is indeed a Hintikka structure.

Step 1. We define a grid $\mathscr{F}$ of size $m \times n$, where m is the number of eventualities occurring in $\mathscr{T}^\theta$ and n the number of states of $\mathscr{T}^\theta$. Each row of that grid is labelled by one of the eventualities and each column by a state of $\mathscr{T}^\theta$ previously ordered by name ($\Delta_i < \Delta_j$ if $i < j$). We denote by ξ_i the eventuality associated to row $0 \leq i \leq m$, we denote by Δ_j the state associated to the column $0 \leq j \leq n$. The content $\mathscr{F}(i,j)$ of each intersection between a row i and a column j of $\mathscr{F}$ is as follows: if $\xi_i \in \Delta_j$, then $\mathscr{F}(i,j)$ is the realizing $\mathscr{T}^\theta$-tree for ξ_i rooted at Δ_j; otherwise, $\mathscr{F}(i,j)$ is the simple $\mathscr{T}^\theta$-tree rooted at Δ_j.

Step 2. We make a queue Q that will contain eventualities occurring in $\mathscr{T}^\theta$. The first element of Q is either θ, if θ is an eventuality, or the eventuality associated to the first row of the grid defined just above. Let ξ_i be the first element of the queue, so that $Q(0) = \xi_i$. Then we add to Q all the eventualities following the order of grid's rows and cycling if necessary, that is $Q(k) = \xi_{((i+k) \bmod m)}$ for $k \in [1, m-1]$.

Step 3. Let Δ be one of the states containing θ. Next, we take the element $\mathscr{F}(Q(0), \Delta)$ of the grid. The root of $\mathscr{F}(Q(0), \Delta)$ is then the root of $\mathfrak{F}$. Then we take one-by-one in order all the elements of the rest of the queue and do the following:

Let $Q(i)$ be the current element of the queue to be treated. For every dead-end state $w \in \mathfrak{F}$, that is a state without successors, such that $c(w) = \Delta_j$, we add the tree $\mathscr{F}(Q(i), \Delta_j)$ by merging the dead-end state w with the root of $\mathscr{F}(Q(i), \Delta_j)$;

Step 4. Finally, we ensure that $\mathfrak{F}$ finite. While there is a dead-end state in $\mathfrak{F}$, say w with $c(w) = \Delta_j$, we choose a component from the row $\mathscr{F}(\Delta_j)$ as follows:

- With priority we choose a component $\mathscr{F}(i,\Delta_j), 0 \leq i \leq m$ already occurring in $\mathfrak{F}$. Let r be the root of the component $\mathscr{F}(i,\Delta_j), 0 \leq i \leq m$ inside $\mathfrak{F}$. Then we add an arrow $\rightsquigarrow$ between every predecessor v of w and the root r and labelled this arrow with $l(v \rightsquigarrow w)$. Then we delete the node $w \in \mathfrak{F}$.
- Otherwise, if the chosen component $\mathscr{F}(i,\Delta_j)$ is not already occurring in $\mathfrak{F}$ then we add the new component to $\mathfrak{F}$ as usual by merging the root of the component with the dead-end state w.

When there are no longer dead-ends in $\mathfrak{F}$, the structure is completed and we have obtained our final structure.

The next lemma will be used to prove that the structure $\mathfrak{F}$ obtained at the end of the construction is indeed a Hintikka structure.

Lemma B.13. *Let $\mathscr{T}$ be a $\mathscr{T}^{\theta}$-tree rooted at $\Delta = c(w)$. Then, the following holds:*

1. *If $\langle\langle A\rangle\rangle\bigcirc\varphi \in \Delta$, then there exists an A-action $\sigma_A \in \mathsf{act}_A(\Delta)$ such that $\varphi \in c(w') = \Delta'$ where $l(w \rightsquigarrow w') \ni \sigma$ for every $\sigma \sqsupseteq \sigma_A$.*
2. *If $[\![A]\!]\bigcirc\varphi \in \Delta$, where $A \neq \mathbb{A}$, then there exists a A-co-action $\sigma^c_A \in \mathsf{act}^{\mathsf{c}}{}_A(\Delta)$ such that $\varphi \in c(w') = \Delta'$ where $l(w \rightsquigarrow w') \ni \sigma$ for every $\sigma \sqsupseteq \sigma^c_A(\sigma_A)$.*

Proof. We recall that all successor formulae of $\Delta \in S^{\theta}$ are ordered at the application of the rule Next to Δ.

(1) Suppose that $\langle\langle A\rangle\rangle\bigcirc\varphi \in \Delta$. Then the required A-action is $\sigma_A[\langle\langle A\rangle\rangle\bigcirc\varphi]$. Indeed, it immediately follows from the rule Next that for every $\sigma \sqsupseteq \sigma_A$ in the initial tableau $\mathscr{T}^{\theta}_0$, if $\Delta \xrightarrow{\sigma} \Gamma$, then $\varphi \in \Gamma$ and $\varphi \in \Delta'$ since Δ' is a full expansion of Γ. The statement (1) of the lemma follows.

(2) Suppose that $[\![A]\!]\bigcirc\varphi \in \Delta$, *Case 1:*$A \neq \mathbb{A}$. We consider an arbitrary $\sigma_A \in \mathsf{act}_A(\Delta)$. Then σ_A can be extended to an action vector $\sigma' \sqsupseteq \sigma$. Let $N(\sigma_A)$ be the set $\{i \mid \sigma_A(i) \geq m\}$, where m is the number of successor formulae of the form $\langle\langle A\rangle\rangle\bigcirc\varphi$ in Δ, and let $\mathsf{co}(\sigma_A) = \left(\sum_{i\in N(\sigma_A)}(\sigma_A(i)-m)\right) \bmod l$, where l is the number of successor formulae of the form $[\![A]\!]\bigcirc\varphi$ in Δ. Now, we consider $\sigma' \sqsupseteq \sigma_A$ defined as follows: $\sigma'_{\mathsf{b}} = ((q - \mathsf{co}(\sigma_A)) \bmod l + m$ and $\sigma'_{\mathsf{a}'} = m$ for any $\mathsf{a}' \in \mathbb{A} - (A \cup \{\mathsf{b}\})$, where $\mathsf{b} \in \mathbb{A} - A$. Thus, we have $\mathbb{A} - A \subseteq N(\sigma)$ and also $\mathsf{co}(\sigma') = (\mathsf{co}(\sigma_A) + (q - \mathsf{co}(\sigma_A))) \bmod l) + m = q$. Therefore, for this arbitrarily chosen σ_A there exists at least one state, say Δ', such that $\Delta \xrightarrow{\sigma} \Gamma \Longrightarrow \Delta'$ and $\varphi \in \Delta'$.

Case 2: $A = \mathbb{A}$. Then, by virtue of (H2), $\langle\langle\emptyset\rangle\rangle\bigcirc\neg\varphi \in \Delta$ and thus, by the rule Next, $\neg\varphi \in \Gamma$ for every successor $Gamma$ of Δ. Then, $\neg\varphi \in \Delta'$ for every Δ' that is a successor of Δ in $\mathscr{T}^{\theta}$ and hence the colouring set of every leaf of $\mathscr{T}$. Then, the (unique) co-$\mathbb{A}$-actions, which is an identity function, has the required properties.

The statement (2) of the lemma follows. ■

Theorem B.2. *The tableau-based decision procedure for ATL^* is complete with respect to unsatisfiability, that is if a tableau for an input formula θ is open, then the formula θ is satisfiable.*

Proof. The structure $\mathfrak{F}$ constructed from $\mathscr{T}^\theta$ is a Hintikka structure. Indeed, H1-H4 of Definition A.7 are satisfied since the nodes of $\mathfrak{F}$ are nodes of $\mathscr{T}^\theta$. H5 of the same definition essentially follows from Lemma B.13. Whenever a node w of $\mathfrak{F}$ contains a potential eventuality ξ, this means that this eventuality will stay in the queue (see construction of $\mathfrak{F}$ above) until realized. Moreover, if the $\mathscr{T}^\theta$-tree $\mathscr{W}$ chosen to complete $\mathfrak{F}$ from w does not realize ξ, either ξ or one of its descendants is present in each newly generated dead-end of $\mathfrak{F}$. So, when it is the turn to realize ξ we add to each dead-end state the realizing $\mathscr{T}^\theta$-tree for ξ. This, together with Lemma B.13, guarantees that there exists a realization witness tree for ξ on $\mathfrak{F}$ at w. Thus, H6 of Definition A.7 is satisfied, too.

By construction, the structure $\mathfrak{F}$ is a concurrent game Hintikka structure for θ, thus Theorem A.1 can be applied to obtain from it a model for θ. Thus θ is satisfiable. ■

APPENDIX C

NOTATION LIST

Notation	Description	Section
$\Box$ (or G)	temporal operator "Always"	1.2; 2.2.1
$\Diamond$ (or F)	temporal operator "Sometime"	1.2; 2.2.1
$\bigcirc$ (or X)	temporal operator "Next"	1.2; 2.2.1
U	temporal operator "Until"	1.2; 2.2.1
$\overset{\infty}{\Diamond}$ (or GF)	temporal operator "Infinitely often"	2.2.1
$\overset{\infty}{\Box}$ (or FG)	temporal operator "Sometime always"	2.2.1
$\theta, \varphi, \psi, \ldots$	state formula; θ is an initial formula	2.2.1
$\Phi, \Psi, \ldots$	path formula	2.2.1
$\oplus^+, \otimes^+$	operators for the dec^+ function	4.1.1
$\oplus^*, \otimes^*$	operators for the dec^* function	4.1.2
$\dot{\wedge}, \dot{\vee}$	version of the operators $\wedge$ and $\vee$ respectively where the associativity, commutativity, idempotence and identity element properties is embedded in the syntax	4.1.2
$\sqsupseteq$	extension of vector	A.1
$\rightsquigarrow$	relation on $\mathscr{W}$	B.3:Def. B.2
$\mathbb{A}$	non-empty set of all agents of a CGM	2.1
$\mathbb{A}_\theta$	set of agents that are mentioned in θ	3.1
A	a coalition of agents	1.2
A	CTL path quantifier "for all paths"	1.2
$\langle\langle A \rangle\rangle$	Strategic quantifier "the coalition A has a strategy to enforce"	1.2
$[\![A]\!]$	Strategic quantifier "the coalition A cannot avoid"	1.2

Notation	Description	Section
act_a	map $\mathsf{act}_\mathsf{a} : \mathbb{S} \to \mathscr{P}(\mathsf{Act}_\mathsf{a}) \setminus \emptyset$. Define for each state s the actions available to a at s	2.1:Def. 2.2
act_A	map $\mathsf{act}_A : \mathbb{S} \to \mathscr{P}(\mathsf{Act}_A) \setminus \emptyset$. Define for each state s the actions available to the coalition A at s	2.1:Def. 2.2
Act_a	non-empty set of actions for a given agent a	2.1:Def. 2.2
Act_A	non-empty set of actions for a given coalition A	2.1:Def. 2.2
$\mathsf{Act}^\mathsf{c}{}_A$	an A-co-actions $\mathsf{Act}^\mathsf{c}{}_A : \mathsf{Act}_A \to \mathsf{Act}_{\mathbb{A}\setminus A}$	2.2.2
$\mathsf{ATL}^+_\mathsf{p}$	the set of ATL^+ path formulae	4.1
$\mathsf{ATL}^+_\mathsf{s}$	the set of ATL^+ state formulae	4.1
$\mathsf{ATL}^*_\mathsf{p}$	the set of ATL^* path formulae	4.1
$\mathsf{ATL}^*_\mathsf{s}$	the set of ATL^* state formulae	4.1
B	Bob	1
$\mathsf{cl}(\theta)$	closure for the formula θ	3.2.1
$\mathsf{co}(\sigma)$	function used for the rule next	3.2.3; 4.3
dec^+	decomposition function of a path formula for ATL^+	4.1.1
$\mathsf{dec}^*(\Phi)$	decomposition function of a path formula for ATL^*	4.1.2
$D(\Delta)$	the set of action vectors starting from a state Δ in a tableau	3.2.3; 4.3
$D_A(\Delta)$	the set of A-actions starting from a state Δ in a tableau	A.3.2;
$D^c_A(\Delta)$	the set of A-co-actions starting from a state Δ in a tableau	A.3.2
Δ	state in a tableau	
E	CTL path quantifier “for some path”	1.2
$\mathscr{F}$	grid for the construction of the Hintikka structure	B.3 (P.91)
$\mathfrak{F}$	structure to get Hintikka structure	B.3 (P.91)
F_A	an A-strategy	2.2.2
F^c_A	an A-co-strategie $F^c_A : \mathsf{Strat}_\mathscr{M}(A) \times \mathsf{Hist}_\mathscr{M} \to \mathsf{Act}_{\mathbb{A}\setminus A}$	2.2.2
F_{P_A}	a partial A-strategy	B.1
$\mathbf{FS}(\Gamma)$	full saturated sets of the set Γ for ATL	3.2.2
$\mathbf{FS}^+(\Gamma)$, $\mathbf{FS}^*(\Gamma)$	full saturated sets of the set Γ for ATL^+ and ATL^*	4.2
Γ	prestate in a tableau	
$\gamma(\psi,\Psi)$	the γ-component for ATL^+ associated to the pair $\langle\psi,\Psi\rangle$	4.1.3
$\gamma_c(\psi,\Psi,S)$	the γ-component for ATL^* associated to the triple $\langle\psi,\Psi,S\rangle$	4.1.3
$\gamma_s(\psi,\Psi,S)$	the γ-set associated to the triple $\langle\psi,\Psi,S\rangle$	4.1.3
$\gamma_l(\xi,\Delta)$	γ-component linked to an eventuality ξ and a state Δ	4.2
$\gamma_{sl}(\xi,\Delta)$	γ-set linked to an eventuality ξ and a state Δ	4.2
h	history in a CGM	2.2.2
H	Hugo	1
$\mathscr{H}$	Hintikka structure	A.4

Notation	Description	Section
$\mathscr{H}_\theta$	a Hintikka structure for the formula θ	B.3 (P.89)
H1...H6	conditions in Hintikka structures	A.4.2:Def. A.7
$\mathsf{Hist}_{\mathscr{M}}$	the set of all histories in the CGM $\mathscr{M}$	2.2.2
L	CGM labelling function $L : \mathbb{S} \to \mathscr{P}(\mathbb{P})$	2.1
λ	play in a CGM	2.2.2
$\|\lambda\|$	the length of λ when λ is finite	2.2.2
$\mathsf{last}(\lambda)$	the final state of λ when λ is finite	2.2.2
λ_0	the initial state of the play λ	2.2.2
λ_i	the (i+1)th state of the play λ	2.2.2
$\lambda_{\leq i}$	the prefix $\lambda_0, \ldots, \lambda_i$ of the play λ	2.2.2
$\lambda_{\geq i}$	the suffix $\lambda_i, \lambda_{i+1}, \ldots$ of the play λ	2.2.2
$\lambda_{i\sim j}$	the subsequence $\lambda_i, \ldots, \lambda_j$ of the play λ	2.2.2
$\mathscr{M}$	a CGM	2.1:Def. 2.4
$\mathsf{out}(s, \sigma_{\mathbb{A}})$	transition function in a CGM	2.1:Def. 2.2
$\mathsf{Out}(s, \sigma_A)$	outcome of an A-action at a given state s	2.1:Def. 2.3
$\mathsf{Out}(s, \sigma_A^c)$	outcome of an A-co-action at a given state s	2.1:Def. 2.3
$\mathbb{P}$	non-empty set of atomic propositions	2.1
$\mathsf{Plays}_{\mathscr{M}}$	the set of plays in the CGM $\mathscr{M}$	2.2.2
$\mathsf{Plays}_{\mathscr{M}}$ (s)	the set of plays in the CGM $\mathscr{M}$ with initial state s	2.2.2
$\mathsf{Plays}_{\mathscr{M}}(s, F_A)$	the set of plays in the CGM $\mathscr{M}$ starting at s and consistent with F_A	2.2.2
Q	queue of eventualities to realize	B.3 (P.91)
$\mathscr{R}$	realization tree	A
$\mathscr{R}_\xi$	realization tree for the eventuality ξ	B.3:Def. B.5
Realized	function that determines if an eventuality is immediately realized (ATL^+)	4.4.1
$\mathscr{S}$	a CGS	2.1:Def. 2.2
$\mathbb{S}$	non-empty set of states	2.1
S^θ	set of nodes (states and prestates) in $\mathscr{T}^\theta$	3.3
S^θ_n	set of nodes (states and prestates) in $\mathscr{T}^\theta_n$	3.3
$\mathsf{Strat}_{\mathscr{M}}(A)$	the set of strategies for the coalition A in the CGM $\mathscr{M}$	2.2.2
$Succ(\Delta, \langle\langle A\rangle\rangle \bigcirc \varphi)$, $Succ(\Delta, [\![A]\!] \bigcirc \varphi)$	function to get successors of Δ associated to $\langle\langle A\rangle\rangle \bigcirc \varphi$ and $[\![A]\!] \bigcirc \varphi$ respectively	3.1; 4.1
$\mathscr{T}^\theta$	(final) tableau for the formula θ	3.1
$\mathscr{T}^\theta_0$	initial tableau for the formula θ	3.1
$\mathscr{T}^\theta_n$	nth intermediate tableau for the formula θ	3.1

Notation	**Description**	**Section**
σ_A	an A-action	2.1:Def. 2.2
$\sigma_{\mathbb{A}}$	an action vector	2.1:Def. 2.2
σ_A^c	an A-co-Action	A.1
$\sigma_A[\langle\langle A\rangle\rangle \bigcirc \varphi]$	the unique A-action enforcing φ from a given state of a tableau	A.3.2
$\sigma_A^c[[\![A]\!] \bigcirc \varphi]$	the A-co-Action enforcing φ from a given state of a tableau	A.3.2
$\mathscr{W}$	a $\mathscr{T}^\theta$-tree	B.3:Def.B.1
$\mathscr{W}_\xi$	a $\mathscr{T}^\theta$-tree for ξ	B.3:Def. B.5
WF	function that determines which objectives of a path formula are not immediately realized (ATL*)	4.4.2
ξ	a potential eventuality	4.4.1:Def. 4.2
ξ^i	ith descendant potential eventuality of the eventuality ξ	4.4.1:Def. 4.2
ξ^1_Δ	the descendant potential eventuality of the eventuality ξ with regards to $\gamma_l(\xi,\Delta)$	4.4.1:Def. 4.2
X_ψ	outcome set that is relative to the formula ψ	B.2:Cor. B.1

BIBLIOGRAPHY

[1] L. ACETO, A. INGOLFSDOTTIR, AND J. SRBA, *The algorithmics of bisimilarity*, in Advanced Topics in Bisimulation and Coinduction, D. Sangiorgi and J. Rutten, eds., Cambridge University Press, 2011, pp. 100–172.

[2] T. ÅGOTNES, V. GORANKO, AND W. JAMROGA, *Alternating-time temporal logics with irrevocable strategies*, in Proceedings of the 11th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-2007), Brussels, Belgium, June 25-27, 2007, D. Samet, ed., 2007, pp. 15–24.

[3] R. ALUR, T. A. HENZINGER, AND O. KUPFERMAN, *Alternating-time temporal logic*, in 38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997, IEEE Computer Society, 1997, pp. 100–109.

[4] R. ALUR, T. A. HENZINGER, AND O. KUPFERMAN, *Alternating-time temporal logic*, Journal of the ACM (JACM), 49 (2002), pp. 672–713.

[5] W. B. ARTHUR, *On designing economic agents that behave like human agents*, Journal of Evolutionary Economics, 3 (1993), pp. 1–22.

[6] R. AXTELL, *Why agents?: on the varied motivations for agent computing in the social sciences*, Center on Social and Economics Dynamics - The Brookings Institution, (2000), pp. 1–23.

[7] E. W. BETH, *Semantic Entailment and Formal Derivability*, vol. 18 of Nieuwe Reeks, Med. Konkl. Nederl. Akad. v. Wetensch, 1955.

[8] ——, *The foundations of mathematics: a study in the philosophy of science*, Studies in logic and the foundations of mathematics, North-Holland Pub. Co., 1959.

[9] T. BRIHAYE, A. D. C. LOPES, F. LAROUSSINIE, AND N. MARKEY, *ATL with strategy contexts and bounded memory*, in Logical Foundations of Computer Science, International Symposium, LFCS 2009, Deerfield Beach, FL, USA, January 3-6, 2009. Proceedings, 2009, pp. 92–106.

[10] M. BROWNE, E. CLARKE, AND O. GRÜMBERG, *Characterizing finite Kripke structures in propositional temporal logic*, 1988.

[11] R. M. BURSTALL, *Program proving as hand simulation with a little induction*, in IFIP Congress, 1974, pp. 308–312.

[12] C. E. G. CENA, P. F. CÁRDENAS, R. S. PAZMIÑO, L. PUGLISI, AND R. A. SANTONJA, *A cooperative multi-agent robotics system: Design and modelling*, Expert Syst. Appl., 40 (2013), pp. 4737–4748.

[13] S. CERRITO, A. DAVID, AND V. GORANKO, *Optimal tableaux-based decision procedure for testing satisfiability in the alternating-time temporal logic ATL+*, Automated Reasoning - 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 19-22, 2014. Proceedings, 8562 (2014), pp. 277–291.

[14] ——, *Optimal tableaux method for constructive satisfiability testing and model synthesis in the alternating-time temporal logic ATL+*, CoRR, abs/1407.4 (2014).

[15] ——, *Optimal tableau method for constructive satisfiability testing and model synthesis in the alternating-time temporal logic ATL+*, Transactions On Computational Logic, To be published (2015).

[16] E. CLARKE, O. GRUMBERG, AND D. PELED, *Model Checking*, MIT Press, 1999.

[17] E. M. CLARKE, A. BIERE, R. RAIMI, AND Y. ZHU, *Bounded model checking using satisfiability solving*, Formal Methods in System Design, 19 (2001), pp. 7–34.

[18] E. M. CLARKE AND E. A. EMERSON, *Design and synthesis of synchronization skeletons using branching-time temporal logic*, in Logics of Programs, Workshop, Yorktown Heights, New York, May 1981, D. Kozen, ed., vol. 131 of Lecture Notes in Computer Science, Springer, 1981, pp. 52–71.

[19] A. DAVID, *TATL: implementation of ATL tableau-based decision procedure*, in Automated Reasoning with Analytic Tableaux and Related Methods - 22th International Conference, TABLEAUX 2013, Nancy, France, September 16-19, 2013. Proceedings, D. Galmiche and D. Larchey-Wendling, eds., vol. 8123 of Lecture Notes in Computer Science, Springer, 2013, pp. 97–103.

[20] ——, *Deciding ATL* satisfiability by tableaux*, in Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings, A. P. Felty and A. Middeldorp, eds., vol. 9195 of Lecture Notes in Computer Science, Springer, 2015, pp. 214–228.

[21] M. DAVIS AND H. PUTNAM, *A computing procedure for quantification theory*, J. ACM, 7 (1960), pp. 201–215.

[22] G. v. DRIMMELEN, *Satisfiability in alternating-time temporal logic*, 18th Annual IEEE Symposium of Logic in Computer Science, 2003. Proceedings., (2003).

[23] E. A. EMERSON AND E. M. CLARKE, *Characterizing correctness properties of parallel programs using fixpoints*, in Automata, Languages and Programming, 7th Colloquium, Noordweijkerhout, The Netherland, July 14-18, 1980, Proceedings, J. W. de Bakker and J. van Leeuwen, eds., vol. 85 of Lecture Notes in Computer Science, Springer, 1980, pp. 169–181.

[24] E. A. EMERSON AND J. Y. HALPERN, *"sometimes" and "not never" revisited: on branching versus linear time temporal logic*, Journal of the ACM, 33 (1986), pp. 151–178.

[25] J. FERBER, *Les Systèmes multi-agents: vers une intelligence collective*, I.I.A. Informatique intelligence artificielle, InterEditions, 1995.

[26] M. FITTING, *Proof methods for modal and intuitionistic logics*, Synthese library ; v. 169., D. Reidel ; Sold and distributed in the U.S.A. and Canada by Kluwer Boston, Dordrecht, Holland ; Boston, U.S.A. Hingham, MA, 1983.
Melvin Fitting. Includes index. Bibliography: p. 526-539.

[27] J. FRANCO AND J. MARTIN, *A history of satisfiability*, in Handbook of Satisfiability, A. Biere, M. Heule, H. van Maaren, and T. Walsh, eds., vol. 185 of Frontiers in Artificial Intelligence and Applications, IOS Press, 2009, pp. 3–74.

[28] O. FRIEDMANN, M. LATTE, AND M. LANGE, *A decision procedure for CTL* based on tableaux and automata*, Automated Reasoning, (2010), pp. 331–345.

[29] V. GORANKO, *Coalition games and alternating temporal logics*, in Proceedings of the 8th conference on Theoretical aspects of rationality and knowledge, Morgan Kaufmann Publishers Inc., 2001, p. 259–272.

[30] V. GORANKO AND W. JAMROGA, *Comparing semantics of logics for multi-agent systems*, Synthese, (2004), pp. 77–116.

[31] V. GORANKO AND D. SHKATOV, *Tableau-based decision procedures for logics of strategic ability in multiagent systems*, ACM Transactions on Computational Logic, 11 (2009), p. 3:1–3:51.

[32] V. GORANKO AND G. VAN DRIMMELEN, *Complete axiomatization and decidability of alternating-time temporal logic*, Theoretical Computer Science, 353 (2006), pp. 93–117.

[33] K. J. J. HINTIKKA, *Two papers on symbolic logic: Form and content in quantification theory and Reductions in the theory of types*, Acta philcsophica Fennica. Fasc. 8, 1955.

[34] J. H. HOLLAND AND J. H. MILLER, *Artificial adaptive agents in economic theory*, American Economic Review, 81 (1991), pp. 365–371.

[35] W. JAMROGA AND N. BULLING, *Comparing variants of strategic ability*, IJCAI, (2011), p. 252–257.

[36] H. W. KAMP, *Tense Logic and the Theory of Linear Order*, PhD thesis, 1968.

[37] P. C. KANELLAKIS AND S. A. SMOLKA, *CCS expressions, finite state processes, and three problems of equivalence*, Inf. Comput., 86 (1990), pp. 43–68.

[38] F. KRÖGER, *A uniform logical basis for the description*, in Formal Description of Programming Concepts, E. Neuhold, ed., North Holland, Amsterdam, 1978, pp. 441–459.

[39] ——, *Temporal Logic of Programs*, vol. 8 of {EATCS} Monographs on Theoretical Computer Science, Springer, 1987.

[40] O. KUPFERMAN, *Recent challenges and ideas in temporal synthesis*, in SOFSEM 2012: Theory and Practice of Computer Science - 38th Conference on Current Trends in Theory and Practice of Computer Science, Špindlerův Mlýn, Czech Republic, January 21-27, 2012. Proceedings, M. Bieliková, G. Friedrich, G. Gottlob, S. Katzenbeisser, and G. Turán, eds., vol. 7147 of Lecture Notes in Computer Science, Springer, 2012, pp. 88–98.

[41] F. LAROUSSINIE, N. MARKEY, AND G. OREIBY, *On the expressiveness and complexity of ATL*, Foundations of Software Science and Computational Structures, 10th International Conference, FOSSACS 2007, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2007, Braga, Portugal, March 24-April 1, 2007, Proceedings, 4423 (2007), pp. 243–257.

[42] F. LAROUSSINIE, N. MARKEY, AND G. OREIBY, *On the expressiveness and complexity of ATL*, Logical Methods in Computer Science, 4 (2008).

[43] J. LIU AND J. WU, *Multi-agent Robotic Systems*, CRC Press, Inc., 1st ed., 2001.

[44] C. NALON, L. ZHANG, C. DIXON, AND U. HUSTADT, *A resolution-based calculus for coalition logic*, J. Log. Comput., 24 (2014), pp. 883–917.

[45] C. L. PAGE, P. D'AQUINO, M. ETIENNE, AND F. BOUSQUET, *Processus participatif de conception et d'usage de simulations multi-agents. application à la gestion des ressources renouvelables*, in Systèmes Multi-Agents Défis scientifiques et nouveaux usages - JFSMA 04 - Douxièmes journées francophones sur les systèmes multi-agents, Paris, France, November 24-26, 2004, O. Boissier and Z. Guessoum, eds., Lavoisier, 2004, pp. 33–46.

[46] R. PAIGE AND R. TARJAN, *Three partition refinement algorithms*, SIAM Journal on Computing, 16 (1987), pp. 973–989.

[47] M. PAULY, *Logic for Social Software*, PhD thesis, 2001.

[48] ——, *A modal logic for coalitional power in games*, Journal of Logic and Computation, 12 (2002), pp. 149–166.

[49] A. PEREIRA, L. P. REIS, AND P. DUARTE, *EcoSimNet: A multi-agent system for ecological simulation and optimization*, in Progress in Artificial Intelligence, 14th Portuguese Conference on Artificial Intelligence, EPIA 2009, Aveiro, Portugal, October 12-15, 2009. Proceedings, L. S. Lopes, N. Lau, P. Mariano, and L. M. Rocha, eds., vol. 5816 of Lecture Notes in Computer Science, Springer, 2009, pp. 473–484.

[50] N. PITERMAN, *Synthesis from temporal specifications: New applications in robotics and model-driven development*, Mathematical Foundations of Computer Science 2013, (2013), pp. 45–49.

[51] A. PNUELI, *The temporal logic of programs*, in 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977, {IEEE} Computer Society, 1977, pp. 46–57.

[52] ——, *The temporal semantics of concurrent programs*, in Semantics of Concurrent Computation, Proceedings of the International Symposium, Evian, France, July 2-4, 1979, G. Kahn, ed., vol. 70 of Lecture Notes in Computer Science, Springer, 1979, pp. 1–20.

[53] ——, *The temporal semantics of concurrent programs*, Theoretical Computer Science, 13 (1981), pp. 45–60.

[54] A. PRIOR, *Time and Modality*, Oxford University Press, 1957.

[55] J.-P. QUEILLE AND J. SIFAKIS, *Specification and verification of concurrent systems in cesar*, in Proceedings of the 5th Colloquium on International Symposium on Programming, vol. 137, 1982, pp. 337–351.

[56] M. REYNOLDS, *A faster tableau for CTL**, arXiv preprint arXiv:1307.4468, 119 (2013), pp. 50–63.

[57] ——, *A faster tableau for CTL* - long report*, tech. rep., 2013.

[58] J. A. ROBINSON, *A machine-oriented logic based on the resolution principle*, J. ACM, 12 (1965), pp. 23–41.

[59] B. ROCHE, J.-F. GUÉGAN, AND F. BOUSQUET, *Multi-agent systems in epidemiology : a first step for computational biology in the study of vector-borne disease transmission*, Bmc Bioinformatics, 9 (2008), p. 435.

[60] S. J. RUSSELL AND P. NORVIG, *Artificial Intelligence - A Modern Approach (3. internat. ed.)*, Pearson Education, 2010.

[61] S. SCHEWE, *ATL* satisfiability is 2ExpTime-complete*, Automata, Languages and Programming, 5126 (2008), pp. 373–385.

[62] J. SIFAKIS, *The control of asynchronous systems: concepts, properties, static analysis*, PhD thesis, 1979.

[63] R. M. SMULLYAN, *First-order logic*, Ergebnisse der Mathematik und ihrer Grenzgebiete, Springer-Verlag, 1968.

[64] G. H. VON WRIGHT, *And Next*, Acta philosophica Fennica, 18 (1965), pp. 293–304.

[65] D. WALTHER, C. LUTZ, F. WOLTER, AND M. WOOLDRIDGE, *ATL satisfiability is indeed EXPTIME-complete*, Journal of Logic and Computation, 16 (2006), pp. 765–787.

[66] P. WOLPER, *The tableau method for temporal logic: An overview*, Logique et Analyse, 28 (1985), pp. 119–136.

[67] M. WOOLDRIDGE, *Intelligent agents*, in Multiagent Systems, G. Weiss, ed., MIT Press, 2013, ch. 1, pp. 3–50.

TOWARDS SYNTHESIZING OPEN SYSTEMS:
TABLEAUX FOR MULTI-AGENT TEMPORAL LOGICS

Keywords: Alternating-time temporal logic, ATL*, Satisfiability, Tableaux, Automated theorem prover.

In this thesis, we try to provide automated tools to design safe open systems. Open systems, which can be viewed as multi-agent systems, may be specified in ATL. The logic ATL has been especially introduced for that purpose. There exist two relevant extensions of ATL, namely ATL$^+$ and ATL* (ATL$^+$ being a restriction of ATL*). ATL$^+$ allows Boolean combination of temporal operators, and ATL* also allows nesting of temporal operators. The tableau-based decision procedure for ATL is a constructive method to test the satisfiability of a given specification. It is constructive in the sense that it is possible to extract a model from the obtained tableau, whenever the root formula is indeed satisfiable.

In this thesis, we propose two tableau-based decision procedures for ATL$^+$ and ATL*, as well as an implementation of these procedures. Our procedures are sound, complete and optimal. Indeed, our two procedures run in 2EXPTIME. Up to our knowledge our implementation is the first running tool to decide satisfiability of both ATL and ATL* formulae.

In the perspectives of this thesis, we discuss how it is possible to improve the extraction of models from tableaux for ATL, ATL$^+$ and ATL*. We would like to obtain relatively small models at the end.

Dans cette thèse, nous essayons de fournir des outils automatisés pour élaborer des systèmes ouverts sûrs. Les systèmes ouverts, qui peuvent être vus comme des systèmes multi-agents, peuvent être spécifiés en ATL. La logique ATL a été introduite dans ce but précis. Il existe deux extensions intéressantes d'ATL, à savoir ATL$^+$ et ATL* (ATL$^+$ étant une restriction d'ATL*). ATL$^+$ permet la combinaison Booléenne d'opérateurs temporels, et ATL * permet également l'imbrication d'opérateurs temporels.

La procédure de décision basée sur les tableaux pour ATL est une méthode constructive pour tester la satisfiabilité d'une spécification donnée. Elle est constructive dans le sens qu'il est possible d'extraire un modèle depuis le tableau obtenu, lorsque la formule de départ est satisfiable.

Dans cette thèse, nous proposons deux procédures de décision basées sur les tableaux pour ATL$^+$ and ATL*, ainsi qu'une implémentation de ces procédures. Nos procédures sont correctes, complètes et optimales. En effet, nos deux procédures s'exécutent en 2EXPTIME. A notre connaissance, notre implémentation est le premier exécutable pour décider la satisfiabilité des formules ATL et ATL*.

En perspective de cette thèse, nous discutons de la possibilité d'améliorer l'extraction de modèles depuis les tableaux pour ATL, ATL$^+$ and ATL*. Nous aimerions obtenir à la fin des modèles relativement petits.