On the Control of Asynchronous Automata

Hugo Gimbert

On the Control of Asynchronous Automata

Hugo Gimbert
LaBRI, CNRS, Université de Bordeaux, France
hugo.gimbert@cnrs.fr

April 22, 2017

Abstract

The decidability of the distributed version of the Ramadge and Wonham controller synthesis problem [10], where both the plant and the controllers are modeled as asynchronous automata [11, 1] and the controllers have causal memory is a challenging open problem [6, 7]. There exist three classes of plants for which the existence of a correct controller with causal memory has been shown decidable: when the dependency graph of actions is series-parallel, when the processes are connectedly communicating and when the dependency graph of processes is a tree. We design a class of plants, called decomposable games, with a decidable controller synthesis problem. This provides a unified proof of the three existing decidability results as well as new examples of decidable plants.

1 Introduction

The decidability of the distributed version of the Ramadge and Wonham control problem [10], where both the plant and the controllers are modeled as asynchronous automata [11, 1] and the controllers have causal memory is a challenging open problem. Very good introductions to this problem are given in [6, 7].

In this setting a controllable plant is distributed on several finite-state processes which interact asynchronously using shared actions. On every process, the local controller can choose to block some of the actions, called *controllable* actions, but it cannot block the *uncontrollable* actions from the environment. The choices of the local controllers are based on two sources of information.

- First the controller monitors the sequence of states and actions of the local process. This information is called the *local view* of the controller.
- Second when a shared action is played by several processes then all the controllers of these processes can exchange as much information as they want. In particular together they can compute their mutual view of the global execution: their *causal past*.

A controller is correct if it guarantees that every possible execution of the plant satisfies some specification. The controller synthesis problem is a decision problem which, given a plant as input, asks whether the system admits a correct

controller. In case such a controller exists, the algorithm should as well compute one.

The difficulty of controller synthesis depends on several factors, e.g.:

- the size and architecture (pipeline, ring, ...) of the system,
- the information available to the controllers,
- the specification.

Assuming that processes can exchange information upon synchronization and use their causal past to take decisions is one of the key aspects to get decidable synthesis problems [3]. In early work on distributed controller synthesis, for example in the setting of [9], the only source of information available to the controllers is their local view. In this setting, distributed synthesis is not decidable in general, except for very particular architectures like the pipeline architecture. The paper [2] proposes information forks as an uniform notion explaining the (un)decidability results in distributed synthesis. The idea of using causal past as a second source of information appeared in [3].

We adopt a modern terminology and call the plant a *distributed game* and the controllers are *distributed strategies* in this game. A distributed strategy is a function that maps the causal past of processes to a subset of controllable actions. In the present paper we focus on the *termination condition*, which is satisfied when each process is guaranteed to terminate its computation in finite time, in a final state. A distributed strategy is winning if it guarantees the termination condition, whatever uncontrollable actions are chosen by the environment.

We are interested in the following problem, whose decidability is an open question.

DISTRIBUTED SYNTHESIS PROBLEM: given a distributed game decide whether there exists a winning strategy.

There exists three classes of plants for which the DISTRIBUTED SYNTHESIS PROBLEM has been shown decidable:

1. when the dependency graph of actions is series-parallel [3],
2. when the processes are connectedly communicating [5],
3. and when the dependency graph of processes is a tree [4, 8].

A series-parallel game is a game such that the dependency graph (A, D) of the alphabet A is a co-graph. Series-parallel games were proved decidable in [3], for a different setup than ours: in the present paper we focus on process-based control while [3] was focusing on action-based control. Actually action-based control is more general than process-based control, see [6] for more details. The results of the present paper could probably be extended to action-based control however we prefer to stick to process-based control in order to keep the model intuitive. To our knowledge, the result of [3] was the first discovery of a class of asynchronous distributed system with causal memory for which the DISTRIBUTED SYNTHESIS PROBLEM is decidable

Connectedly communicating games have been introduced [5]. A game is connectedly communicating if there is a bound k such that if a process p executes

k steps in parallel of another process q then all further actions of p will be parallel to q. The event structure of a connectedly communicating games has a decidable MSO theory [5] which implies that the DISTRIBUTED SYNTHESIS PROBLEM is decidable for these games.

An acyclic game is a game where processes are arranged as a tree and actions are either local or synchronize a father and its son. Even in this simple setting the DISTRIBUTED SYNTHESIS PROBLEM is non-elementary hard [4].

Our contribution We develop a new proof technique to address the distributed controller synthesis problem, and provide a unified proof of decidability for series-paralell, connectedly communicating and acyclic games. We design a class of games, called *decomposable games*, for which the DISTRIBUTED SYNTHESIS PROBLEM is decidable. This leads to new examples of decidable architectures for controller synthesis.

Our proof technique consists in simplifying a winning strategy by looking for useless parts to be removed in order to get a smaller winning strategy. These parts are called *useless repetitions*. Whenever a useless repetition exists, we remove it using an operation called a *shortcut* in order to get a simpler strategy. Intuitively, a shortcut is a kind of cut-and-paste operation which makes the strategy smaller. By taking shortcuts again and again, we make the strategy smaller and smaller, until it does not have any useless repetition anymore.

If a winning strategy exists, there exists one with no useless repetition. In decomposable games, there is a computable upper bound on the size of strategies with no useless repetition, which leads to decidability of the controller synthesis problem.

Performing cut-and-paste in a distributed game is not as easy as doing it in a single-process game. In a single-process game, strategies are trees and one can cut a subtree from a node A and paste it to any other node B, and the operation makes sense as long as the state of the process in the same in both A and B. In the case of a general distributed strategy, designing cut-and-paste operations is more challenging. Such operations on the strategy tree should be consistent with the level of information of each process, in order to preserve the fundamental property of distributed strategies: the decisions taken by a process should depend only of its causal view, not on parallel events.

The decidability of series-parallel games established in [3] relies also on some simplification of the winning strategies, in order to get *uniform* strategies. The series-parallel assumption is used to guarantee that the result of the replacement of a part of a strategy by a uniform strategy is still a strategy, as long as the states of all processes coincide. Here we work without the series-parallel assumption, and matching the states is not sufficient for a cut-and-paste operation to be correct.

This is the reason for introducing the notion of *lock*. A lock is a part of a strategy where an information is guaranteed to spread in a team of processes before any of these processes synchronize with a process outside the team. When two locks A and B are similar, in some sense made precise in the paper, the lock B can be cut and paste on lock A. Upon arrival on A, a process of the team initiates a change of strategy, which progressively spreads across the team. All processes of the team should eventually play as if the play from A to B had already taken place, although it actually did not.

The complexity of our algorithm is really bad, so probably this work has no immediate practical applications. This is not surprising since the problem is non-elementary even for the class of acyclic games [4]. Nevertheless we think this paper sheds new light on the difficult open problem of distributed synthesis.

Parity games The winning condition of the present paper is the termination of all processes in a final state. Richer specifications can be expressed by parity conditions. In the present paper we stick to termination conditions for two reasons.

- The long-term goal of this research is to establish the decidability or undecidability of the distributed controller synthesis problem. A possible first step is to prove decidability for games with termination conditions.
- It seems that the results of the present paper can be lifted to parity games, using the same concepts (locks and shortcuts) but at the cost of some extra technical details: even for safety games one needs to consider maximal infinite play and finite-memory strategies.

The paper is organized as follows. Section 2 introduces the DISTRIBUTED SYNTHESIS PROBLEM. Section 3 provides several examples. In section 4 we show how to simplify strategies which contains useless repetitions, and proof that if a winning strategy exists, there exists one without any useless repetition. Finally, section 5 introduces the class of decomposable games and show their controller synthesis problem is decidable. Missing proofs can be found in the appendix.

2 Definitions and basic properties

Mazurkiewicz traces The theory of Mazurkiewicz traces is very rich and extensively developed in [1]. Here we only fix notations and recall the notions of traces, prime traces and views, and list a few elementary properties of traces that we will use throughout the paper.

We fix an alphabet A and a symmetric and reflexive dependency relation $D \subseteq A \times A$ and the corresponding independency relation $\mathbb{I} \subseteq A \times A$ defined as $\forall a, b \in A, (a \mathbb{I} b) \iff (a, b) \notin D$. A *Mazurkiewicz trace* or, more simply, a *trace*, is an equivalence class for the smallest equivalence relation $\equiv$ on A^* which commutes independent letters i.e. for every letters a, b and every words x, y,

$$a \mathbb{I} b \implies xaby \equiv xbay \ .$$

The words in the equivalence class are the *linearizations* of the trace. The trace whose only linearization is the empty word is denoted ϵ. All linearizations of a trace u have the same set of letters and length, denoted respectively $\mathrm{Alph}(u)$ and $|u|$. Given $B \subseteq A$, the set of traces such that $\mathrm{Alph}(u) \subseteq B$ is denoted $B^*_{\equiv}$ in particular the set of all traces is $A^*_{\equiv}$.

The concatenation on words naturally extends to traces. Given two traces $u, v \in A^*_{\equiv}$, the trace uv is the equivalence class of any word in uv. The prefix relation $\sqsubseteq$ is defined by:

$$(u \sqsubseteq v \iff \exists w \in A^*_{\equiv}, uw = v) \ .$$

Prime traces and parallel traces A trace $u \in A^*_{\equiv}$ is *prime* if all its linearizations have the same last letter, denoted

$$\mathrm{last}(u)$$

and called the last letter of u. Two prime traces u and v are said to be *parallel* if

- neither u is a prefix of v nor v is a prefix of u,
- there is a trace w such that both u and v are prefixes of w.

Processes and automata Asynchronous automata are to traces what finite automata are to finite words, as witnessed by Zielonka's theorem [11]. An asynchronous automaton is a collection of automata on finite words, whose transition tables do synchronize on certain actions.

Definition 1. *An asynchronous automaton on alphabet A with processes $\mathbb{P}$ is a tuple $\mathcal{A} = ((A_p)_{p\in\mathbb{P}}, (Q_p)_{p\in\mathbb{P}}, (i_p)_{p\in\mathbb{P}}, (F_p)_{p\in\mathbb{P}}, \Delta)$ where:*

- *every process $p \in \mathbb{P}$ has a set of actions A_p, a set of states Q_p and $i_p \in Q_p$ is the initial state of p and $F_p \subseteq Q_p$ its set of final states.*
- *$A = \bigcup_{p\in\mathbb{P}} A_p$. For every letter $a \in A$, the domain of a is*

$$\mathrm{dom}(a) = \{p \in \mathbb{P} \mid a \in A_p\} \ .$$

- *Δ is a set of transitions of the form $(a, (q_p, q'_p)_{p\in\mathrm{dom}(a)})$ where $a \in A$ and $q_p, q'_p \in Q_p$. Transitions are* deterministic*: for every $a \in A$, if $(a, (q_p, q'_p)_{p\in\mathrm{dom}(a)}) \in \Delta$ and $(a, (q_p, q''_p)_{p\in\mathrm{dom}(a)}) \in \Delta$ then $\forall p \in \mathrm{dom}(a), q'_p = q''_p$.*

Such an automaton works asynchronously: each time a letter a is processed, the states of the processes in $\mathrm{dom}(a)$ are updated according to the corresponding transition, while the states of other processes do not change.

This induces a natural commutation relation $\mathbb{I}$ on A: two letters commute iff they have no process in common i.e.

$$a \mathbb{I} b \iff \mathrm{dom}(a) \cap \mathrm{dom}(b) = \emptyset \ .$$

The set of *plays* of the automaton $\mathcal{A}$ is a set of traces denoted $\mathrm{plays}(\mathcal{A})$ and defined inductively, along with $\mathrm{state} : \mathrm{plays}(\mathcal{A}) \to \Pi_{p\in\mathbb{P}} Q_p$.

- ϵ is a play and $\mathrm{state}(\epsilon) = (i_p)_{p\in\mathbb{P}}$,
- for every play u if there exists a transition $(a, (\mathrm{state}_p(u), q_p)_{p\in\mathrm{dom}(a)}) \in \Delta$ then ua is a play and $\forall p \in \mathbb{P}, \mathrm{state}_p(ua) = \begin{cases} \mathrm{state}_p(u) & \text{if } p \notin \mathrm{dom}(a), \\ q_p & \text{otherwise.} \end{cases}$

For every play u, $\mathrm{state}(u)$ is called the *global state* of u. The definition of global states given above is for words. It extends to traces because the global state is invariant by commutation of independent letters. Actually, $\mathrm{state}_p(u) = \mathrm{state}_p(\partial_p(u))$.

Counting actions of a process For every trace u we can count how many times a process p has played an action in u, which we denote $|u|_p$. Formally, $|u|_p$ is first defined for words, as the length of the projection of u on A_p, which is invariant by commuting letters. The domain of a trace is defined as

$$\mathrm{dom}(u) = \{p \in \mathbb{P} \mid |u|_p > 0\} \quad .$$

Views, strategies and games Given an automaton $\mathcal{A}$, we want the processes to choose actions which guarantee that every play eventually terminates in a final state.

To take into account the fact that some actions are controllable by processes while some other actions are not, we assume that A is partitioned in

$$A = A_c \sqcup A_e$$

where A_c is the set of controllable actions and A_e the set of (uncontrollable) environment actions. Intuitively, processes cannot prevent their environment to play actions in Ae, while they can block any action in A_c.

We adopt a modern terminology and call the automaton $\mathcal{A}$ together with the partition $A = A_c \sqcup A_e$ a *distributed game*, or even more simply a *game*.

In this game the processes play distributed strategies, which are individual plans of action for each process. The choice of actions by a process p is dynamic: at every step, p chooses a new set of controllable actions, depending on its information about the way the play is going on. This information is limited because processes cannot communicate together unless they synchronize on a common action. In this case they exchange as much information about the play as they want. Finally, the information missing to a process is the set of actions which happened in parallel of its own actions. The information which remains is called the p-view of the play, and is defined formally as follows.

Definition 2 (View of a process)**.** *For every process p and trace u, the* p-view *of u, denoted $\partial_p(u)$, is the unique prime trace such that*

- *u factorizes as $u = \partial_p(u) \cdot v$ with $p \notin \mathrm{dom}(v)$.*
- *p belongs to the domain of the last letter of $\partial_p(u)$; and*

We can now define what is a distributed strategy.

Definition 3 (Distributed strategies, consistent and maximal plays)**.** *Let $G = (\mathcal{A}, A_c, A_e)$ be a distributed game. A* strategy for process p *in G is a mapping which associates with every trace u a set of actions $\sigma_p(u)$ such that:*

- *environment actions are allowed: $A_e \subseteq \sigma_p(u)$,*
- *the decision depends only on the view of the process: $\sigma_p(u) = \sigma_p(\partial_p(u))$.*

A distributed strategy *is a tuple $\sigma = (\sigma_p)_{p \in \mathbb{P}}$ where each σ_p is a strategy of process p. A play $u = a_1 \cdots a_{|u|} \in \mathrm{plays}(\mathcal{A})$ is* consistent with σ*, or equivalently is a* σ-play *if:*

$$\forall i \in 1 \ldots |u|, \forall p \in \mathrm{dom}(a_i), a_i \in \sigma_p(a_1 \cdots a_{i-1}) \quad .$$

A σ-play is maximal *if it is not the strict prefix of another σ-play.*

Note that a strategy is forced to allow every environment action to be executed at every moment. This may seem to be a huge strategic advantage for the environment. However depending on the current state, not every action can be effectively used in a transition because the transition function is not assumed to be total. So in general not every environment actions can actually occur in a play. In particular it may happen that a process enters a final state with no outgoing transition, where no further action is possible.

Winning games Our goal is to synthesize strategies which ensure that the game terminates and all processes are in a final state.

Definition 4 (Winning strategy)**.** *A strategy σ is winning if the set of σ-plays is finite and in every maximal σ-play u, every process is in a final state i.e.* $\forall p \in \mathbb{P}, \text{state}_p(u) \in F_p$.

We are interested in the following problem, whose decidability is an open question.

Distributed synthesis problem: given a distributed game decide whether there exists a winning strategy.

If the answer to the problem is positive, the algorithm should as well compute a correct distributed controller.

3 Three examples

Series-parallel games A game is *series-parallel* if its dependency alphabet (A, D) is a co-graph i.e. it belongs to the smallest class of graphs containing singletons and closed under parallel product and complementation. In this case A has a *decomposition tree*, this is a binary tree whose nodes are subsets of A, its leaves are the singletons $(\{a\})_{a\in A}$, its root is A. Moreover every node B with two children B_0 and B_1 is the disjoint union of B_0 and B_1 and either $B_0 \times B_1 \subseteq D$ (serial product) or $(B_0 \times B_1) \cap D = \emptyset$ (parallel product).

The synthesis problem for series-parallel games is decidable [3].

Connectedly communicating games A game is *k-connectedly communicating* if for every pair p, q of processes, if process p plays k times in parallel of process q then all future actions of q will be parallel to p. Formally, for every prime play uvw,

$$(q \notin \text{dom}(v) \text{ and } |v|_p \geq k) \implies q \notin \text{dom}(w) \ .$$

The MSO theory of the event structure of a k-connectedly communicating games is decidable [5], which implies that controller synthesis is decidable for theses games.

Acyclic games An acyclic game is a game where processes $\mathbb{P}$ are arranged as a tree $T_{\mathbb{P}}$ and the domain of every action is a sconnected set of nodes of $T_{\mathbb{P}}$. The synthesis problem is decidable for acyclic games where the domain of each action has size ≤ 2 [4].

4 Simplifying strategies

In this section we present an elementary operation called a *shortcut*, which can be used to simplify and reduce the duration of a winning strategy.

To create a shortcut, one selects a σ-play xy and modify the strategy σ so that as soon as any of the processes sees the play x in its causal past, this process assumes that not only x but also xy has actually occurred. In other words, a shortcut is a kind of *cut-and-paste* in the strategy: we glue on node x the substrategy rooted at node xy.

The choice of x and y should be carefully performed so that the result of the shortcut is still a strategy. We provide a sufficient condition for that: (x, y) should be a *useless repetition*.

The interest of taking shortcuts is the following: if the original strategy is winning, then the strategy obtained by taking the shortcut is winning as well, and strictly smaller than the original one.

In the remainder of this section, we formalize these concepts.

4.1 Locks

In order to simplify strategies, we need to limit the communication between a set of processes, called a team, and processes outside the team. This leads to the notion of a $\mathbb{Q}$-lock: this is a play u such that there is no information transfer between $\mathbb{Q}$ and $\mathbb{P} \setminus \mathbb{Q}$ in parallel of u.

Definition 5 ($\mathbb{Q}$-lock)**.** *Let $\mathbb{Q} \subseteq \mathbb{P}$. An action b is $\mathbb{Q}$-safe if* ($\mathrm{dom}(b) \subseteq \mathbb{Q}$ *or* $\mathrm{dom}(b) \cap \mathbb{Q} = \emptyset$)*. A prime play u is a* $\mathbb{Q}$-lock *if the last action of every prime play parallel to u is $\mathbb{Q}$-safe.*

Locks can arise in a variety of situations, including the three classes of games presented in the examples section.

Lemma 1 (Sufficient conditions for $\mathbb{Q}$-locks)**.** *Let u be a prime play and $\mathbb{Q} \subseteq \mathbb{P}$. Each of the following conditions is sufficient for u to be a $\mathbb{Q}$-lock:*

i) $\mathbb{Q} = \mathbb{P}$.

ii) u is a $(\mathbb{P} \setminus \mathbb{Q})$-lock.

iii) $\mathbb{Q} \subseteq \mathrm{dom}(\mathrm{last}(u))$.

iv) The game is series-parallel and $\mathbb{Q} = \mathrm{dom}(B)$ *where B is the smallest node of the decomposition tree of A which contains* $\mathrm{Alph}(u)$.

v) The game is connectedly communicating game with bound k, $\mathbb{Q} = \mathrm{dom}(u)$ *and* $\forall p \in dom(u), |u|_p \geq k$.

vi) The game is acyclic with respect to a tree $T_{\mathbb{P}}$ and $\mathbb{Q}$ is the set of descendants in $T_{\mathbb{P}}$ of the processes in $\mathrm{dom}(\mathrm{last}(u))$.

Proof. Let v be a play parallel to u and denote $b = \mathrm{last}(u)$ and $c = \mathrm{last}(v)$, thus $c \mathbb{I} b$.

Condition i) is sufficient because every letter is $\mathbb{P}$-safe. Condition ii) is sufficient because an action is $\mathbb{Q}$-safe iff it is $\mathbb{P} \setminus \mathbb{Q}$-safe. Condition iii) is sufficient because by hypothesis $c \mathbb{I} b$thus $\mathrm{dom}(c) \cap \mathbb{Q} \subseteq \mathrm{dom}(c) \cap \mathrm{dom}(b) = \emptyset$.

For series-parallel games (condition iv)) we distinguish between two cases. If B is a leaf of the decomposition tree then it is a singleton $\{a\}$ thus $\mathrm{dom}(\mathbb{Q}) = \mathrm{dom}(u)$ and we apply iii). Otherwise B has two sons B_0 and B_1 and B is either the parallel or serial product of these two nodes. However, by minimality of B, u has letters both in B_0 and B_1 and since u is prime only the serial product case can occur i.e. $\forall b_0 \in B_0, b_1 \in B_1, \mathrm{dom}(b_0) \cap \mathrm{dom}(b_1) \neq \emptyset$. Without loss of generality assume $b \in B_0$. In case $c \in B$ then $\mathrm{dom}(c) \subseteq \mathrm{dom}(B) = \mathbb{Q}$ thus c is $\mathbb{Q}$-safe. In case $c \notin B$ then let C be the smallest node containing both B and $\{c\}$. Since v is parallel to u then $\mathrm{dom}(b) \cap \mathrm{dom}(c) = \emptyset$ thus C is a parallel node with two children C_0 and C_1 such that $B \subseteq C_0$ while $c \in C_1$. Thus $\mathrm{dom}(c) \cap \mathrm{dom}(B) = \emptyset$ and c is $\mathbb{Q}$-safe.

For connectedly communicating games (v), the hypothesis $(\forall p \in \mathrm{dom}(u), |u|_p \geq k)$ implies that for every prime play uv, $\mathrm{dom}(v) \subseteq \mathrm{dom}(u)$. Thus all actions played in v are $\mathrm{dom}(u)$-safe.

For acyclic games (vi), the domain of every action is connected in $T_{\mathbb{P}}$ thus $\mathrm{dom}(b)$ has a maximum process p in $T_{\mathbb{P}}$ and $\mathbb{Q}$ is the set of descendants of p. Since $\mathrm{dom}(c)$ is connected as well in $T_{\mathbb{P}}$ then $(\mathrm{dom}(c) \cap \mathbb{Q} \neq \emptyset \wedge \mathrm{dom}(c) \cap (\mathbb{P} \setminus \mathbb{Q}) \neq \emptyset) \implies (p \in \mathrm{dom}(c)) \implies \neg(c \mathbb{I} b)$. □

4.2 Taking shortcuts

In this section we present a basic operation used to simplify a strategy, called a *shortcut*, which consists in modifying certain parts of a strategy, called *useless repetitions*. These notions rely on the notion of $\mathbb{Q}$-*state* as well as two operations on strategies called *shifting* and *projection*.

Definition 6 (Shifting and projection of a strategy)**.** *Let σ be a strategy and u a σ-play. The* shift *of σ by u is the distributed strategy defined by*

$$\sigma[u](v) = \sigma(uv) \ .$$

Let $\mathbb{Q} \subseteq \mathbb{P}$. The projection *of σ on $\mathbb{Q}$ is:*

$$\pi_{\mathbb{Q}}(\sigma) = \{v \in A^*_{\equiv} \mid \ v \text{ is a } \sigma\text{-play and } \mathrm{dom}(v) \subseteq \mathbb{Q}\} \ .$$

After a play, the global state in $\Pi_{p \in \mathbb{P}} Q_p$ gives an instantaneous picture of the game but does not picture the relative knowledge of the processes which is required in order to define useless repetitions. For that we need the notion of strategic state:

Definition 7 (Strategic state)**.** *For every action $b \in A$ we denote*

$$A_{\mathbb{Q}||b} = \{a \in A \mid \mathrm{dom}(a) \subseteq \mathbb{Q} \text{ and } \mathrm{dom}(a) \cap \mathrm{dom}(b) = \emptyset\}$$
$$\mathbb{Q}_{?b} = \mathrm{dom}\left(A_{\mathbb{Q}||b}\right) \ .$$

Let σ be a strategy, u be a prime σ-play and $\mathbb{Q}$ a team of processes. The strategic $\mathbb{Q}$-state of σ after u is the tuple

$$\mathrm{strate}_{\sigma,\mathbb{Q}}(u) = \left(\mathrm{last}(u), \left(\mathrm{state}_p(u)\right)_{p \in \mathbb{P}}, \left(\sigma_p(u)\right)_{p \in \mathbb{Q}}, \sigma'\right) \in A \times (\Pi_{p \in \mathbb{P}} Q_p) \times 2^A \times 2^{\mathbb{Q}} \times 2^{A^*_{\equiv}} \ ,$$

where σ' is the projection of $\sigma[u]$ on $\mathbb{Q}_{?\,\mathrm{last}(u)}$.

Definition 8 (Useless repetition)**.** *A useless $\mathbb{Q}$-repetition in a strategy σ is a pair of traces (x, y) such that xy is a σ-play, y is not empty,* $\mathrm{dom}(y) \subseteq \mathbb{Q}$, $\mathrm{strate}_{\sigma,\mathbb{Q}}(x) = \mathrm{strate}_{\sigma,\mathbb{Q}}(xy)$ *and both x and xy are $\mathbb{Q}$-locks.*

The following theorem is the key to our decidability results.

Theorem 1. *If there exists a winning strategy then there exists a winning strategy without any useless repetition.*

The proof of this theorem relies on the notion of shortcuts, an operation which turns a winning strategy into another strategy with strictly shorter duration.

Definition 9 (Duration of a strategy)**.** *The duration of a strategy σ is*

$$\mathrm{dur}(\sigma) = \sum_{u \text{ maximal } \sigma\text{-play}} |u| \ .$$

The duration of a strategy σ maybe infinite in general but is finite if σ is winning.

Lemma 2 (Shortcuts)**.** *Let (x, y) be a useless $\mathbb{Q}$-repetition in a strategy σ. Then the mapping τ from traces to traces defined by*

$$\tau : u \mapsto \begin{cases} \sigma(u) & \text{if } x \not\sqsubseteq u \\ \sigma(xyu') & \text{if } u = xu' \end{cases}$$

is a distributed strategy called the (x, y)-shortcut of σ*. Moreover for every trace z,*

$$(xz \text{ is a } \tau\text{-play}) \iff (xyz \text{ is a } \sigma\text{-play}) \ . \tag{1}$$

If σ is a winning strategy then τ is winning as well and has a strictly smaller duration.

The full proof can be found in appendix.

Sketch of proof of Lemma 2. To prove that τ is a distributed strategy, we should prove that for every process p and play u, $\tau_p(u) = \tau_p(\partial_p(u))$. There are several cases:

1. x has not occurred ($x \not\sqsubseteq u$),
2. x has occurred in parallel of the process ($x \sqsubseteq u \wedge x \not\sqsubseteq \partial_p(u)$),
3. x has occurred in the causal past of the process ($x \sqsubseteq \partial_p(u)$).

It may happen that $x \sqsubseteq u$ and there exists a process p_2 in case 2 and a process p_3 in case 3. Then process p_3 is playing the modified strategy $xz \to \sigma(xyz)$ while process p_2 is still playing the original strategy σ, which may *a priori* creates some plays unrelated with the original strategy σ. The match of the strategic states in x and xy ensures that the equivalence (1) stays valid.

Moreover, thanks to (1), $\mathrm{dur}(\sigma) < \infty$ implies $\mathrm{dur}(\tau) < \mathrm{dur}(\sigma)$ because y is not empty. And according to (1) again, the set of global states of maximal plays is the same for σ and τ thus if σ is winning then τ is winning as well. □

Proof of Theorem 1. As long as there exists a useless repetition, take the corresponding shortcut. According to Lemma 2, this creates a sequence $\sigma_0, \sigma_1, \ldots$ of winning strategies whose duration strictly decreases. Thus the sequence is finite and its last element is a winning strategy without useless repetition. □

5 Decomposable games

In this section we introduce decomposable games, for which the DISTRIBUTED SYNTHESIS PROBLEM is decidable. We give two definitions of decomposable games: process-decomposability and the more general action-decomposability. We show that connectedly-communicating games and acyclic games are process-decomposable while series-parallel games are action-decomposable. We provide extra examples of decidable games.

Process-decomposability The definition of process-decomposable games relies on a preorder $\preceq$ on $2^{\mathbb{P}}$ (i.e. a reflexive and transitive relation) which is monotonic with respect to inclusion, i.e. for every $\mathbb{Q}, \mathbb{Q}' \subseteq \mathbb{P}$,

$$\mathbb{Q} \subseteq \mathbb{Q}' \implies \mathbb{Q} \preceq \mathbb{Q}' \ .$$

We denote $\prec$ the strict version: $\mathbb{Q} \prec \mathbb{Q}' \iff (\mathbb{Q} \preceq \mathbb{Q}') \wedge \neg(\mathbb{Q}' \preceq \mathbb{Q})$.

Definition 10 (Process-decomposable games)**.** *Let k be an integer. A game is* $(\preceq, k)$-decomposable *if for every prime play xy satisfying*

$$\forall p \in \mathrm{dom}(y), |y|_p \geq k$$

there exists a superset $\mathbb{Q} \supseteq \mathrm{dom}(y)$ and a prefix $xz \sqsubseteq xy$ which is a $\mathbb{Q}$-lock and such that

$$(\mathbb{Q} \setminus \mathrm{dom}(\mathrm{last}(xz))) \prec \mathrm{dom}(y) \ .$$

We have already seen two examples of process-decomposable games.

Lemma 3. *Connectedly communicating games and acyclic games are process-decomposable.*

Proof. Let xy be a prime play such that $\forall p \in \mathrm{dom}(y), |y|_p \geq k$.

Assume the game is k-connectedly communicating. Let $\preceq$ be the inclusion preorder. Since $(\forall p \in \mathrm{dom}(y), |y|_p \geq k)$ then we choose $\mathbb{Q} = \mathrm{dom}(y)$ and $xz = xy$. Then xy is a $\mathbb{Q}$-lock according to Lemma 1. And since $\mathrm{dom}(\mathrm{last}(xz)) \subseteq \mathrm{dom}(y)$ then $(\mathbb{Q} \setminus \mathrm{dom}(y\,\mathrm{last}(xz))) \subsetneq \mathbb{Q}$.

Assume the game is acyclic with process tree $T_{\mathbb{P}}$. Set $k = 1$ and $\mathbb{Q} \preceq \mathbb{Q}'$ iff every process in $\mathbb{Q}$ has a $T_{\mathbb{P}}$-ancestor in $\mathbb{Q}'$. Let $p \in \mathbb{P}$ be the least common ancestor of processes in $\mathrm{dom}(y)$ and $\mathbb{Q}$ the set of descendants of p. Since y is prime and since the domain of every action is a connected subset of $T_{\mathbb{P}}$ then $\mathrm{dom}(y)$ is connected as well thus $p \in \mathrm{dom}(y)$. Let xz be a prime prefix of xy such that $p \in \mathrm{dom}(\mathrm{last}(xz))$. Then xz is a $\mathbb{Q}$-lock according to iv) in Lemma 1. And since $p \in \mathrm{dom}(\mathrm{last}(xz))$ then $(\mathbb{Q} \setminus \mathrm{dom}(\mathrm{last}(xz))) \preceq \mathbb{Q} \setminus \{p\} \prec \mathbb{Q} \preceq \mathrm{dom}(y)$. □

Action-decomposability Action-decomposability generalizes process-decomposability. Fix a preorder $\preceq$ on 2^A which is monotonic with respect to inclusion and denote $\prec$ its strict version.

Definition 11 (Action-decomposable games)**.** *Let k be an integer. A game is* $(\preceq, k)$-decomposable *if for every prime play xy satisfying*

$$\forall p \in \mathrm{dom}(y), |y|_p \geq k$$

there exists a superset $\mathbb{Q} \supseteq \mathrm{dom}(y)$ and a $\mathbb{Q}$-lock $xz \sqsubseteq xy$ such that

$$\{a \in A \mid \mathrm{dom}(a) \subseteq \mathbb{Q} \text{ and } a \mathbb{I} \ \mathrm{last}(xz)\} \prec \mathrm{Alph}(y) \ .$$

Lemma 4. *Every process-decomposable game is action-decomposable.*

Proof. Every preorder $\preceq_{\mathbb{P}}$ on $2^{\mathbb{P}}$ induces a preorder $\preceq_A$ on 2^A defined by $B \preceq_A B' \iff \mathrm{dom}(B) \preceq_{\mathbb{P}} \mathrm{dom}(B')$. Then every $\preceq_{\mathbb{P}}$-decomposable game is $\preceq_A$-decomposable because, for every action b, $\mathrm{dom}\left(\{a \in A \mid \mathrm{dom}(a) \subseteq \mathbb{Q} \text{ and } a \mathbb{I} b\}\right) \subseteq (\mathbb{Q} \setminus \mathrm{dom}(b))$. □

Lemma 5. *Series-parallel games are action-decomposable.*

Proof. Let T_A be the decomposition tree of A. For every non-empty subset $B \subseteq A$ the set of nodes containing B is a branch of T_A whose smallest node is denoted $B_\uparrow$. Denote $\emptyset_\uparrow = \emptyset$. The order $\preceq$ on 2^A is defined as $B \preceq B' \iff B_\uparrow \subseteq B'_\uparrow$.

For every prime play xy, we show that the definition of $\preceq$-decomposable games is satisfied with $\mathbb{Q} = \mathrm{dom}(\mathrm{Alph}(y)_\uparrow)$ and $xz = xy$. According to Lemma 1, xy is a $\mathbb{Q}$-lock. Let $b = \mathrm{last}(xy)$ and $A_{\mathbb{Q}||b} = \{a \in A \mid \mathrm{dom}(a) \subseteq \mathbb{Q} \text{ and } a \mathbb{I} b\}$, we show that $A_{\mathbb{Q}||b} \prec \mathrm{Alph}(y)$.

Since y is prime then $\mathrm{Alph}(y)$ is a connected subset of the dependency graph of the alphabet thus $\mathrm{Alph}(y)_\uparrow$ is either the singleton $\{b\}$ or a serial product node with two sons B and C. In the first case, $\mathbb{Q} = \mathrm{dom}(b)$ thus $A_{\mathbb{Q}||b} = \emptyset \prec \mathrm{Alph}(y)$. In the second case w.l.o.g. assume that $b \in B$. Then no action of C is independent of b thus $A_{\mathbb{Q}||b} \subseteq B$. Then $B_\uparrow = B \subsetneq \mathrm{Alph}(y) \preceq \mathrm{Alph}(y)_\uparrow$ thus $B \prec \mathrm{Alph}(y)$ hence $A_{\mathbb{Q}||b} \prec \mathrm{Alph}(y)$. □

5.1 Decidability

In this section we show that decomposability is a decidable property and decomposable games have a decidable controller synthesis problem.

Lemma 6 (Decomposability is decidable)**.** *Whether a game is decomposable is decidable. There exists a computable function* decomp *from games to integers such that whenever a game G is $(\preceq, k)$-decomposable for some k, it is $(\preceq, \mathrm{decomp}(G))$-decomposable.*

The proof is elementary and can be found in appendix.

Theorem 2. *The distributed control problem is decidable for decomposable games.*

Proof of Theorem 2. We show that there exists a computable function f from games to integers such that in every decomposable distributed game G every strategy with no useless repetition has duration $\leq f(G)$.

Let G be a decomposable distributed game. Then according to Lemma 6, the game is $(\preceq, \mathrm{decomp}(G))$-decomposable for some preorder $\preceq$ on 2^A. Set $k = \mathrm{decomp}(G)$. For every set of actions B, denote G_B the game restricted to actions in B. Denote $R_B(m)$ the largest size of a complete graph whose edges are labelled with 2^B and which contains no complete monochromatic subgraph of size $\geq m$. According to Ramsey theorem, $R_B(m)$ is finite and computable. Then $G = G_A$ and for every $B \subseteq A$, $f(G_B)$ is defined inductively by:

$$f(G_B) = R_B\left(k \cdot |B| \cdot |Q|^{|\mathbb{P}|} \cdot 2^{|B||\mathbb{P}|} \cdot 2^{|B|^{\max_{B' \prec B} f(G_{B'})}}\right) \ ,$$

with the convention $\max \emptyset = 0$, thus $f(G_\emptyset) = 0$.

Let σ be a strategy with no useless repetition. We prove that for every prime σ-play u,

$$|u| \leq f\left(G_{\mathrm{Alph}(u)}\right) \quad . \tag{2}$$

The proof is by induction on $\mathrm{Alph}(u)$ with respect to $\preceq$. The base case when $\mathrm{Alph}(u) = \emptyset$ is easy, in this case $|u| = 0$.

Now assume inductively that for every prime play u', if $\mathrm{Alph}(u') \prec \mathrm{Alph}(u)$ then $|u'| \leq f(G_{\mathrm{Alph}(u')})$. We start with computing, for every $B \prec \mathrm{Alph}(u)$, an upper bound on the length of every factorization $u = u_0 u_1 \cdots u_N u_{N+1}$ such that

$$B = \mathrm{Alph}(u_1) = \mathrm{Alph}(u_2) = \ldots = \mathrm{Alph}(u_N) \quad .$$

Let $u = a_1 a_2 \cdots a_{|u|}$ and for every $1 \leq i \leq j \leq |u|$ denote $u[i,j] = a_i a_{i+1} \cdots a_j$. Let $1 < i_1 < \ldots < i_N \leq |u| + 1$ the indices such that for every $\ell \in 1 \ldots N$, $u_\ell = u[i_\ell, i_{\ell+1} - 1]$. For $1 \leq \ell < \frac{N}{k}$, denote w_ℓ the concatenation $w_\ell = u_{k\cdot\ell} u_{k\cdot\ell+1} \cdots u_{k\cdot\ell+k-1} = u[i_{k\cdot\ell}, i_{k\cdot\ell+k} - 1]$.

Let $1 \leq \ell < \frac{N}{k}$. Then $u_0 w_1 \cdots w_\ell \sqsubseteq u$ is a play and for every process $p \in \mathrm{dom}(B), |w_\ell|_p \geq k$. Since the game is decomposable with witness k, there exists a superset $\mathbb{T}^{(\ell)}$ of $\mathrm{dom}(B)$, an action b_ℓ, and a prime prefix $w'_\ell b_\ell \sqsubseteq w_\ell$ such that $w_0 w_1 \ldots w'_\ell b_\ell$ is a $\mathbb{T}^{(\ell)}$-lock and

$$B_\ell \prec B \text{ where } B_\ell = A_{\mathbb{T}^{(\ell)}||b_\ell} = \{a \in A \mid \mathrm{dom}(a) \subseteq \mathbb{T}^{(\ell)} \text{ and } a \mathbb{I} b_\ell\} \quad .$$

Denote $z_\ell = w_0 w_1 \ldots w'_\ell b_\ell$ and $\mathrm{strate}_\ell = \left(b_\ell, (s_{\ell,p})_{p\in\mathbb{P}}, (\sigma_p(z_\ell))_{p\in\mathbb{T}^{(\ell)}}, \sigma_\ell\right)$ the $\mathbb{T}^{(\ell)}$-state of σ after z_ℓ. By definition of $\mathbb{T}^{(\ell)}$-states, σ_ℓ is the projection of $\sigma[z_\ell]$ on $\mathbb{T}^{(\ell)}_{?b_\ell}$. Moreover $B_\ell = A_{\mathbb{T}^{(\ell)}||b_\ell}$ thus $\sigma_\ell \subseteq 2^{B_l}$. Since $\preceq$ is monotonic with respect to inclusion, $B_\ell \prec B \preceq \mathrm{Alph}(u)$. Since $\sigma_\ell \subseteq 2^{B_l}$, we can apply the induction hypothesis to every play consistent with σ_ℓ. This shows $\sigma_\ell \subseteq A^{|f(G_{B_\ell})|}$. Let $m = \max_{B' \prec \mathrm{Alph}(u)} f(G_{B'})$ then $\sigma_\ell \subseteq A^m$. Since there is no useless repetition in σ, all the strategic states in $(\mathrm{state}_\ell)_{\ell \in 1 \ldots \frac{N}{k}}$ are different. We obtain the upper bound: $\frac{N}{k} \leq K$ with

$$K = |B| \cdot |Q|^{|\mathbb{P}|} \cdot 2^{|B||\mathbb{P}|} \cdot 2^{|B|^m} \quad .$$

Let J_u be the graph with vertices $1, \ldots, |u|$ and the label of the edge $\{i < j\}$ is $\{a_i, \ldots, a_j\}$. We proved that every monochromatic clique of J_u has size $\leq k \cdot K$. Thus according to Ramsey theorem , $|u| \leq R_{\mathbb{T}}(k \cdot K)$ which terminates the inductive step.

Finally, winning strategies in G can be looked for in the finite family of strategies of duration $\leq f(G)$ with $f(G)$ computable. Enumerating all these strategies and testing whether one of them is winning is easy, which provide an algorithm to solve the DISTRIBUTED SYNTHESIS PROBLEM in time $\mathcal{O}\left(2^{|A|^{f(G)}}\right)$. □

5.2 New examples of decidable games

The three classes of games whose decidability is already known are decomposable (cf Lemmas 3 and 5), in this section we provide new decidable examples.

Structurally decomposable games The notion of decomposability can be strengthen in order to depend only on the structure of the game, i.e. the actions and their domains, independently of states and transitions.

Definition 12 (Structural decomposability)**.** *A tuple* $(\mathbb{P}, A, (A_p)_{p\in\mathbb{P}})$ *where* $(A_p)_{p\in\mathbb{P}}$ *is a partition of* A *is* structurally decomposable *if there exists a pre-order* $\preceq$ *on* $2^{\mathbb{P}}$ *with the following property: for every prime trace* $y \in A^*$ *there is a a superset* $\mathbb{Q} \supseteq \mathrm{dom}(y)$ *and a letter* $b \in \mathrm{Alph}(y)$ *such that* $\mathbb{Q} \setminus \mathrm{dom}(b) \prec \mathbb{Q}$ *and every letter* a *with* $a \mathbb{I} b$ *is* $\mathbb{Q}$*-safe.*

By extension a game if structurally decomposable if its alphabet and processes are. Clearly if a game is structurally decomposable it is process-decomposable.

Lemma 7. *Four players games are structurally decomposable.*

Proof. Assume $|\mathbb{P}| = 4$, we show that the game is structurally decomposable for the pre-order $\preceq$ on $2^{\mathbb{P}}$ defined by: $\mathbb{Q} \preceq \mathbb{Q}' \iff |\mathbb{Q}| \leq |\mathbb{Q}'|$. Let y be a prime trace. We set

$$\mathbb{Q} = \begin{cases} \mathrm{dom}(y) & \text{if } |\mathrm{dom}(y)| \leq 2 \\ \mathbb{P} & \text{if } |\mathrm{dom}(y)| \geq 3 \end{cases}$$

If $\mathrm{dom}(y)$ contains a single process $\{p\}$ then we set $b = \mathrm{last}(y)$. Then $\mathbb{Q} = \{p\}$ thus $\{p\} \setminus \mathrm{dom}(b) = \emptyset \prec \{p\}$ and every letter $a \mathbb{I} b$ satisfies $p \notin \mathrm{dom}(a)$ thus is $\{p\}$-safe.

If $|\mathrm{dom}(y)| \geq 2$ then, since y is prime, y contains a letter b such that $|\mathrm{dom}(b)| \geq 2$. Then $\mathbb{Q} \setminus \mathrm{dom}(b) \prec \mathrm{dom}(y)$: in case $|\mathrm{dom}(y)| = 2$ then $\mathbb{Q} \setminus \mathrm{dom}(b) = \emptyset$ and otherwise $|\mathrm{dom}(y)| \geq 3$ and $|\mathbb{Q} \setminus \mathrm{dom}(b)| \leq 2$. And every letter $a \mathbb{I} b$ is $\mathbb{Q}$-safe, if $\mathbb{Q} = \mathbb{P}$ this is obvious and otherwise $\mathbb{Q} = \mathrm{dom}(b)$ thus $\mathrm{dom}(a) \cap \mathbb{Q} = \emptyset$. □

Another example are *majority games*:

Lemma 8 (Majority games)**.** *Assume that every non-local action synchronizes a majority of the processes i.e. for every action* a, $|\mathrm{dom}(a)| = 1$ *or* $|\mathrm{dom}(a)| \geq |\mathbb{P} \setminus \mathrm{dom}(a)|$. *Then the game is structurally decomposable.*

Combining structurally decomposable games The class of uniformly decomposable games is stable under projection and merge.

Definition 13 (Projecting games)**.** *Let* G *be a game with processes* $\mathbb{P}$*, alphabet* A*. Let* $\mathbb{P}' \subseteq \mathbb{P}$ *a subset of the processes. The projection of* G *on* $\mathbb{P}'$ *is the game* G' *with processes and alphabet* $(\mathbb{P}', A', (A' \cap A_p)_{p\in\mathbb{P}'})$ *where* $A' = \{a \in A \mid \mathrm{dom}(a) \cap \mathbb{P}' \neq \emptyset\}$*, the states of a process* $p \in \mathbb{P}'$ *are the same in* G *and* G'*, every transition* $\delta \in \{a\} \times \Pi_{p\in\mathrm{dom}(a)} Q_p \times Q_p$ *of* G *on a letter* $a \in A'$ *is projected to* $\{a\} \times \Pi_{p\in\mathrm{dom}(a)\cap\mathbb{P}'} Q_p \times Q_p$*, and every transition on a letter* $a \notin A'$ *is simply deleted.*

The following result allows to combine two structurally decomposable games in order to obtain a new one. The proof can be found in appendix.

Lemma 9 (Merging games)**.** *Let* G *be a game, and* $\mathbb{P}_0, \mathbb{P}_1 \subseteq \mathbb{P}$ *two set of processes such that for every action* $a \in A$*,*

$$\mathrm{dom}(a) \cap \mathbb{P}_0 \neq \emptyset \wedge \mathrm{dom}(a) \cap \mathbb{P}_1 \neq \emptyset \implies \mathbb{P}_0 \cap \mathbb{P}_1 \subseteq \mathrm{dom}(a)) \ .$$

If both projections of G on $(\mathbb{P}_0 \setminus \mathbb{P}_1)$ and $(\mathbb{P}_1 \setminus \mathbb{P}_0)$ are structurally decomposable then G is structurally decomposable.

The merge operation can combine two structurally decomposable games in order to create a new one. For example all acyclic games can be obtained this way, since 3-player games are structurally decomposable and every tree with more than three nodes can be obtained by merging two strictly smaller subtrees.

This technique goes beyond acyclic games, by merging together several four player games and majority games, which preserves structural decomposability according to Lemma 9. The graph of processes is an undirected graph with nodes $\mathbb{P}$ and there is an edge between p and q if and only if there is an action $a \in A$ such that $\{p, q\} \subseteq \mathrm{dom}(a)$. For example, all the games whose graph of processes is the one depicted on Fig. 1 are structurally decomposable.

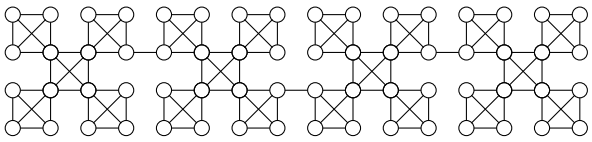

Figure 1: A decidable process architecture.

Games with five players Although our techniques do not provide an algorithm for solving games with five processes, they can address a special case.

Lemma 10. *Let G be a distributed game with five processes $\mathbb{P}$. Assume that the number of actions that a process can successively play in a row without synchronizing simultaneously with two other processes is bounded. Then G is process-decomposable.*

Proof. Let B be the corresponding bound. Let $\preceq$ the order on $\mathbb{P}$ which compares cardinality: $\mathbb{Q} \preceq \mathbb{Q}' \iff |\mathbb{Q}| \leq |\mathbb{Q}'|$. Then the game is $(\preceq, B)$-decomposable. Let xy a prime play such that $\forall p \in \mathrm{dom}(y), |y|_p \geq B$. Then by hypothesis, y has a prime prefix z such that $|\mathrm{dom}(\mathrm{last}(z))| \geq 3$. We set $\mathbb{Q} = \mathbb{P}$ then xz is a $\mathbb{Q}$-lock and $|\mathbb{Q} \setminus \mathrm{dom}(\mathrm{last}(z))| \leq 2 < 3 \leq |\mathrm{dom}(y)|$ thus the conditions of process decomposability are satisfied. □

Conclusion

We have presented a theorem and proof techniques that unifies several known decidability results for distributed games, and we have presented new examples of distributed games for which controller synthesis is decidable.

The decidability of distributed synthesis in the general case is still open to our knowledge, even in the simple case of a ring of five processes where each process can interact only with both processes on its left and on its right.

Another intriguing open problem is the case of *weakly k-connectedly communicating* plants. In such a plant, whenever two processes play both k times in a row without hearing from each other, they will never hear from each other

anymore. It is not known whether the MSO theory of the corresponding event structures is decidable or not [5], and we do not know either how to use techniques of this paper to solve this class of games.

Acknowledgements

We thank Blaise Genest, Anca Muscholl, Igor Walukiewicz, Paul Gastin and Marc Zeitoun for interesting discussions on the topic.

References

[1] V. Diekert and G. Rozenberg. *The Book of Traces*. World Scientific, 1995. URL: `https://books.google.co.uk/books?id=vNFLOE2pjuAC`.

[2] Bernd Finkbeiner and Sven Schewe. Uniform distributed synthesis. In *Logic in Computer Science, 2005. LICS 2005. Proceedings. 20th Annual IEEE Symposium on*, pages 321–330. IEEE, 2005.

[3] Paul Gastin, Benjamin Lerman, and Marc Zeitoun. Distributed games with causal memory are decidable for series-parallel systems. In *FSTTCS 2004: Foundations of Software Technology and Theoretical Computer Science, 24th International Conference, Chennai, India, December 16-18, 2004, Proceedings*, pages 275–286, 2004. URL: `http://dx.doi.org/10.1007/978-3-540-30538-5_23`, `doi:10.1007/978-3-540-30538-5_23`.

[4] Blaise Genest, Hugo Gimbert, Anca Muscholl, and Igor Walukiewicz. Asynchronous games over tree architectures. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, pages 275–286, 2013. URL: `http://dx.doi.org/10.1007/978-3-642-39212-2_26`, `doi:10.1007/978-3-642-39212-2_26`.

[5] P. Madhusudan, P. S. Thiagarajan, and Shaofa Yang. The MSO theory of connectedly communicating processes. In *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science, 25th International Conference, Hyderabad, India, December 15-18, 2005, Proceedings*, pages 201–212, 2005. URL: `http://dx.doi.org/10.1007/11590156_16`, `doi:10.1007/11590156_16`.

[6] A. Muscholl, I. Walukiewicz, and M. Zeitoun. A look at the control of asynchronous automata. In M. Mukund K. Lodaya and eds. N. Kumar, editors, *Perspectives in Concurrency Theory*. Universities Press, CRC Press, 2009.

[7] Anca Muscholl. Automated synthesis of distributed controllers. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*, pages 11–27, 2015. URL: `http://dx.doi.org/10.1007/978-3-662-47666-6_2`.

[8] Anca Muscholl and Igor Walukiewicz. Distributed synthesis for acyclic architectures. In *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, pages 639–651, 2014. URL: `http://dx.doi.org/10.4230/LIPIcs.FSTTCS.2014.639`.

[9] Amir Pnueli and Roni Rosner. Distributed reactive systems are hard to synthesize. In *Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on*, pages 746–757. IEEE, 1990.

[10] Peter JG Ramadge and W Murray Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.

[11] Wieslaw Ziełonka. Notes on finite asynchronous automata. *ITA*, 21(2):99–135, 1987.

Appendix

6 Elementary properties of traces

Not all properties of the concatenation operator and the prefix relation on words are preserved on traces, however the following are:

$$\forall u, v \in A^*, ((u \sqsubseteq v) \wedge (v \sqsubseteq u) \implies u = v) \ , \tag{3}$$
$$\forall u, v, w \in A^*, (uv = uw) \implies (v = w) \ , \tag{4}$$
$$\forall u, v, w \in A^*, (uv \sqsubseteq uw) \implies (v \sqsubseteq w) \ . \tag{5}$$

The following lemma lists some basic properties of traces used in the proofs.

Let $\mathbb{Q} \subseteq \mathbb{P}$ and u a trace. The trace u is said to be $\mathbb{Q}$-prime if for every linearization of u, the domain of the last letter intersects $\mathbb{Q}$. There exists a shortest prefix $\partial_{\mathbb{Q}}(u)$ of u, called the $\mathbb{Q}$-*view* and denoted $\partial_{\mathbb{Q}}(u)$ such that u factorizes as $u = \partial_{\mathbb{Q}}(u) \cdot v$ with $\mathrm{dom}(v) \cap \mathbb{Q} = \emptyset$. If $\mathbb{Q}$ is a singleton $\{p\}$ then $\partial_{\mathbb{Q}} = \partial_{\{p\}} = \partial_p$.

Lemma 11. *For every trace $u, v, x \in A^*$ and $a \in A$ and $B \subseteq A$,*

$$(u \ \mathbb{I} \ \mathbb{Q}) \iff (\partial_{\mathbb{Q}}(u) = \epsilon) \tag{6}$$
$$\partial_{\mathbb{Q}}(uv) = \partial_{\mathbb{Q}'}(u)\, \partial_{\mathbb{Q}}(v) \ \textit{where} \ \mathbb{Q}' = \mathrm{dom}(\partial_{\mathbb{Q}}(v)) \tag{7}$$
$$uv \ \textit{is} \ \mathbb{Q}\textit{-prime} \implies v \ \textit{is} \ \mathbb{Q}\textit{-prime} \tag{8}$$
$$u \ \textit{and} \ v \ \textit{are} \ \mathbb{Q}\textit{-prime} \implies uv \ \textit{is} \ \mathbb{Q}\textit{-prime} \tag{9}$$
$$\textit{If } ua \textit{ is prime}, (av \ \textit{is} \ \mathbb{Q}\textit{-prime} \iff uav \ \textit{is} \ \mathbb{Q}\textit{-prime}\ ) \tag{10}$$
$$(u \ \textit{is} \ \mathbb{Q}\textit{-prime} \wedge \neg(a \ \mathbb{I} \ u)) \implies (au \ \textit{is} \ \mathbb{Q}\textit{-prime}\ ) \tag{11}$$
$$\partial_{\mathbb{Q}}(\partial_{\mathbb{Q}}(u)) = \partial_{\mathbb{Q}}(u) \tag{12}$$
$$\partial_{\mathbb{Q}}(uv) = \partial_{\mathbb{Q}}(u\, \partial_{\mathbb{Q}}(v)) \tag{13}$$
$$(u \sqsubseteq \partial_{\mathbb{Q}}(uv)) \iff (\partial_{\mathbb{Q}}(uv) = u\, \partial_{\mathbb{Q}}(v)) \tag{14}$$
$$(uw \sqsubseteq \partial_{\mathbb{Q}}(uv)) \implies (w \sqsubseteq \partial_{\mathbb{Q}}(v)) \tag{15}$$
$$\textit{If } ua \textit{ is prime},$$
$$\partial_{\mathbb{Q}}(uav) = ua\, \partial_{\mathbb{Q}}(v) \iff \partial_{\mathbb{Q}}(av) = a\, \partial_{\mathbb{Q}}(v) \tag{16}$$
$$(\partial_{\mathbb{Q}}(uv) = \partial_{\mathbb{Q}}(u)) \implies \partial_{\mathbb{Q}}(v) = \epsilon \ . \tag{17}$$

Proof. The equivalence (6) is immediate from the definition of $\partial_{\mathbb{Q}}$. Equality (7) is proved by induction on $|u| + |v|$.

We prove (8). If the last letter of a word $v' \in v$ is not in $\mathbb{Q}$, then the same holds for every $u'v'$ where u' is a linearization of the trace u thus uv is not $\mathbb{Q}$-prime since $u'v'$ is a linearization of the trace uv.

We prove (9). Assume both u and v are $\mathbb{Q}$-prime. Every linearization of uv is an interleaving of a linearization of u and a linearization of v thus it terminates with a letter whose domain intersects $\mathbb{Q}$. Hence uv is $\mathbb{Q}$-prime.

We prove (10). Assume ua prime. The converse implication follows from (8). Assume av is $\mathbb{Q}$-prime. We prove that uav is $\mathbb{Q}$-prime by induction on $|u|$. If $|u| = 0$ then $u = \epsilon$ and $uav = av$ is $\mathbb{Q}$-prime by hypothesis. By induction let $n \in \mathbb{N}$ and assume $u'av$ is $\mathbb{Q}$-prime for all u' such that $|u'| \leq n$. Let u such that $|u| = n + 1$, we prove that uav is $\mathbb{Q}$-prime. Since $|u| = n + 1$, there exists

$b \in A$ and $u' \in A^*$ such that $u = bu'$ and $|u'| = n$. Using (8) and the induction hypothesis so on one hand we know that $u'av$ is $\mathbb{Q}$-prime. By definition of a trace, for any trace w,

$$bw = \{xbz \mid x, z \text{ words on } A \ , xz \in w, b \mathbb{I} x\} \ . \tag{18}$$

Let y a linearization of $uav = bu'av$, we prove that the last letter of y is in $A_{\mathbb{Q}} = \cup_{p \in \mathbb{Q}} A_p$. According to (18), y factorizes as $y = xbz$ with $xz \in u'av$ and $x \mathbb{I} b$. Since $xz \in u'av$ and $u'av$ is $\mathbb{Q}$-prime, if z is not empty then it ends with a letter in $\cup_{p \in \mathbb{Q}} A_p$ and so does y. Assume now that z is empty, then $y = xb$ with $x \in u'av$ and $x \mathbb{I} b$. Since $y \in bu'av$ then $\mathrm{Alph}(u'a) \subseteq \mathrm{Alph}(y)$ and $\mathrm{Alph}(v) \subseteq \mathrm{Alph}(y)$. Since $\mathrm{Alph}(y) = \mathrm{Alph}(x) \cup \{b\}$ and $x \mathbb{I} b$ every letter of $u'a$ and av commute with b thus $bu'a = u'ab$ and $bv = vb$. Since $bu'a = ua$ is prime, $bu'a = u'ab$ implies $a = b$. Since $bv = vb$ then $av = va$ and since av is $\mathbb{Q}$-prime, $a = b \in B$. Finally $b \in A_{\mathbb{Q}}$ and since $y = xb$ the last letter of y is in $A_{\mathbb{Q}}$, which terminates the proof of the inductive step, and the proof of (10).

We prove (11) by contradiction. Assume au is not $\mathbb{Q}$-prime then there exists a word v' and a letter c such $\mathrm{dom}(c) \cap \mathbb{Q} = \emptyset$ and $v'c \in au$. Let $u' \in u$ then $au' \in au$ and $v'c \approx au'$ thus $\mathrm{Alph}(v') \cup \{c\} = \mathrm{Alph}(u') \cup \{a\}$. If $a \notin \mathrm{Alph}(v')$ then $a = c$ and $v'c \approx au'$ implies $a \mathbb{I} u$ which is false by hypothesis. Thus $a \in \mathrm{Alph}(v')$. Let w' be the longest prefix of v' which does not contain a and x' the suffix of v' such that $v' = w'ax'c$. Then $au' \approx w'ax'c$ and $a \notin \mathrm{Alph}(w')$ thus $w' \mathbb{I} a$. Then $w'ax'c \approx aw'x'c$ thus $aw'x'c \approx au'$ hence $w'x'c \approx u'$ and $w'x'c \in u$. Since $\mathrm{dom}(c) \cap \mathbb{Q} = \emptyset$, this contradicts the hypothesis u is $\mathbb{Q}$-prime.

We prove (12). Since $\partial_{\mathbb{Q}}(u) \sqsubseteq u$, then $\partial_{\mathbb{Q}}((\partial_{\mathbb{Q}}(u)) \sqsubseteq \partial_{\mathbb{Q}}(u)$ according to (3) it is enough to prove $\partial_{\mathbb{Q}}(u) \sqsubseteq \partial_{\mathbb{Q}}(\partial_{\mathbb{Q}}(u))$. By definition of $\partial_{\mathbb{Q}}$, $u = \partial_{\mathbb{Q}}(u)u'$ with $\mathrm{dom}(u') \cap \mathbb{Q} = \emptyset$ and $\partial_{\mathbb{Q}}(u) = \partial_{\mathbb{Q}}(\partial_{\mathbb{Q}}(u))u''$ with $\mathrm{dom}(u'') \cap \mathbb{Q} = \emptyset$. Thus $u = \partial_{\mathbb{Q}}(\partial_{\mathbb{Q}}(u))u''u'$ with $\mathrm{dom}(u''u') \cap \mathbb{Q} = \emptyset$ hence by definition of $\partial_{\mathbb{Q}}$, $\partial_{\mathbb{Q}}(u) \sqsubseteq \partial_{\mathbb{Q}}(\partial_{\mathbb{Q}}(u))$.

We prove (13). Since $u\,\partial_{\mathbb{Q}}(v) \sqsubseteq uv$, then $\partial_{\mathbb{Q}}(u\,\partial_{\mathbb{Q}}(v)) \sqsubseteq \partial_{\mathbb{Q}}(uv)$ and according to (3) it is enough to prove $\partial_{\mathbb{Q}}(uv) \sqsubseteq \partial_{\mathbb{Q}}(u\,\partial_{\mathbb{Q}}(v))$. By definition of $\partial_{\mathbb{Q}}(v)$ there exists $\mathrm{dom}(v') \cap \mathbb{Q} = \emptyset$ such that $v = \partial_{\mathbb{Q}}(v)v'$ then $uv = u\,\partial_{\mathbb{Q}}(v)v'$ thus $\partial_{\mathbb{Q}}(uv) \sqsubseteq u\,\partial_{\mathbb{Q}}(v)$. According to (12), $\partial_{\mathbb{Q}}(\partial_{\mathbb{Q}}(uv)) = \partial_{\mathbb{Q}}(uv)$ thus $\partial_{\mathbb{Q}}(uv) \sqsubseteq \partial_{\mathbb{Q}}(u\,\partial_{\mathbb{Q}}(v))$ which terminates the proof of (13).

We prove (14). The converse implication in (14) is obvious so it is enough to prove the direct implication. Assume $u \sqsubseteq \partial_{\mathbb{Q}}(uv)$. According to (3) it is enough to prove both $\partial_{\mathbb{Q}}(uv) \sqsubseteq u\,\partial_{\mathbb{Q}}(v)$ and $u\,\partial_{\mathbb{Q}}(v) \sqsubseteq \partial_{\mathbb{Q}}(uv)$. We start with $u\,\partial_{\mathbb{Q}}(v) \sqsubseteq \partial_{\mathbb{Q}}(uv)$. Since $u \sqsubseteq \partial_{\mathbb{Q}}(uv)$, then $\partial_{\mathbb{Q}}(uv) = uw$ for some $w \in A^*$ and $uv = uww'$ for some w' such that $\mathrm{dom}(w') \cap \mathbb{Q} = \emptyset$. Then $v = ww'$ according to (4) and since $\mathrm{dom}(w') \cap \mathbb{Q} = \emptyset$, then $\partial_{\mathbb{Q}}(v) \sqsubseteq w$, thus $u\,\partial_{\mathbb{Q}}(v) \sqsubseteq uw = \partial_{\mathbb{Q}}(uv)$ and we got the first prefix relation. Now we prove the converse prefix relation. Since $\partial_{\mathbb{Q}}(v) \sqsubseteq w$ then by definition of $\partial_{\mathbb{Q}}$ there exists $w'' \in A^*$ such that $w = \partial_{\mathbb{Q}}(v)w''$ and $\mathrm{dom}(w'') \cap \mathbb{Q} = \emptyset$. Then $uv = u\,\partial_{\mathbb{Q}}(v)w''w'$ and $\mathrm{dom}(w''w') \cap \mathbb{Q} = \emptyset$ thus by definition of $\partial_{\mathbb{Q}}$, $\partial_{\mathbb{Q}}(uv) \sqsubseteq u\,\partial_{\mathbb{Q}}(v)$. By definition of w this implies $uw \sqsubseteq u\,\partial_{\mathbb{Q}}(v)$ thus according to (5) $w \sqsubseteq \partial_{\mathbb{Q}}(v)$. Finally $w = \partial_{\mathbb{Q}}(v)$ and $u\,\partial_{\mathbb{Q}}(v) = uw = u\,\partial_{\mathbb{Q}}(v)$ which terminates the proof of (14).

Equation (15) is a direct corollary of (14). Let v', w' such that $\partial_{\mathbb{Q}}(uv) = uwv'$ and $uv = \partial_{\mathbb{Q}}(uv)w'$. Then according to (14), $uwv' = u\,\partial_{\mathbb{Q}}(wv'w')$ thus according to (4), $wv' = \partial_{\mathbb{Q}}(wv'w')$ and since $v = wv'w'$, we get $w \sqsubseteq \partial_{\mathbb{Q}}(v)$.

By definition $\partial_{\mathbb{Q}}(uv)$ is the shortest prefix of uv such that $uv{=}\partial_{\mathbb{Q}}(uv)v'$ with

$\mathrm{dom}(v') \cap \mathbb{Q} = \emptyset$, thus by hypothesis there exists w' such that $uv = uww'v'$. $v = ww'v'$ thus by definition of $\partial_{\mathbb{Q}}(v)$ again, $ww' \sqsubseteq \partial_{\mathbb{Q}}(v)$ thus $w \sqsubseteq \partial_{\mathbb{Q}}(v)$.

We prove (16). Let $\mathbb{Q}' = \mathrm{dom}(\partial_{\mathbb{Q}}(v))$ Then according to (7) $\partial_{\mathbb{Q}}(uav) = \partial_{\mathbb{Q}'}(ua)\,\partial_{\mathbb{Q}}(v)$ and $\partial_{\mathbb{Q}}(av) = \partial_{\mathbb{Q}'}(a)\,\partial_{\mathbb{Q}}(v)$ thus $\partial_{\mathbb{Q}}(av) = a\,\partial_{\mathbb{Q}}(v)$ iff $\mathrm{dom}(a) \cap \mathrm{dom}(\mathbb{Q}') \neq \emptyset$. iff $\partial_{\mathbb{Q}'}(ua) = ua$ (since ua is prime).

Finally we prove (17). Let $\mathbb{Q}' = \mathbb{Q} \cup \mathrm{dom}(\partial_{\mathbb{Q}}(v))$. Then according to (7), $\partial_{\mathbb{Q}}(uv) = \partial_{\mathbb{Q}'}(u)\,\partial_{\mathbb{Q}}(v)$. Since $\mathbb{Q} \subseteq \mathbb{Q}'$ then $\partial_{\mathbb{Q}}(u) \sqsubseteq \partial_{\mathbb{Q}'}(u)$ thus $\partial_{\mathbb{Q}}(uv) = \partial_{\mathbb{Q}}(u)$ implies $\partial_{\mathbb{Q}}(u) = \partial_{\mathbb{Q}'}(u) = \partial_{\mathbb{Q}'}(u)\,\partial_{\mathbb{Q}}(v)$ hence $\partial_{\mathbb{Q}}(v) = \emptyset$. □

7 Proof of Lemma 2

Lemma 2 *Let (x, y) be a useless $\mathbb{Q}$-repetition in a strategy σ. Then the mapping τ from traces to traces defined by*

$$\tau : u \mapsto \begin{cases} \sigma(u) & \text{if } x \not\sqsubseteq u \\ \sigma(xyu') & \text{if } u = xu' \end{cases}$$

is a distributed strategy. Moreover for every trace z,

$$(xz \text{ is a } \tau\text{-play}) \iff (xyz \text{ is a } \sigma\text{-play}) \ . \tag{19}$$

If σ is a winning strategy then τ is winning as well and has a strictly smaller duration.

Proof. Let $b = \mathrm{last}(x)$. Like in the definition of useless repetitions, set

$$A_{\mathbb{Q}||b} = \{a \in A \mid \mathrm{dom}(a) \subseteq \mathbb{Q} \text{ and } \mathrm{dom}(a) \cap \mathrm{dom}(b) = \emptyset\}$$
$$\mathbb{Q}_{?b} = \mathrm{dom}\left(A_{\mathbb{Q}||b}\right) \ .$$

Since (x, y) is a useless repetition then $\mathrm{strate}_{\mathbb{Q},\sigma}(x) = \mathrm{strate}_{\mathbb{Q},\sigma}(xy)$ thus

$$\text{both } x \text{ and } xy \text{ are prime} \tag{20}$$
$$\mathrm{dom}(y) \subseteq \mathbb{Q} \tag{21}$$
$$\mathrm{last}(x) = \mathrm{last}(xy) = b \tag{22}$$
$$\forall p \in \mathbb{Q}, \sigma_p(x) = \sigma_p(xy) \tag{23}$$
$$\forall p \in \mathbb{P}, \mathrm{state}_p(x) = \mathrm{state}_p(xy) \tag{24}$$
$$\text{the projections of } \sigma[x] \text{ and } \sigma[xy] \text{ on } \mathbb{Q}_{?b} \text{ coincide.} \tag{25}$$
$$\text{both } x \text{ and } xy \text{ are } \mathbb{Q}\text{-locks} \tag{26}$$

Proof that τ is a distributed strategy. To prove that τ is a strategy, we prove that for every process p and trace u,

$$\tau_p(u) = \tau_p(\partial_p(u)) \ .$$

Assume $x \not\sqsubseteq u$. Then $\tau_p(u) = \sigma_p(u) = \sigma_p(\partial_p(u)) = \tau_p(\partial_p(u))$ thus this case is easy.

Assume $x \sqsubseteq u$ then $u = xv$ for some trace v. By definition of τ, $\tau_p(u) = \sigma_p(xyv)$ and since σ is a distributed strategy $\sigma_p(xyv) = \sigma_p(\partial_p(xyv))$, thus it is enough to prove that for every process p and trace v,

$$\sigma_p(\partial_p(xyv)) = \tau_p(\partial_p(xv)) \ . \tag{27}$$

According to (13), $\partial_p(xyv) = \partial_p(xy\,\partial_p(v))$ and $\partial_p(xv) = \partial_p(x\,\partial_p(v))$ thus in (27) we can replace v by $\partial_p(v)$ and assume w.l.o.g. that

$$v = \partial_p(v) \ . \tag{28}$$

We distinguish between several cases.

First case: proof of (27) **in case** $x \sqsubseteq \partial_p(xv)$**.** We start with proving

$$\partial_p(xyv) = xy\,\partial_p(v) \ . \tag{29}$$

Since $x \sqsubseteq \partial_p(xv)$, (14) implies

$$\partial_p(xv) = x\,\partial_p(v) \ . \tag{30}$$

According to (20) both x and xy are prime. Since moreover $\partial_p(xv) = x\,\partial_p(v)$ we can apply (16) twice and get first $\partial_p(bv) = b\,\partial_p(v)$ and then (29). Now that (29) is proved we can conclude the first case:

$$\begin{aligned} \sigma_p(\partial_p(xyv)) &= \sigma_p(xy\,\partial_p(v)) && (31)\\ &= \tau_p(x\,\partial_p(v)) && (32)\\ &= \tau_p(\partial_p(xv)) \ , && (33) \end{aligned}$$

where (31) comes from (29), (32) hold by definition of τ, and (33) comes from (30). Thus (27) holds in this first case.

Second case: proof of (27) **in case** $x \not\sqsubseteq \partial_p(xv)$ **and** $v = \epsilon$ Since $x \not\sqsubseteq \partial_p(xv)$ then $\tau_p(\partial_p(xv)) = \sigma_p(\partial_p(xv))$. And since $v = \epsilon$ then (27) is equivalent to

$$\sigma_p(\partial_p(x)) = \sigma_p(\partial_p(xy)) \ . \tag{34}$$

There are two subcases. In case $p \notin \mathrm{dom}(y)$ then $\partial_p(xy) = \partial_p(x)$. In case $p \in \mathrm{dom}(y)$ then $p \in \mathbb{Q}$ according to (21) and since (x, y) is a useless repetition then: $\sigma_p(\partial_p(x)) = \sigma_p(x) = \sigma_p(xy) = \sigma_p(\partial_p(xy))$ where the central equality is (23).

Third case: proof of (27) **in case** $x \not\sqsubseteq \partial_p(xv)$ **and** $v \neq \epsilon$**.** Since $x \not\sqsubseteq \partial_p(xv)$ then $\tau_p(\partial_p(xv)) = \sigma_p(\partial_p(xv))$ and (27) is equivalent to

$$\sigma_p(\partial_p(xyv)) = \sigma_p(\partial_p(xv)) \ . \tag{35}$$

We show that either $\mathrm{dom}(v) \subseteq \mathbb{Q}$ or $\mathrm{dom}(v) \cap \mathbb{Q} = \emptyset$. The proof is by contradiction. Assume otherwise that $\mathrm{dom}(v)$ intersects both $\mathbb{Q}$ and $\mathbb{P} \setminus \mathbb{Q}$. Since $\partial_p(v) = v$ (i.e. (28)) then v is a suffix of $\partial_p(xv)$. Thus $\mathrm{dom}(\partial_p(xv))$ intersects both $\mathbb{Q}$ and $\mathbb{P} \setminus \mathbb{Q}$. Since $\partial_p(xv)$ is a prime play, $\mathrm{dom}(\partial_p(xv))$ contains a letter a whose domain intersects both $\mathbb{Q}$ and $\mathbb{P} \setminus \mathbb{Q}$. Notice that x and $\partial_p(xv)$ have a common extension xv and x is a $\mathbb{Q}$-lock because (x, y) is a useless $\mathbb{Q}$-repetition. Thus by definition of $\mathbb{Q}$-locks, the two prime plays x and $\partial_p(xv)$ cannot be parallel hence $x \sqsubseteq \partial_p(xv)$ or $\partial_p(xv) \sqsubseteq x$. Since by hypothesis $x \not\sqsubseteq \partial_p(xv)$ then $\partial_p(xv) \sqsubseteq x$ thus $\partial_p(xv) \sqsubseteq \partial_p(x)$ according to (13) hence $\partial_p(xv) = \partial_p(x)$ hence $\partial_p(v) = \epsilon$ according to (17), a contradiction with $\partial_p(v) = v \neq \epsilon$.

Thus $\mathrm{dom}(v) \subseteq \mathbb{Q}$ or $\mathrm{dom}(v) \cap \mathbb{Q} = \emptyset$ and the proof splits into two subcases.

First subcase: proof of (35) **in case** $x \not\sqsubseteq \partial_p(xv)$**,** $v \neq \epsilon$ **and** $\mathrm{dom}(v) \cap \mathbb{Q} = \emptyset$
Since $\mathrm{dom}(y) \subseteq \mathbb{Q}$, the hypothesis $\mathrm{dom}(v) \cap \mathbb{Q} = \emptyset$ (i.e. (21)) implies $\mathrm{dom}(v) \cap \mathrm{dom}(y) = \emptyset$. And since $\partial_p(v) = v$ (i.e. (28)) and $v \neq \epsilon$ then $p \in \mathrm{dom}(v)$ thus $p \notin \mathrm{dom}(y)$. We can terminate the proof of (35):

$$\sigma_p(\partial_p(xyv)) = \sigma_p(\partial_p(xvy) \tag{36}$$
$$= \sigma_p(\partial_p(xv)) \tag{37}$$
$$= \tau_p(\partial_p(xv)) \ , \tag{38}$$

where (36) holds because $\mathrm{dom}(v) \cap \mathrm{dom}(y) = \emptyset$, (37) holds because $p \notin \mathrm{dom}(y)$ and (38) holds by definition of τ_p, since by hypothesis $x \not\sqsubseteq \partial_p(xv)$.

Second subcase: proof of (35) **in case** $x \not\sqsubseteq \partial_p(xv)$**,** $v \neq \epsilon$ **and** $\mathrm{dom}(v) \subseteq \mathbb{Q}$
We first establish

$$\mathrm{dom}(v) \subseteq \mathbb{Q}_{?b} \ . \tag{39}$$

By hypothesis $v = \partial_p(v)$ thus according to (7), $\partial_p(xv) = \partial_{\mathrm{dom}(v)}(x)v$. By hypothesis $x \not\sqsubseteq \partial_p(xv)$ thus $x \neq \partial_{\mathrm{dom}(v)}(x)$. Since x is prime and $b = \mathrm{last}(x)$ then (16) implies $\partial_{\mathrm{dom}(v)}(b) \neq b$ thus $\mathrm{dom}(v) \cap \mathrm{dom}(b) = \emptyset$. Since moreover $\mathrm{dom}(v) \subseteq \mathbb{Q}$ then $\mathrm{Alph}(v) \subseteq A_{\mathbb{Q}||b}$ hence (39).

Since (x, y) is a useless repetition in σ, we can apply (26) hence

$$\sigma_p(xv) = \sigma_p(xyv) \ , \tag{40}$$

which shows (35) since σ_p is a strategy.

This terminates the proof of (35) in the second subcase, thus (27) holds and as a consequence, τ is a distributed strategy. □

Proof of property (19). Let $xv \in A^*$ be a τ-play, we prove that xyv is a σ-play by induction on v. When $v = \epsilon$ then xy is a σ-play because by hypothesis (x, y) is a useless repetition in σ. For the inductive step, assume xyv is a σ-play, let $c \in A$ such that xvc is a τ-play, and let us prove that $xyvc$ is a σ-play. Since xvc is a τ-play,

$$\forall p \in \mathrm{dom}(c), c \in \tau_p(xv)$$

Thus by definition of τ,

$$\forall p \in \mathrm{dom}(c), c \in \sigma_p(xyv)$$

hence $xyvc$ is a σ-play. □

Proof that τ *is winning.* Since σ is winning, the set of σ-plays is finite. According to property (19) and the definition of τ, every τ-play is either a σ-play or is a subword of a σ-play thus the set of τ-plays is finite as well. Let u be a maximal τ-play.

If $x \not\sqsubseteq u$ then u is a maximal σ-play and since σ is winning u is a winning play.

Otherwise $x \sqsubseteq u$ and u factorizes as $u = xw$. Since (x, y) is a useless $\mathbb{Q}$-repetition then according to (24) all processes are in the same state in x and xy. Since transitions are deterministic, all processes are in the same state in xw and xyw. According to (19), since xw is a maximal τ-play, xyw is a maximal σ-play, and since σ is winning, $\forall p \in \mathbb{P}, \mathrm{state}_p(xyw) \in F_p$. Thus $\forall p \in \mathbb{P}, \mathrm{state}_p(xw) \in F_p$. This terminates the proof that τ is winning. □

All statements of Lemma 2 have been proved. □

8 Proof of Lemma 6

Lemma 6 *Whether a game is decomposable is decidable. In case a game is* $(\preceq, k)$*-decomposable for some* k*, it is* $(\preceq, 1 + |Q|^{|\mathbb{P}|} \cdot 2^{|A|})$*-decomposable.*

Proof. The definition of locks can be reformulated using the notion of locked states.

Definition 14 ($\mathbb{Q}$-locked global states)**.** *A global state* $(q_p)_{p\in\mathbb{P}} \in \Pi_{p\in\mathbb{P}} Q_p$ *is* $\mathbb{Q}$-locked *iff for every prime play* v *starting from this state either* $\mathrm{dom}(v) \subseteq \mathbb{Q}$ *or* $\mathrm{dom}(v) \cap \mathbb{Q} = \emptyset$*.*

Lemma 12. *Let* $\mathbb{Q} \subseteq \mathbb{P}$*. A prime play with last letter* b *and global state* $(q_p)_{p\in\mathbb{P}}$ *is a* $\mathbb{Q}$*-lock iff the global state* $(q_p)_{p\in\mathbb{P}}$ *is* $\mathbb{Q}$*-locked in the automaton restricted to letters whose domain does not intersect* $\mathrm{dom}(b)$*.*

Proof. Simple reformulation. □

We reformulate what it means for a game *not* to be decomposable, in terms of computations of synchronous automata. For every $b \in A$, denote $\mathcal{A}_b$ the automaton restricted to letters whose domain does not intersect $\mathrm{dom}(b)$. Let

$$\mathcal{C} = \{(\mathbb{Q}, b, (q_p)_{p\in\mathbb{P}})) \mid b \in A, \mathbb{Q}_{?b} \prec \mathbb{Q} \subseteq \mathbb{P} \text{ and } (q_p)_{p\in\mathbb{P}} \text{ is a } \mathbb{Q}\text{-locked in } \mathcal{A}_b\} \ .$$

Checking whether a global state is $\mathbb{Q}$-locked reduces to checking accessibility in the graph of the global states of the automaton, thus the set of $\mathbb{Q}$-locked global states is computable and $\mathcal{C}$ is computable as well.

Denote $\mathcal{A}'$ the automaton obtained by turning $\mathcal{A}$ into a synchronous automaton on A^* which computes on-the-fly the current global state in $\Pi_{p\in\mathbb{P}} Q_p$ as well as the list $L \subseteq A$ of maximal actions of the current play. In particular, $\mathcal{A}'$ can detect whether the current input word is a prime trace, which is equivalent to $|L| = 1$. Denote Z the set of states of $\mathcal{A}'$ (i.e. global states of $\mathcal{A}$) accessible from the initial state by a *prime* trace. We say that a transition (z, a, z') of $\mathcal{A}'$ is $\mathbb{Q}$-good if $(\mathbb{Q}, a, z') \in \mathcal{C}$. The following properties are equivalent.

i) The game is decomposable.

ii) There exists an integer $k \in \mathbb{N}$ such that for every prime play xy, if $\forall p \in \mathrm{dom}(y), |y|_p \geq k$ then xy has a prime prefix $xy' \sqsubseteq xy$ such that $(\mathrm{dom}(y), \max_A(xy'), \mathrm{state}(xy')) \in \mathcal{C}$.

iii) For every state $z \in Z$ and every word $u \in A^*$, if $\mathcal{A}'$ has a computation on u from z to z this computation uses at least one transition (z, a, z') such that $(\mathrm{dom}(u), a, z') \in \mathcal{C}$.

Property i) and ii) are equivalent according to Lemma 12.

We show that if property iii) does not hold then ii) does not hold either. In this case there exists $z \in Z$ such that

$$L_z = \{v \in A^* \mid \ \mathcal{A}' \text{ has a computation on } v \text{ from } z \text{ to } z \text{ without any } \mathrm{dom}(u)\text{-good transition}\}$$

is non-empty. The set of transitions used by a computation does not change when commuting independent letters thus L_z is closed by commutation, let u be a trace whose all linearizations are in L_z. Let x a trace such that the

computations of $\mathcal{A}'$ on the linearizations of x reach z from the initial state. Since L_z is closed by commutation, by prefix, and $L_z^* = L_z$ then, for every $k \in \mathbb{N}$, no prime prefix xy' of xu^k satisfies $(\mathrm{dom}(y), \max_A(xy'), \mathrm{state}(xy')) \in \mathcal{C}$, hence ii) does not hold.

We show that if property iii) holds then property ii) holds as well. Denote $|\mathcal{A}'|$ the number of states of $\mathcal{A}'$ and $N = |\mathcal{A}'| + 1$. Let k be some integer. and let xy be a prime play such that $\forall p \in \mathrm{dom}(y), |y|_p \geq k$. Let $y = a_1 \cdots a_n$. According to Ramsey theorem, for k large enough there must exist a factorization $y = y_1 y_2 \cdots y_N$ such that every play $(y_1 \cdots y_i)_{1 \leq i \leq N}$ is prime and $\forall 1 \leq i \leq k, \mathrm{dom}(u_i) = \mathrm{dom}(y)$. The way to apply Ramsey theorem is to consider the set of indices between 1 and $|u|$ as the nodes of a graph and the edge from i to j with $i < j$ is labelled with $\mathrm{dom}(a_i \cdots a_j)$. Then if the graph is large enough (more than k indices), it contains a monochromatic clique of size N.

Since $\mathcal{A}'$ has $< N$ states, there exists $1 \leq i < j \leq N$ such that $\mathcal{A}'$ has the same state z after words $xy_1 \cdots y_i$ and $xy_1 \cdots y_j$. Since $y_1 \cdots y_j$ is prime then $u = y_{i+1} \cdots y_j$ is prime as well thus $z \in Z$. Since $\mathrm{dom}(u) = \mathrm{dom}(y)$ and the computation of $\mathcal{A}'$ on u is a loop from z to z then according to iii) this loop contains at least one $\mathrm{dom}(y)$-good transition δ. Let x a trace such that the computation of $\mathcal{A}'$ on x (independently of the linearization) reach z from the initial state. Then a prime prefix of xu contains the same transition δ, thus ii) holds.

Then according to iii), xu has a prefix $xy'' \sqsubseteq xu$ such that $(\mathrm{dom}(y), \mathrm{last}(y''), \mathrm{state}(xy'')) \in \mathcal{C}$. Then $y' = \partial_{\mathbb{Q}}(y'')$ is a prime prefix of xu such that $(\mathrm{dom}(y), \mathrm{last}(y'), \mathrm{state}(xy')) \in \mathcal{C}$ Hence property ii) holds.

Property iii) is decidable using standard techniques of automata theory, for example one can build a non-deterministic automaton on finite words which non-deterministically guesses z, $\mathbb{Q} \subseteq \mathbb{P}$ and a loop of $\mathcal{A}'$ on z where only processes of $\mathbb{Q}$ play, each one of them plays at least once but no $\mathbb{Q}$-good transition is visited. Then property iii) is equivalent to the emptiness of this automaton.

The upper-bound on k is a by-product of the proof of iii) $\Longrightarrow$ ii). □

9 Majority games: proof of Lemma 8

Lemma 8 *Assume that every non-local action synchronizes a majority of the processes i.e. for every action a, $|\mathrm{dom}(a)| = 1$ or $|\mathrm{dom}(a)| \geq |\mathbb{P} \setminus \mathrm{dom}(a)|$. Then the game is structurally decomposable.*

Proof. We show that the game is structurally decomposable for the pre-order $\preceq$ on $2^{\mathbb{P}}$ defined by: $\mathbb{Q} \preceq \mathbb{Q}' \iff |\mathbb{Q}| \leq |\mathbb{Q}'|$. Let y be a prime trace.

If $\mathrm{dom}(y)$ contains a single process $\{p\}$ then we set $b = \mathrm{last}(y)$ and $\mathbb{Q} = \{p\}$. Then every letter $a \mathbb{I} b$ is $\mathbb{Q}$-safe and $\mathbb{Q} \setminus \mathrm{dom}(b) = \emptyset \prec \{p\}$.

If $|\mathrm{dom}(y)| \geq 2$ then, since y is prime, y contains a letter b such that $|\mathrm{dom}(b)| \geq 2$. By hypothesis $|\mathbb{P} \setminus \mathrm{dom}(b)| \leq |\mathrm{dom}(b)|$. We set

$$\mathbb{Q} = \begin{cases} \mathbb{P} & \text{if } 2 * |\mathrm{dom}(y)| > |\mathbb{P}| \\ \mathrm{dom}(y) & \text{otherwise.} \end{cases}$$

In the first case $2 * |\operatorname{dom}(y)| > |\mathbb{P}|$ then $\mathbb{Q} = \mathbb{P}$ and every action is $\mathbb{Q}$-safe. Since $|\mathbb{P} \setminus \operatorname{dom}(b)| \leq |\operatorname{dom}(b)|$ then $2 * |\operatorname{dom}(b)| \geq |\mathbb{P}|$ hence $2(|\mathbb{P}| - |\operatorname{dom}(b)|) \leq |\mathbb{P}| < 2 * |\operatorname{dom}(y)|$. Thus $(\mathbb{P} \setminus \operatorname{dom}(b)) \prec \operatorname{dom}(y)$.

In the second case $2 \leq |\operatorname{dom}(y)|$ and $2 * |\operatorname{dom}(y)| \leq |\mathbb{P}|$. Then $(\mathbb{Q} \setminus \operatorname{dom}(b)) \prec \operatorname{dom}(y)$ because $\operatorname{dom}(b) \subseteq \mathbb{Q} = \operatorname{dom}(y)$. Since $|\mathbb{P}| \leq 2 * |\operatorname{dom}(b)| \leq 2 * |\operatorname{dom}(y)| = |\mathbb{P}|$ then $\operatorname{dom}(b) = \operatorname{dom}(y) = \mathbb{Q}$ thus every action $a \mathbb{I} b$ is $\mathbb{Q}$-safe. □

10 Merging games: proof of Lemma 9

Lemma 9 *Let G be a game, and $\mathbb{P}_0, \mathbb{P}_1 \subseteq \mathbb{P}$ two set of processes such that for every action $a \in A$, $(\operatorname{dom}(a) \cap \mathbb{P}_0 \neq \emptyset \wedge \operatorname{dom}(a) \cap \mathbb{P}_1 \neq \emptyset \implies \mathbb{P}_0 \cap \mathbb{P}_1 \subseteq \operatorname{dom}(a))$. If both projections of G on $(\mathbb{P}_0 \setminus \mathbb{P}_1)$ and $(\mathbb{P}_1 \setminus \mathbb{P}_0)$ are structurally decomposable then G is structurally decomposable.*

Proof. Let G_0 and G_1 the projections of G on $\mathbb{P}_0$ and $\mathbb{P}_1$ and $\preceq_0, \preceq_1$ some preorders witnessing that G_0 and G_1 are structurally decomposable. Let $\preceq$ be the preorder on $2^{\mathbb{P}}$ defined by:

$$\mathbb{Q} \preceq \mathbb{Q}' \iff \begin{cases} & \mathbb{Q}' \cap \mathbb{P}_0 \neq \emptyset \wedge \mathbb{Q}' \cap \mathbb{P}_1 \neq \emptyset \\ \vee & (\mathbb{Q}' \subseteq \mathbb{P}_0 \setminus \mathbb{P}_1) \wedge \mathbb{Q} \cap (\mathbb{P}_0 \setminus \mathbb{P}_1) \preceq_0 \mathbb{Q}' \\ \vee & (\mathbb{Q}' \subseteq \mathbb{P}_1 \setminus \mathbb{P}_0) \wedge \mathbb{Q} \cap (\mathbb{P}_1 \setminus \mathbb{P}_0) \preceq_1 \mathbb{Q}' \end{cases}$$

which coincides with $\preceq_0$ and $\preceq_1$ on $2^{\mathbb{P}_0 \setminus \mathbb{P}_1}$ and $2^{\mathbb{P}_1 \setminus \mathbb{P}_0}$ respectively and all sets intersecting both $\mathbb{P}_0$ and $\mathbb{P}_1$ are $\preceq$-equivalent and strictly $\prec$-greater than sets in $2^{\mathbb{P}_0 \setminus \mathbb{P}_1} \cup 2^{\mathbb{P}_1 \setminus \mathbb{P}_0}$. Then $\preceq$ is monotonic with respect to inclusion.

We show that G is structurally $\preceq$-decomposable. Let y be a prime trace.

Assume first $(\operatorname{dom}(y) \cap \mathbb{P}_0 \neq \emptyset \wedge \operatorname{dom}(y) \cap \mathbb{P}_1 \neq \emptyset)$. We set $\mathbb{Q} = \mathbb{P}$. Since y is prime, y has at least one letter b whose domain intersects both $\mathbb{P}_0$ and $\mathbb{P}_1$ thus by hypothesis $\mathbb{P}_0 \cap \mathbb{P}_1 \subseteq \operatorname{dom}(b)$. Hence by definition of $\preceq$, $(\mathbb{P} \setminus \operatorname{dom}(b)) \prec \operatorname{dom}(b) \preceq \operatorname{dom}(y)$. And every action is $\mathbb{P}$-safe thus conditions for structural decomposability are fulfilled in this case.

Assume that $\operatorname{dom}(y) \subseteq \mathbb{P}_0 \setminus \mathbb{P}_1$ (the case $\operatorname{dom}(y) \subseteq \mathbb{P}_1 \setminus \mathbb{P}_0$ is symmetric). Since G_0 is structurally $\preceq_0$-decomposable, there exists $\mathbb{Q}_0 \subseteq \mathbb{P}_0 \setminus \mathbb{P}_1$ and a letter b of y such that:

$$\forall a \in A, \operatorname{dom}(a) \cap (\mathbb{P}_0 \setminus \mathbb{P}_1) \neq \emptyset \wedge a \mathbb{I} b \implies a \text{ is } \mathbb{Q}_0\text{-safe in } G_0 \tag{41}$$

$$\mathbb{Q}_0 \setminus \operatorname{dom}(b) \prec_0 \operatorname{dom}(y) \ . \tag{42}$$

Set $\mathbb{Q} = \mathbb{Q}_0 \cup \mathbb{P}_1$. Since $\operatorname{dom}(y) \subseteq (\mathbb{P}_0 \setminus \mathbb{P}_1)$ and $\mathbb{Q} \cap (\mathbb{P}_0 \setminus \mathbb{P}_1) = \mathbb{Q}_0$ then (42) and the definition of $\prec$ implies $\mathbb{Q} \setminus \operatorname{dom}(b) \prec \operatorname{dom}(y)$. We show that every letter $a \mathbb{I} b$ is $\mathbb{Q}$-safe for that we assume $\operatorname{dom}(a) \cap \mathbb{Q} \neq \emptyset$ and we prove that $\operatorname{dom}(a) \subseteq \mathbb{Q}$ or equivalently $\operatorname{dom}(a) \cap (\mathbb{P}_0 \setminus \mathbb{P}_1) \subseteq \mathbb{Q}_0$. If $\operatorname{dom}(a) \cap (\mathbb{P}_0 \setminus \mathbb{P}_1) = \emptyset$ there is nothing to prove. Otherwise since $\operatorname{dom}(a) \cap \mathbb{Q} \neq \emptyset$ then $\operatorname{dom}(a) \cap \mathbb{Q}_0 \neq \emptyset$. Moreover according to (41), a is $\mathbb{Q}_0$-safe in G_0 thus since $\operatorname{dom}(a) \cap \mathbb{Q}_0 \neq \emptyset$ then $\operatorname{dom}(a) \cap (\mathbb{P}_0 \setminus \mathbb{P}_1) \subseteq \mathbb{Q}_0$ which terminates to prove that every action $a \mathbb{I} b$ is $\mathbb{Q}$-safe. Thus G is structurally decomposable. □