On the Control of Asynchronous Automata

Hugo Gimbert

On the Control of Asynchronous Automata

Hugo Gimbert[1]

1 **LaBRI, CNRS, Université de Bordeaux, France**
hugo.gimbert@cnrs.fr

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.

1998 ACM Subject Classification B.1.2 Automatic synthesis, H.3.4 Distributed systems

Keywords and phrases asynchronous automata, Controller synthesis

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 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.

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. First, 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. Second, it seems that the results of the present paper can be lifted to parity games, using the same concepts but at the cost of some extra technical details needed to reason about infinite plays.

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.

Organization of the paper

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 prove 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.

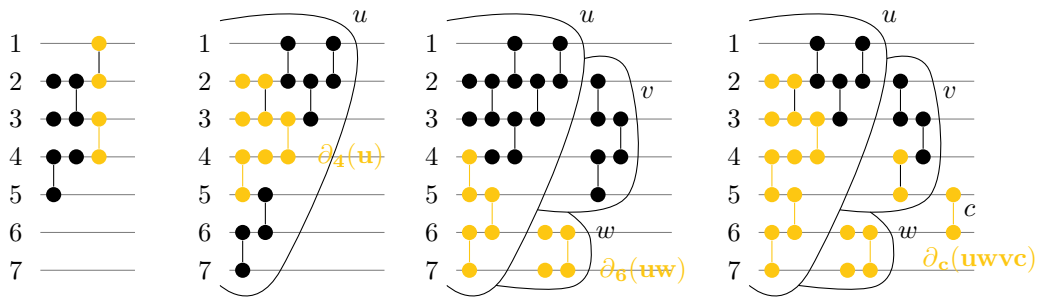

Figure 1 The set processes is $\{1 \dots 7\}$. A letter is identified with its domain. Here the domains are either singletons, represented by a single dot, or pairs of contigous processes, represented by two dots connected with a vertical segment. On the left handside is represented the trace $\{2\}\{3\}\{4,5\}\{2,3\}\{4\}\{1,2\}\{3,4\} = \{4,5\}\{4\}\{2\}\{3\}\{2,3\}\{3,4\}\{1,2\}$ which has two maximal letters $\{1,2\}$ and $\{3,4\}$ thus is not prime. Center left: process 4 sees only its causal view $\partial_4(u)$ (in yellow). Center right: $uvw = uwv$ since $\mathrm{dom}(v) \cap \mathrm{dom}(w) = \emptyset$. Both uv and $\partial_6(uw)$ (in yellow) are prime prefixes of uvw and they are parallel. Right: uv and $\partial_c(uvwc)$ (in yellow) are parallel.

2 The distributed synthesis problem

The theory of Mazurkiewicz traces is very rich, for a thorough presentation see [1]. Here we only fix notations and recall the notions of traces, views, prime traces and parallel traces.

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:

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

Maxima, prime traces and parallel traces

A letter $a \in A$ is a *maximum* of a trace u if it is the last letter of one of the linearizations of u . A trace $u \in A^*_{\equiv}$ is *prime* if it has a unique maximum, 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; and
- there is a trace w such that both u and v are prefixes of w.

These notions are illustrated on Fig. 1.

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 $\delta = (a, (q_p, q'_p)_{p\in\mathrm{dom}(a)}) \in \Delta$ and $\delta' = (a, (q_p, q''_p)_{p\in\mathrm{dom}(a)}) \in \Delta$ then $\delta = \delta'$ (hence $\forall p \in \mathrm{dom}(a), q'_p = q''_p$).

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 a mapping $\mathrm{state} : \mathrm{plays}(\mathcal{A}) \to \Pi_{p\in\mathbb{P}} Q_p$.

- ϵ is a play and $\mathrm{state}(\epsilon) = (i_p)_{p\in\mathbb{P}}$,
- for every play u such that $(\mathrm{state}_p(u))_{p\in\mathbb{P}}$ is defined and $\big(a, (\mathrm{state}_p(u), q_p)_{p\in\mathrm{dom}(a)}\big)$ is a transition 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 inductive definition of $\mathrm{state}(u)$ is correct because it is invariant by commutation of independent letters of 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

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

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 A_e, while they can decide whether to block or allow 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 since processes cannot communicate together unless they synchronize on a common action. In that case however 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** (Views)**.** For every set of processes $\mathbb{Q} \subseteq \mathbb{P}$ and trace u, the $\mathbb{Q}$-*view* of u, denoted $\partial_{\mathbb{Q}}(u)$, is the unique trace such that u factorizes as $u = \partial_{\mathbb{Q}}(u) \cdot v$ and v is the longest suffix of u such that $\mathbb{Q} \cap \mathrm{dom}(v) = \emptyset$. In case $\mathbb{Q}$ is a singleton $\{p\}$ the view is denoted $\partial_p(u)$ and is a prime trace. For every letter $a \in A$ we denote $\partial_a(u) = \partial_{\mathrm{dom}(a)}(u)$.

The well-definedness of the $\mathbb{Q}$-view is shown in the appendix, where we also establish:

$$\partial_{\mathbb{Q}}(uv) = \partial_{\mathbb{Q}'}(u)\,\partial_{\mathbb{Q}}(v) \text{ where } \mathbb{Q}' = \mathbb{Q} \cup \mathrm{dom}(\partial_{\mathbb{Q}}(v)) \tag{1}$$

$$(\mathbb{Q} \subseteq \mathbb{Q}') \implies (\partial_{\mathbb{Q}}(u) \sqsubseteq \partial_{\mathbb{Q}'}(u)) \ . \tag{2}$$

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 play 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:

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

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 uncontrollable action can happen.

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. $\forall p \in \mathbb{P}, \mathrm{state}_p(u) \in F_p$.

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 is positive, the algorithm should as well compute a winning strategy.

3 Three decidable classes

Series-parallel games

A game is *series-parallel* if its dependency alphabet (A, D) is a co-graph i.e. 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 is decidable for series-parallel games [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 further actions of q will be parallel to p. Formally, for every prime play uvw, $(q \notin \mathrm{dom}(v)$ and $|v|_p \geq k) \implies q \notin \mathrm{dom}(w)$.

The MSO theory of the event structure of a k-connectedly communicating game 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 the nodes of a tree $T_{\mathbb{P}}$ and the domain of every action is a connected set of nodes of $T_{\mathbb{P}}$. The synthesis problem is known to be decidable for acyclic games such that the domain of each action has size 1 or 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 view, 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 sub-strategy 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

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 prime play u such that there is no synchronization between $\mathbb{Q}$ and $\mathbb{P} \setminus \mathbb{Q}$ in parallel of u.

▶ **Definition 5.** 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 play u is a $\mathbb{Q}$-*lock* if it is prime and the last action of every prime play parallel to u is $\mathbb{Q}$-safe.

The notion of lock is illustrated on the right handside of Fig. 1. Set $\mathbb{Q} = \{1, 2, 3, 4, 5\}$. Then uv is not a $\mathbb{Q}$-lock because $\partial_c(uvwc)$ is parallel to uv but c is not $\mathbb{Q}$-safe. Locks occur in a variety of situations, including the three decidable classes.

▶ **Lemma 6** (Sufficient conditions for $\mathbb{Q}$-locks)**.** *Let u be a prime play of a game G 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* $\mathrm{dom}(\mathrm{last}(u))$.
vii) *There are two traces x and z such that $u = xz$ and z is a $\mathbb{Q}$-lock in the game G_x identical to G except the initial state is changed to* $\mathrm{state}(x)$.

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 *strategic state* as well as two operations on strategies called *shifting* and *projection*.

▶ **Definition 7** (Residual)**.** Let σ be a strategy, u a σ-play and $\mathbb{Q} \subseteq \mathbb{P}$. The $\mathbb{Q}$-residual of σ after u is the set:

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

A winning strategy may take unnecessarily complicated detours in order to ensure termination. Such detours are called *useless repetitions*.

▶ **Definition 8** (Strategic state)**.** Let $\mathbb{Q} \subseteq \mathbb{P}$ be a state of processes, σ a strategy and u a prime σ-play with maximal letter b. The strategic $\mathbb{Q}$-state of σ after u is the tuple

$$\mathrm{strate}_{\sigma,\mathbb{Q}}(u) = (b, \mathrm{state}(u), \pi\left(\sigma, u, \mathbb{Q} \setminus \mathrm{dom}(b)\right)) \ .$$

▶ **Definition 9** (Useless repetition)**.** A useless $\mathbb{Q}$-repetition in a strategy σ is a pair of traces (x, y) such that y is not empty, xy is a σ-play, $\mathrm{dom}(y) \subseteq \mathbb{Q}$, both x and xy are $\mathbb{Q}$-locks and $\mathrm{strate}_{\sigma,\mathbb{Q}}(x) = \mathrm{strate}_{\sigma,\mathbb{Q}}(xy)$.

The following theorem is the key to our decidability results.

▶ **Theorem 10.** *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 11** (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 may in general be infinite but is finite if σ is winning.

▶ **Lemma 12.** *Let (x, y) be a useless $\mathbb{Q}$-repetition in a strategy σ. Let $\Phi : A^*_{\equiv} \to A^*_{\equiv}$ and τ defined by* $\Phi(u) = \begin{cases} u \text{ if } x \not\sqsubseteq u \\ xyu' \text{ if } x \sqsubseteq u \text{ and } u = xu' \end{cases}$ *and*

$\forall p \in \mathbb{P}, \tau_p(u) = \sigma_p(\Phi(\partial_p(u)))$.

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

$$(u \text{ is a } \tau\text{-play}) \iff (\Phi(u) \text{ is a } \sigma\text{-play}) \ . \tag{3}$$

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

Sketch of proof of Lemma 12. The full proof can be found in the appendix. That τ is a strategy follows from the definition: $\tau_p(u)$ only depends on $\partial_p(u)$. To establish (3), the central point is to show that for every σ-play xu',

$$\forall p \in \mathbb{P}, \sigma_p(\partial_p(\Phi(xu')) = \sigma_p(\Phi(\partial_p(xu'))) \ .$$

There are three types of plays depending whether:
1. x has not occurred ($x \not\sqsubseteq u$),
2. x has occurred in parallel of the process p ($x \sqsubseteq u \wedge x \not\sqsubseteq \partial_p(u)$),
3. p knows that x has occurred ($x \sqsubseteq \partial_p(u)$).

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* create some τ-plays unrelated with σ. The equality of the strategic states in x and xy ensures that the equivalence (3) stays valid.

Moreover, thanks to (3), $\mathrm{dur}(\sigma) < \infty$ implies $\mathrm{dur}(\tau) < \mathrm{dur}(\sigma)$ because y is not empty. And according to (3) again, the set of global states of the maximal plays is the same for σ and τ thus if σ is winning then τ is winning as well. ◀

Proof of Theorem 10. As long as there exists a useless repetition, take the corresponding shortcut. According to Lemma 12, 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 (Theorem 21). There are actually three notions of decomposability: structural decomposability, process decomposability and action decomposability. These three notions form a hierarchy: structural decomposability implies process decomposability which itself implies action decomposability (Lemma 19). Known decidable classes are decomposable: acyclic games are structurally decomposable (Lemma 14), connectedly-communicating games are process decomposable (Lemma 16) and series-parallel games are action decomposable (Lemma 18). Structural decomposability is stable under some operations between games which leads to new examples of decidable games (Lemma 26).

5.1 Decomposability

The notions of decomposability rely on *preorders* defined on $2^{\mathbb{P}}$ or 2^A. A preorder $\preceq$ is a reflexive and transitive relation. We denote $\prec$ the relation $(x \prec y) \iff (x \preceq y \wedge y \not\preceq x)$.

Structural decomposability

The notion of structural decomposability relies on a preorder $\preceq$ on $2^{\mathbb{P}}$ which is monotonic with respect to inclusion, i.e. $\forall \mathbb{Q}, \mathbb{Q}' \subseteq \mathbb{P}$, $(\mathbb{Q} \subseteq \mathbb{Q}' \implies \mathbb{Q} \preceq \mathbb{Q}')$.

▶ **Definition 13** (Structural decomposability)**.** A game is $\preceq$-*structurally decomposable* if for every non-empty prime trace $y \in A^*$ there exists $\mathbb{Q} \supseteq \mathrm{dom}(y)$ and $b \in \mathrm{Alph}(y)$ such that:

$$(\mathbb{Q} \setminus \mathrm{dom}(b)) \prec \mathbb{Q}$$
$$\forall a \in A, (a \mathbb{I} b \implies a \text{ is } \mathbb{Q}\text{-safe}) \ .$$

We have already seen one example of structurally decomposable game.

▶ **Lemma 14.** *Acyclic games are structurally decomposable.*

Proof. Assume the game is acyclic with process tree $T_{\mathbb{P}}$. Set $\mathbb{Q} \preceq \mathbb{Q}'$ iff every process in $\mathbb{Q}$ has a $T_{\mathbb{P}}$-ancestor in $\mathbb{Q}'$, which is monotonic with respect to inclusion. Let y be a prime trace, $p \in \mathbb{P}$ the least common ancestor in $T_{\mathbb{P}}$ of processes in $\mathrm{dom}(y)$ and $\mathbb{Q}$ the set of descendants of p. Then $\mathrm{dom}(y) \subseteq \mathbb{Q}$. Moreover, 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 $p \in \mathrm{dom}(y)$ and there exists a letter $b \in \mathrm{Alph}(y)$ such that $p \in \mathrm{dom}(b)$. We show that b satisfies the conditions in the definition of structural decomposability. First, $(\mathbb{Q} \setminus \mathrm{dom}(b)) \preceq \mathbb{Q}$ and the inequality is strict because the only ancestor of p in $\mathbb{Q}$ is p itself and $p \in \mathrm{dom}(b)$. Second, let $a \in A$ such that $a \mathbb{I} b$. Then $p \notin \mathrm{dom}(a)$ and since $\mathrm{dom}(a)$ is connected in $T_{\mathbb{P}}$, then either none of the processes in $\mathrm{dom}(a)$ or all of them are descendants of p in $T_{\mathbb{P}}$, i.e. a is $\mathbb{Q}$-safe. ◀

Process decomposability

The definition of process decomposable games relies on a parameter $k \in \mathbb{N}$ and a preorder $\preceq$ on $2^{\mathbb{P}}$ which is monotonic with respect to inclusion.

▶ **Definition 15** (Process decomposable games)**.** Fix an integer k. A trace y is k-*repeating* if

$$y \text{ is not empty and } \forall p \in \mathrm{dom}(y), |y|_p \geq k \ .$$

A game is $(\preceq, k)$-*process decomposable* if for every prime play xy, if y is k-repeating then there exists $\mathbb{Q} \supseteq \mathrm{dom}(y)$ and a prime prefix $z \sqsubseteq y$ such that $\partial_{\mathrm{last}(z)}(xz)$ is a $\mathbb{Q}$-lock and

$$(\mathbb{Q} \setminus \mathrm{dom}(\mathrm{last}(z))) \prec \mathrm{dom}(y) \ . \tag{4}$$

We have already seen one example of process decomposable games.

▶ **Lemma 16.** *Connectedly communicating games are process decomposable.*

Action decomposability

Action decomposability is defined with respect to a parameter $k \in \mathbb{N}$ and a preorder $\preceq$ on 2^A which is monotonic with respect to inclusion.

▶ **Definition 17** (Action decomposable games)**.** Let k be an integer. A game is $(\preceq, k)$ *action decomposable* if for every prime play xy such that y is k-repeating, there exists $\mathbb{Q} \supseteq \mathrm{dom}(y)$ and a prime prefix $z \sqsubseteq y$ such that $\partial_{\mathrm{last}(z)}(xz)$ is a $\mathbb{Q}$-lock and

$$\{a \in A \mid \mathrm{dom}(a) \subseteq (\mathbb{Q} \setminus \mathrm{dom}(\mathrm{last}(z))\} \prec \mathrm{Alph}(y) \ .$$

We have already seen one example of action decomposable games.

▶ **Lemma 18.** *Series-parallel games are action decomposable.*

A hierarchy

▶ **Lemma 19.** *Every structurally decomposable game is process decomposable and every process decomposable game is action decomposable.*

Thus *action decomposability* is the most general notion of decomposability. In the sequel for the sake of conciseness, it is simply called *decomposability*.

5.2 Decidability

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

▶ **Lemma 20** (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 the appendix.

▶ **Theorem 21.** *The distributed synthesis problem is decidable for decomposable games.*

Proof of Theorem 21. 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 $\preceq$ be a preorder on 2^A compatible with inclusion, k' an integer and G a $(\preceq, k')$ action decomposable distributed game. Assume $k' = \mathrm{decomp}(G)$ w.l.o.g. (cf. Lemma 20).

For every set of actions $B \subseteq A$, denote G_B the game with actions B and the same processes, initial state and final states than G. The transitions of G_B are all transitions of G whose action is in B. An action $a \in B$ is controllable in G_B iff it is controllable in G.

We show that for every $B \subseteq A$ the game G_B is $(\preceq_B, k')$ decomposable, where $\preceq_B$ denotes the restriction of $\preceq$ to 2^B. Let xy be a prime play of G_B such that y is k'-repeating. Since G is $(\preceq, k')$ decomposable, there exists $\mathbb{Q} \supseteq \mathrm{dom}(y)$ and a prime prefix $z \sqsubseteq y$ such that $\partial_{\mathrm{last}(z)}(xz)$ is a $\mathbb{Q}$-lock in G and $C \prec \mathrm{Alph}(y)$ where $C = \{a \in A \mid \mathrm{dom}(a) \subseteq \mathbb{Q} \text{ and } a \mathbb{I} \mathrm{last}(z)\}$. Since $\preceq$ is monotonic with respect to inclusion then $\{b \in B \mid \mathrm{dom}(b) \subseteq \mathbb{Q} \text{ and } b \mathbb{I} \mathrm{last}(z)\} = (C \cap B) \preceq C \prec \mathrm{Alph}(y)$ thus $(C \cap B) \prec_B \mathrm{Alph}(y)$. Since xy is a play in G_B then $\partial_{\mathrm{last}(z)}(xz) \sqsubseteq xy$ is a play in G_B as well. And since every play in G_B is a play in G, $\partial_{\mathrm{last}(z)}(xz)$ is a $\mathbb{Q}$-lock not only in G but also in G_B. All conditions of action decomposability are met : G_B is $(\preceq_B, k')$ decomposable.

Denote $R_B(m)$ the largest size of a complete undirected graph whose edges are labelled with 2^B and which contains no monochromatic clique of size $\geq m$. According to Ramsey theorem, $R_B(m)$ is finite and computable. For every $B \subseteq A$, defined inductively $f(G_B)$ as :

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

with the convention $\max \emptyset = 0$.

Fix a strategy σ with no useless repetition. We prove that for every σ-play zu,

$$|u| \leq f\left(G_{\mathrm{Alph}(u)}\right) \quad . \tag{5}$$

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 let zu be a σ-play consistent with σ. Assume the induction hypothesis holds: for every σ-play $z'u'$, if $\mathrm{Alph}(u') \prec \mathrm{Alph}(u)$ then $|u'| \leq f\left(G_{\mathrm{Alph}(u')}\right)$.

We start with computing, for every non-empty set of letters $B \prec \mathrm{Alph}(u)$ an upper bound on the length of every factorization $u = u_0u_1 \cdots u_Nu_{N+1}$ such that

$$B = \mathrm{Alph}(u_1) = \mathrm{Alph}(u_2) = \ldots = \mathrm{Alph}(u_N) \ . \tag{6}$$

For a start, we consider the case where B is connected in the sense where the dependency graph $D_B = (B, D \cap B \times B)$ is connected. Set $k = k' + |\mathbb{P}|$. For $0 \leq \ell < \frac{N}{k}$, denote w_ℓ the concatenation $w_\ell = u_{1+\ell k} \cdot u_{2+\ell k} \cdots u_{k+\ell k}$ and $h_\ell = zu_0w_1 \ldots w_{\ell-1}$. Let $\mathbb{R}_B = \mathrm{dom}(B)$ and fix some $c \in B$.

Let $0 \leq \ell < \frac{N}{k}$. We show that $\partial_c(w_\ell)$ is k'-repeating and $\partial_c(h_\ell w_\ell) = \partial_{\mathbb{R}_B}(h_\ell)\, \partial_c(w_\ell)$. Since $w_\ell = u_{1+\ell k} \cdot u_{2+\ell k} \cdots u_{k+\ell k}$, according to property (1) of views there exists a sequence $\mathbb{P} \supseteq \mathbb{R}_1 \supseteq \ldots \supseteq \mathbb{R}_k$ such that

$$\partial_c(w_\ell) = \partial_{\mathbb{R}_1}(u_{1+\ell k})\, \partial_{\mathbb{R}_2}(u_{2+\ell k}) \cdots \partial_{\mathbb{R}_k}(u_{k+\ell k}) \tag{7}$$

where $\mathbb{R}_k = \{c\}$ and for every $1 \leq i \leq k-1$, $\mathbb{R}_i = \mathbb{R}_{i+1} \cup \mathrm{dom}(\partial_{\mathbb{R}_{i+1}}(u_{i+1+\ell k}))$. Since the sequence $(\mathbb{R}_i)_{1 \leq i \leq k'+|\mathbb{P}|}$ is monotonic, there exists $i \in k' \ldots k' + |\mathbb{P}|$ such that $\mathbb{R}_i = \mathbb{R}_{i+1}$. Denote $\mathbb{R} = \mathbb{R}_i = \mathbb{R}_{i+1}$ and $B' = \{b \in B, \mathrm{dom}(b) \cap \mathbb{R} \neq \emptyset\}$ and $B'' = \{b \in B, \mathrm{dom}(b) \subseteq \mathbb{R}\}$. By definition of views, and according to (6), $B' \subseteq \mathrm{Alph}(\partial_{\mathbb{R}}(u_{i+1+\ell k}))$. Since $\mathbb{R} = \mathbb{R}_i = \mathbb{R}_{i+1}$ and $\mathbb{R}_i = \mathbb{R}_{i+1} \cup \mathrm{dom}(\partial_{\mathbb{R}_{i+1}}(u_{i+1+\ell k}))$ then $\mathrm{dom}(\partial_{\mathbb{R}}(u_{i+1+\ell k})) \subseteq \mathbb{R}$ thus $\mathrm{Alph}(\partial_{\mathbb{R}}(u_{i+1+\ell k})) \subseteq B''$. Since $B'' \subseteq B'$ then finally $B' = \mathrm{Alph}(\partial_{\mathbb{R}}(u_{i+1+\ell k})) = B''$. Thus the set B'' is a connected component of the graph $D_B = (B, D \cap B \times B)$: by definition of B' and B'', all edges with source B'' have target in $B' = B''$. However by hypothesis D_B is connected thus $B = B' = B''$ and $\mathbb{R} = \mathbb{R}_B$. Finally $\mathbb{R}_B \subseteq \mathbb{R}_i \subseteq \mathbb{R}_1$ and since $\mathbb{R}_1 \subseteq \mathrm{dom}(\partial_c(w_\ell)) \subseteq \mathbb{R}_B$, the sequence $(\mathbb{R}_i)_{1 \leq i' \leq i}$ is constant equal to $\mathbb{R}_B$. Thus, according to (6) and the definition of $\mathbb{R}_B$, for every $1 \leq i' \leq i$, $\partial_{\mathbb{R}_{i'}}(u_{i'+\ell k}) = u_{i'+\ell k}$. Thus, according to (6) and (7) and since $k' \leq i'$, every letter of B occurs at least k' times in $\partial_c(w_\ell)$ thus $\partial_c(w_\ell)$ is k'-repeating and $\partial_c(h_\ell w_\ell) = \partial_{\mathbb{R}_B}(h_\ell)\, \partial_c(w_\ell)$.

Since the game is $(\preceq, k')$ decomposable and $\partial_c(w_\ell)$ is k'-repeating, and $\partial_c(h_\ell w_\ell) = \partial_{\mathbb{R}_B}(h_\ell)\, \partial_c(w_\ell)$, there exists a superset $\mathbb{T}^{(\ell)}$ of $\mathbb{R}_B$, an action b_ℓ, and a prime prefix $w'_\ell b_\ell \sqsubseteq \partial_c(w_\ell)$ such that the play $z_\ell = \partial_{b_\ell}(\partial_{\mathbb{R}_B}(h_\ell) w'_\ell b_\ell)$ is a $\mathbb{T}^{(\ell)}$-lock and $B_\ell \prec B$ where $B_\ell = \{a \in A \mid \mathrm{dom}(a) \subseteq (\mathbb{T}^{(\ell)} \setminus \mathrm{dom}(b_\ell))\}$.

For every $0 \leq \ell < \frac{N}{k}$, denote $\mathrm{strate}_\ell = \left(b_\ell, (s_{\ell,p})_{p \in \mathbb{P}}, \sigma^{(\ell)}\right)$ the $\mathbb{T}^{(\ell)}$ strategic state of σ after z_ℓ. We show two properties of $(\mathrm{strate}_\ell)_{0 \leq \ell < \frac{N}{k}}$.

- First, all elements of $(\mathrm{strate}_\ell)_{0 \leq \ell < \frac{N}{k}}$ are distinct. For the sake of contradiction, assume $\mathrm{strate}_\ell = \mathrm{strate}_{\ell'}$ for some $0 \leq \ell < \ell' < \frac{N}{k}$. We show that $z_\ell \sqsubset z_{\ell'}$. Since $\mathrm{strate}_\ell = \mathrm{strate}_{\ell'}$ then $b_\ell = b_{\ell'}$, denote this letter b. Then

$$\begin{aligned} z_\ell = \partial_b(\partial_{\mathbb{R}_B}(h_\ell)w'_\ell b) \sqsubseteq \partial_b(\partial_{\mathbb{R}_B}(h_\ell)\, \partial_c(w_\ell)) = \partial_b(\partial_c(h_\ell w_\ell)) \\ \sqsubseteq \partial_b(\partial_c(h_{\ell'})) \sqsubseteq \partial_b(\partial_{\mathbb{R}_B}(h_{\ell'})) \sqsubset \partial_b(\partial_{\mathbb{R}_B}(h_{\ell'})w'_{\ell'}b) = z_{\ell'} \ , \end{aligned}$$

  where the second inequality holds because $h_\ell w_\ell \sqsubseteq h_{\ell'}$ since $\ell \leq \ell' - 1$, and the third inequality holds because $c \in B$ thus $\mathrm{dom}(c) \subseteq \mathbb{R}_B$ hence property (2) applies. Moreover the last inequality is strict because there is at least one more b in $\partial_b(\partial_{\mathbb{R}_B}(h_{\ell'})w'_{\ell'}b)$ than in $\partial_b(\partial_{\mathbb{R}_B}(h_{\ell'}))$. We get a contradiction because by hypothesis there is no useless repetition in σ, however, denoting $x = z_\ell$ and y such that $xy = z_{\ell'}$, the pair (x, y) is a useless $\mathbb{T}^{(\ell)}$-repetition in σ: by hypothesis the strategic $\mathbb{T}^{(\ell)}$-states of z_ℓ and $z_{\ell'}$ are equal and both x and xy are $\mathbb{T}^{(\ell)}$-locks, moreover y is not empty because $z_\ell \sqsubset z_{\ell'}$ and finally $\mathrm{dom}(y) \subseteq \mathrm{dom}(u_{1+\ell k} \cdots u_{k+\ell' k}) \subseteq \mathbb{R}_B \subseteq \mathbb{T}^{(\ell)}$. Thus (x, y) is a useless repetition in σ.

- Second, for every $0 \leq \ell < \frac{N}{k}$, all plays in $\sigma^{(\ell)} = \pi(\sigma, z_\ell, \mathbb{T}^{(\ell)} \setminus \mathrm{dom}(b_\ell))$ have length $\leq m = \max_{B' \prec B} f(G_{B'})$. Let $z_\ell u'$ be a σ-play such that $\mathrm{dom}(u') \subseteq (\mathbb{T}^{(\ell)} \setminus \mathrm{dom}(b_\ell))$. Then $\mathrm{Alph}(u') \subseteq B_\ell$. Since $\preceq$ is monotonic with respect to inclusion, $\mathrm{Alph}(u') \preceq B_\ell \prec B \preceq \mathrm{Alph}(u)$. Thus by induction hypothesis, $|u'| \leq f\left(G_{\mathrm{Alph}(u')}\right) \leq m$.

According to the second property, there are at most $2^{2^{m|A||\mathbb{P}|}}$ different residuals appearing in the sequence $(\sigma^{(\ell)})_{0 \leq \ell < \frac{N}{k}}$. Thus the sequence $(\mathrm{strate}_\ell)_{0 \leq \ell < \frac{N}{k}}$ takes at most $K = |B| \cdot |Q|^{|\mathbb{P}|} \cdot 2^{2^{m|A||\mathbb{P}|}}$ different values. And according to the first property, all these states are different thus $N \leq k \cdot K$.

The inequality $N \leq k \cdot K$ has been established under the assumption that D_B is connected. The general case reduces to this case: let C be a connected component of D_B and for $1 \leq i \leq N$ let v_i be the projection of u_i on C. Then $\forall 1 \leq i \leq N, \mathrm{Alph}(v_i) = C$ and there exists u'_0 such that $u = u'_0 v_1 v_2 \dots v_N u_{N+1}$ thus $N \leq k \cdot K$.

Let us reformulate the inequality $N \leq k \cdot K$ as a property of an undirected complete graph with edges colored by 2^A. Let $u = a_1 a_2 \cdots a_{|u|}$ the factorization of u into its letters. Let J_u be the complete graph with vertices $1, \dots, |u|$ and the label of the edge $\{i, j\}$ with $i < j$ is the set of letters $\{a_i, \dots, a_j\}$. Then every monochromatic clique of J_u has size $\leq k \cdot K$. Thus, according to Ramsey theorem, $|u| \leq R_{\mathbb{T}}(k \cdot K) = R_{\mathbb{T}}((k' + |\mathbb{P}|) \cdot K)$, which completes the inductive step.

As a consequence, winning strategies in G can be looked for in the finite family of strategies whose all plays have length $\leq f(G)$ with $f(G)$ computable. As a consequence, the synthesis problem can be solved by enumerating all these strategies and testing whether any of them is winning. For testing whether a strategy of finite duration is winning the algorithm simply checks that the global state of all the maximal plays is final. ◀

5.3 New examples of decidable games

The three classes of games whose decidability is already known are decomposable (cf Lemmas 14, 16 and 18). In this section we give some new examples of decidable games.

▶ **Lemma 22.** *Four players games are structurally decomposable.*

Although our techniques do not seem to provide an algorithm for solving games with five processes, they can address a special case of those.

▶ **Lemma 23.** *Let G be a distributed game with five processes. 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.*

Another decidable example is the class of *majority games*:

▶ **Lemma 24** (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.*

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

▶ **Definition 25** (Projecting games)**.** Let G be a game with processes $\mathbb{P}$ and alphabet $(A_p)_{p \in \mathbb{P}}$. Let $\mathbb{P}' \subseteq \mathbb{P}$ a subset of the processes. The projection of G on $\mathbb{P}'$ is the game G' with processes $\mathbb{P}'$ and alphabet $A' = \{a \in A \mid \mathrm{dom}(a) \cap \mathbb{P}' \neq \emptyset\}$ partitioned in $(A' \cap A_p)_{p \in \mathbb{P}'}$. 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 combines two structurally decomposable games into one.

▶ **Lemma 26** (Merging games). *Let G be a game, and $\mathbb{P}_0, \mathbb{P}_1 \subseteq \mathbb{P}$ two set of processes such that $\mathbb{P} = \mathbb{P}_0 \cup \mathbb{P}_1$ and 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 can go beyond acyclic games, by merging together 4-player games and majority games. The graph of processes is an undirected graph with nodes $\mathbb{P}$ and there is an edge between p and q whenever both p and q both belong to the domain of one of the actions. Then all the games whose graph of processes is contained in the one depicted on Fig. 2 are structurally decomposable.

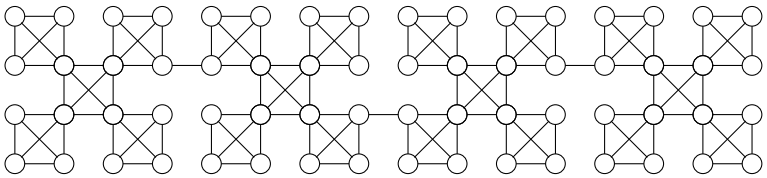

Figure 2 A decidable process architecture.

Conclusion

We considered the DISTRIBUTED SYNTHESIS PROBLEM, which aims at controlling asynchronous automata using automatically synthesized controllers with causal memory. We presented a theorem that unifies several known decidability results and provide new ones.

The decidability of this problem is still open to our knowledge, even in the simple case where the graph of processes is a ring of five processes where each process can interact only with both its neighbors.

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. Moreover we thank one of the reviewers of a previous version, who spotted several mistakes and did provide very useful comments which led to several improvements in the presentation of the results.

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 Definition of the $\mathbb{Q}$-view

For every set of processes $\mathbb{Q}$ and *word* $u \in A^*$, we define inductively the $\mathbb{Q}$-view of u as follows. If u is empty, its view is empty. If u is a word and a is a letter then:

$$\partial_{\mathbb{Q}}(ua) = \begin{cases} \partial_{\mathbb{Q}}(u) & \text{if } \mathrm{dom}(a) \cap \mathbb{Q} = \emptyset \\ \partial_{\mathbb{Q} \cup \mathrm{dom}(a)}(u)a & \text{if } \mathrm{dom}(a) \cap \mathbb{Q} \neq \emptyset \ . \end{cases} \tag{8}$$

An easy induction shows that for every words u, v,

$$\partial_{\mathbb{Q}}(uv) = \partial_{\mathbb{Q}'}(u)\, \partial_{\mathbb{Q}}(v) \text{ where } \mathbb{Q}' = \mathbb{Q} \cup \mathrm{dom}(\partial_{\mathbb{Q}}(v)) \ . \tag{9}$$

▶ **Lemma 27.** *Let $\mathbb{Q}$ be a set of processes and u, v two words and a, b two letters such that $a \mathbb{I} b$, $\partial_{\mathbb{Q}}(uabv) = \partial_{\mathbb{Q}}(ubav)$.*

Proof. According to (9),

$$\partial_{\mathbb{Q}}(uabv) = \partial_{\mathbb{Q}''}(u)\, \partial_{\mathbb{Q}'}(ab)\, \partial_{\mathbb{Q}}(v)$$

with $\mathbb{Q}' = \mathbb{Q} \cup \mathrm{dom}(\partial_{\mathbb{Q}}(v))$ and $\mathbb{Q}'' = \mathbb{Q}' \cup \mathrm{dom}(\partial_{\mathbb{Q}'}(ab))$.

Then $\partial_{\mathbb{Q}'}(ab) = \partial_{\mathbb{Q}'''}(a)\, \partial_{\mathbb{Q}'}(b)$ where $\mathbb{Q}''' = \mathbb{Q}' \cup \mathrm{dom}(\partial_{\mathbb{Q}'''}(a))$. However $a \mathbb{I} b$ thus $\mathrm{dom}(\partial_{\mathbb{Q}'''}(a)) \cap \mathrm{dom}(b) = \emptyset$ hence $\partial_{\mathbb{Q}'''}(a) = \partial_{\mathbb{Q}'}(a)$ thus $\partial_{\mathbb{Q}'}(ab) = \partial_{\mathbb{Q}'}(ba)$ and by symetry $\partial_{\mathbb{Q}}(uabv) = \partial_{\mathbb{Q}}(ubav)$. ◀

According to Lemma 27, the view is independent by commutation of independent letters, thus its definition extends to traces.

A simple induction provides several useful properties of views.

$$\partial_{\mathbb{Q}}(u) \sqsubseteq u \tag{10}$$
$$\partial_{\mathrm{dom}(u)}(u) = u \tag{11}$$
$$(\partial_{\mathbb{Q}}(u) = \epsilon) \iff (\mathrm{dom}(u) \cap \mathbb{Q} = \emptyset) \tag{12}$$
$$(\mathbb{Q} \subseteq \mathbb{Q}') \implies (\partial_{\mathbb{Q}}(u) \sqsubseteq \partial_{\mathbb{Q}'}(u)) \tag{13}$$
$$\partial_{\mathbb{Q}}(\partial_{\mathbb{Q}}(u)) = \partial_{\mathbb{Q}}(u) \tag{14}$$
$$\partial_{\mathbb{Q}}(uv) = \partial_{\mathbb{Q}}(u\, \partial_{\mathbb{Q}}(v)) \ . \tag{15}$$

To establish that the definition of views given in Definition 2 is equivalent to the one given by (8), we have to show:

▶ **Lemma 28.** *For every set of processes $\mathbb{Q}$, every trace u has a longest suffix v such that $\mathrm{dom}(v) \cap \mathbb{Q} = \emptyset$. And $u = \partial_{\mathbb{Q}}(u)v$.*

Proof. According to (10), there exists w such that $u = \partial_{\mathbb{Q}}(u)w$. We show that $\mathrm{dom}(w) \cap \mathbb{Q} = \emptyset$. According to (1), $\partial_{\mathbb{Q}}(u) = \partial_{\mathbb{Q}'}(u)\, \partial_{\mathbb{Q}}(w)$ with $\mathbb{Q}' = \mathbb{Q} \cup \mathrm{dom}(\partial_{\mathbb{Q}}(w))$. Since $\mathbb{Q}' \supseteq \mathbb{Q}$ then $|\partial_{\mathbb{Q}}(u)| \leq |\partial_{\mathbb{Q}'}(u)|$ according to (13) thus $|\partial_{\mathbb{Q}}(w)| = 0$ hence $\mathrm{dom}(w) \cap \mathbb{Q} = \emptyset$ according to (12). Let v be a suffix of u such that $\mathrm{dom}(v) \cap \mathbb{Q} = \emptyset$. Let u' such that $u = u'v$. Then $\partial_{\mathbb{Q}}(u) = \partial_{\mathbb{Q}}(u')$ thus $\partial_{\mathbb{Q}}(u) \sqsubseteq u'$ hence $|u'| \geq |u| - |w|$ hence $|v| \leq |w|$. ◀

7 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:

$$\forall u, v \in A^*, ((u \sqsubseteq v) \wedge (v \sqsubseteq u) \implies u = v) \ , \tag{16}$$
$$\forall u, v, w \in A^*, (uv = uw) \implies (v = w) \ , \tag{17}$$
$$\forall u, v, w \in A^*, (uv \sqsubseteq uw) \implies (v \sqsubseteq w) \ . \tag{18}$$

The following two lemmas list some basic properties of traces used in the proofs.

▶ **Lemma 29.** *Let u, v, x, y some traces such that $uv = xy$. Then there exists factorizations $x = x'x''$ and $y = y'y''$ such that:*

$$u = x'y' \tag{19}$$
$$v = x''y'' \tag{20}$$
$$\mathrm{dom}(x'') \cap \mathrm{dom}(y') = \emptyset \ . \tag{21}$$

Proof. By induction on $|uv|$. The case where uv is empty is trivial. Otherwise let a be a maximal action of v so that $v = v_0 a$. There are two cases.

If $\mathrm{dom}(a) \cap \mathrm{dom}(y) \neq \emptyset$ then since a is a maximal action of xy and does not commute with y, a is a maximal action of y. Thus y factorizes as $y = y_0 a$ and we apply the induction hypothesis to the equality $uv_0 = xy_0$ and append a to y''_0.

If $\mathrm{dom}(a) \cap \mathrm{dom}(y) = \emptyset$ then a is a maximal action of x which factorizes as $x = x_0 a$. We apply the induction hypothesis to the equality $uv_0 = x_0 y$ and append a to x''_0. ◀

We define the notion of $\mathbb{Q}$-prime trace.

▶ **Definition 30** ($\mathbb{Q}$-prime trace)**.** A trace is $\mathbb{Q}$-prime if $\partial_{\mathbb{Q}}(u) = u$.

We make use of the following properties of traces.

▶ **Lemma 31.** *For every trace $u, v, x \in A^*$ and $a \in A$ and $B \subseteq A$,*

$$uv \text{ is } \mathbb{Q}\text{-prime} \implies v \text{ is } \mathbb{Q}\text{-prime} \tag{22}$$
$$u \text{ and } v \text{ are } \mathbb{Q}\text{-prime} \implies uv \text{ is } \mathbb{Q}\text{-prime} \tag{23}$$
$$\text{If } ua \text{ is prime}, (av \text{ is } \mathbb{Q}\text{-prime} \iff uav \text{ is } \mathbb{Q}\text{-prime}) \tag{24}$$
$$(u \sqsubseteq \partial_{\mathbb{Q}}(uv)) \iff (\partial_{\mathbb{Q}}(uv) = u\, \partial_{\mathbb{Q}}(v)) \tag{25}$$
$$\text{If } ua \text{ is prime},$$
$$\partial_{\mathbb{Q}}(uav) = ua\, \partial_{\mathbb{Q}}(v) \iff \partial_{\mathbb{Q}}(av) = a\, \partial_{\mathbb{Q}}(v) \tag{26}$$
$$(\partial_{\mathbb{Q}}(uv) = \partial_{\mathbb{Q}}(u)) \implies \partial_{\mathbb{Q}}(v) = \epsilon \tag{27}$$
$$((a \sqsubseteq u) \wedge (x \sqsubseteq u) \wedge (a \not\sqsubseteq x)) \implies ((\mathrm{dom}(a) \cap \mathrm{dom}(x) = \emptyset) \wedge (ax \sqsubseteq u)) \ . \tag{28}$$

Proof. We prove (22). 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 (23). 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 (24). Assume ua prime. The converse implication follows from (22). 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 (22) 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,

$$bw = \{xbz \mid x, z \text{ words on } A \ , xz \in w, b \mathbb{I} x\} \ . \tag{29}$$

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 (29), 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 (24).

We prove (25). The converse implication in (25) is obvious so it is enough to prove the direct implication. Assume $u \sqsubseteq \partial_{\mathbb{Q}}(uv)$. According to (16) 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 (17) 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 (18) $w \sqsubseteq \partial_{\mathbb{Q}}(v)$. Finally $w = \partial_{\mathbb{Q}}(v)$ and $u\,\partial_{\mathbb{Q}}(v) = uw = u\,\partial_{\mathbb{Q}}(v)$ which terminates the proof of (25).

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 (26). Let $\mathbb{Q}' = \mathbb{Q} \cup \mathrm{dom}(\partial_{\mathbb{Q}}(v))$ Then according to (1), $\partial_{\mathbb{Q}}(uav) = \partial_{\mathbb{Q}'}(ua)\,\partial_{\mathbb{Q}}(v)$ 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).

We prove (27). Let $\mathbb{Q}' = \mathbb{Q} \cup \mathrm{dom}(\partial_{\mathbb{Q}}(v))$. Then according to (1), $\partial_{\mathbb{Q}}(uv) = \partial_{\mathbb{Q}'}(u)\,\partial_{\mathbb{Q}}(v)$. 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 $\partial_{\mathbb{Q}}(v) = \emptyset$.

Finally, we prove (28). Assume $a \sqsubseteq u$ and $x \sqsubseteq u$ and $a \not\sqsubseteq x$. We show that $\mathrm{dom}(a) \cap \mathrm{dom}(x) = \emptyset$ and $ax \sqsubseteq u$. Let u_1, u_2 such that $u = au_1$ and $u = xu_2$. Then according to Lemma 29, there exist factorizations $a = a'a''$ and $u_1 = u_3u_4$ such that $x = a'u_3$ and $u_2 = a''u_4$ and $\mathrm{dom}(a'') \cap \mathrm{dom}(u_3) = \emptyset$. Since a is a letter, either $(a' = a \wedge a'' = \epsilon)$ or $(a' = \epsilon \wedge a'' = a)$. However $a \not\sqsubseteq x$ thus $a' \neq a$ hence $(a' = \epsilon \wedge a'' = a)$. Thus $\mathrm{dom}(a) \cap \mathrm{dom}(x) = \mathrm{dom}(u_3) \cap \mathrm{dom}(x) = \emptyset$. And $u = xu_2 = xau_4 = axu_4$ thus $ax \sqsubseteq u$. ◀

▶ **Lemma 32.** *Let w be a trace and u and v two prefixes of w. The set of prefixes common to both u and v has a maximum (for the prefix relation) called the* longest common prefix *of u and v and denoted* $\mathrm{lcp}(u, v)$. *Let u'', v'' such that $u = \mathrm{lcp}(u, v)u''$ and $v = \mathrm{lcp}(u, v)v''$. Then* $\mathrm{dom}(u'') \cap \mathrm{dom}(v'') = \emptyset$.

Proof. The proof of the lemma is by induction on $|w|$.

The case where $|w| = 0$ is trivial, in this case $\mathrm{lcp}(u, v)$ is the empty trace.

Assume $|w| \geq 1$. Denote $L(u, v)$ the set of prefixes common to both u and v. Let a be a letter and w' be a trace such that $w = aw'$.

Assume $a \not\sqsubseteq u$ and $a \not\sqsubseteq v$ then u and v are two prefixes of w' and the proof of this case follows by induction.

If $a \sqsubseteq u$ and $a \not\sqsubseteq v$ then let $u = au'$. Then $L(u, v) = L(u', v)$. Since both v and u' are prefixes of w' then $\mathrm{lcp}(u', v)$ is inductively well-defined. Since $L(u, v) = L(u', v)$ then $\mathrm{lcp}(u, v) = \mathrm{lcp}(u', v)$ and the proof of this case follows by induction. The case $a \not\sqsubseteq u$ and $a \sqsubseteq v$ is symmetric.

If both $a \sqsubseteq u$ and $a \sqsubseteq v$ then let $u = au'$ and $v = av'$. Since both v' and u' are prefixes of w' then $\mathrm{lcp}(u', v')$ is inductively well-defined. Denote $\ell = a \cdot \mathrm{lcp}(u', v')$. Remark that $a \cdot L(u', v') \subseteq L(u, v)$ thus $\ell \in L(u, v)$ and ℓ is a good candidate for $\mathrm{lcp}(u, v)$. For that we show that every $z \in L(u, v)$ is a prefix of ℓ. There are two cases. First case: $a \sqsubseteq z$ thus there exists z' such that $z = az'$ then $az' \sqsubseteq au'$ and $az' \sqsubseteq av'$ thus $z' \in L(u', v')$ hence $z' \sqsubseteq \mathrm{lcp}(u', v')$ thus $z = az' \sqsubseteq a \cdot \mathrm{lcp}(u', v') = \ell$. Second case, $a \not\sqsubseteq z$. Then (28) implies $\mathrm{dom}(a) \cap \mathrm{dom}(z) = \emptyset$ and $az \in L(u, v)$. Thus $z \in L(u', v')$ hence $z \sqsubseteq \mathrm{lcp}(u', v')$ and there exists z' such that $\mathrm{lcp}(u', v') = zz'$. Then $\ell = a \cdot \mathrm{lcp}(u', v') = azz' = zaz'$ thus $z \sqsubseteq \ell$ which terminates the proof of the second case. Finally, every $z \in L(u, v)$ is a prefix of ℓ and $\ell \in L(u, v)$ thus $\mathrm{lcp}(u, v)$ is well-defined. The second statement follows easily by induction since $u' = \mathrm{lcp}(u', v')u''$ and $v' = \mathrm{lcp}(u', v')v''$. ◀

8 Properties of locks: proof of Lemma 6

Lemma 6. *Let u be a prime play of a game G 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 $\forall p \in \mathrm{dom}(u), |u|_p \geq k$.*

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* $\mathrm{dom}(\mathrm{last}(u))$.

vii) *There are two traces x and z such that $u = xz$ and z is a $\mathbb{Q}$-lock in the game G_x identical to G except the initial state is changed to* $\mathrm{state}(x)$.

Proof. For the remainder of the proof we fix v a prime play parallel to u and $c = \mathrm{last}(v)$. We denote $b = \mathrm{last}(u)$ (thus $c \mathbb{I} b$ since v is parallel to u). To show that u is a $\mathbb{Q}$-lock we need to show that c is $\mathbb{Q}$-safe. We prove that any of the conditions i) to vii) is sufficient to prove that c is $\mathbb{Q}$-safe.

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 (assume iv) holds) we distinguish between two cases. 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 of the decomposition tree containing both B and $\{c\}$. We show that C is a parallel node. Indeed C contains both $b \in B$ and c thus it is not a singleton hence not a leaf. By minimality of C, one son of C contains B while the other contains c. Then node C cannot be a serial product because $\mathrm{dom}(b) \cap \mathrm{dom}(c) = \emptyset$ and one son of C contains $b \in B$ while the other contains c. Thus $\mathrm{dom}(B) \cap \mathrm{dom}(c) = \emptyset$ and c is $\mathbb{Q}$-safe.

For connectedly communicating games (assume (v) holds), we establish that c is $\mathbb{Q}$-safe by showing that $(\mathrm{dom}(c) \cap \mathbb{Q} \neq \emptyset) \implies (\mathrm{dom}(c) \subseteq \mathbb{Q})$. Let $p \in \mathrm{dom}(c) \cap \mathbb{Q}$. Since u and v are parallel there exists a play w such that both $u \sqsubseteq w$ and $v \sqsubseteq w$. Then $\partial_p(u) \sqsubseteq \partial_p(w)$ and $\partial_p(v) \sqsubseteq \partial_p(w)$. Since $c = \mathrm{last}(v)$ and $p \in \mathrm{dom}(c)$ then $\partial_p(v) = v$ thus $v \sqsubseteq \partial_p(w)$, which we reuse later. Let w' such that $\partial_p(w) = \partial_p(u)w'$. By hypothesis, $|u|_p \geq k$ thus $|\,\partial_p(u)|_p \geq k$. By definition of connectedly communicating games, since $\partial_p(w)$ is prime, $\partial_p(w) = \partial_p(u)w'$ and $|\,\partial_p(u)|_p \geq k$ then $\mathrm{dom}(w') \subseteq \mathrm{dom}(\partial_p(u))$ thus $\mathrm{dom}(\partial_p(w)) \subseteq \mathrm{dom}(\partial_p(u))$. Since $v \sqsubseteq \partial_p(w)$, we get $\mathrm{dom}(v) \subseteq \mathrm{dom}(\partial_p(u)) \subseteq \mathrm{dom}(u) = \mathbb{Q}$. In particular $\mathrm{dom}(c) \subseteq \mathbb{Q}$ thus c is $\mathbb{Q}$-safe which terminates the proof in case (v) holds.

For acyclic games (assume (vi) holds), we show that c is $\mathbb{Q}$-safe as follows. By definition of acyclic games, the domain of every action is connected in $T_{\mathbb{P}}$. Let $p \in \mathrm{dom}(b)$ be of minimal depth in the tree $T_{\mathbb{P}}$ among the processes in $\mathrm{dom}(b)$. Then $\mathbb{Q}$ is the set of descendants of p in $T_{\mathbb{P}}$. Since u and v are parallel then $c \mathbb{I} b$ thus $p \notin \mathrm{dom}(c)$. Since $\mathrm{dom}(c)$ is connected in $T_{\mathbb{P}}$ then either all processes in $\mathrm{dom}(c)$ are descendants of p or none of them are. In other words $(\mathrm{dom}(c) \subseteq \mathbb{Q}) \vee (\mathrm{dom}(c) \cap \mathbb{Q} = \emptyset)$ i.e. c is $\mathbb{Q}$-safe.

Now assume property (vii) holds. We show that there exists x', x'', z', z'', v' such that:

$$x = x'x'' \tag{30}$$
$$z = z'z'' \tag{31}$$
$$v = x'z'v' \tag{32}$$
$$\mathrm{dom}(v') \cap \mathrm{dom}(x''z'') = \emptyset \tag{33}$$
$$\mathrm{dom}(x'') \cap \mathrm{dom}(z') = \emptyset \ . \tag{34}$$

For that let y be the longest common prefix of $u = xz$ and v i.e. $y = \mathrm{lcp}(xz, v)$. According to Lemma 32, there exists y' and v' such that $yy' = xz$, $yv' = v$ and $\mathrm{dom}(y') \cap \mathrm{dom}(v') = \emptyset$. Since $yy' = xz$, according to Lemma 29 there exists factorizations $x = x'x''$ and $z = z'z''$ such that $y = x'z'$ and $y' = x''z''$ and $\mathrm{dom}(x'') \cap \mathrm{dom}(z') = \emptyset$.

Now we prove that $xz'v'z''$ is a play. First, $xz = x'x''z'z'' = x'z'x''z''$ thus since xz is a play, $x'z'x''z''$ is also a play. And by hypothesis $v = x'z'v'$ is also a play. Set $w = x'z'$ then to summarize both $wx''z''$ and wv' are plays. Since the processes playing in $x''z''$ and v' are distinct (cf. (33)) then $wv'x''z'' = x'z'v'x''z''$ is also a play. And according to (33) and (34), $x'z'v'x''z'' = x'z'x''v'z'' = x'x''z'v'z'' = xz'v'z''$. Thus $xz'v'z''$ is a play.

Now, we show that $w = z'v'$ and $z = z'z''$ are two parallel prime plays in the game G_x with initial state $\mathrm{state}(x)$. Remark first that both w and z are a prefix of $wz'' = z'v'z'' = zv'$. And since $xz'v'z'' = xwz''$ is a play in G (cf supra) then wz'' is a play in G_x, thus both prefixes w and z are plays in G_x as well. Since w is a suffix of the prime trace $v = x'z'v' = x'w$, it is prime with maximal action $c = \mathrm{last}(v)$. Since z is a suffix of the prime trace $u = xz$, it is prime with maximal action $b = \mathrm{last}(u)$. Hence w and z are two parallel plays in G_x.

By hypothesis, z is a $\mathbb{Q}$-lock in G_x thus $c = \mathrm{last}(w)$ is $\mathbb{Q}$-safe. ◀

9 Taking shortcuts: proof of Lemma 12

Lemma 12. *Let (x, y) be a useless $\mathbb{Q}$-repetition in a strategy σ. Let $\Phi : A^*_{\equiv} \to A^*_{\equiv}$ and τ defined by* $\Phi(u) = \begin{cases} u \text{ if } x \not\sqsubseteq u \\ xyu' \text{ if } x \sqsubseteq u \text{ and } u = xu' \end{cases}$ *and*

$\forall p \in \mathbb{P}, \tau_p(u) = \sigma_p(\Phi(\partial_p(u)))$.

Then τ is a strategy called the (x, y)-shortcut of σ. *And, for every trace u,*

$$(u \text{ is a } \tau\text{-play}) \iff (\Phi(u) \text{ is a } \sigma\text{-play}) \ . \tag{35}$$

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

Proof. That τ is a strategy follows from the definition: $\tau_p(u)$ only depends on $\partial_p(u)$.

Let $b = \text{last}(x)$. Since (x, y) is a useless $\mathbb{Q}$-repetition then

$$\text{both } x \text{ and } xy \text{ are } \mathbb{Q}\text{-locks, in particular they are prime,} \tag{36}$$
$$\text{last}(x) = \text{last}(xy) = b \tag{37}$$
$$\text{dom}(y) \subseteq \mathbb{Q} \tag{38}$$
$$\text{state}(x) = \text{state}(xy) \tag{39}$$
$$\pi\left(\sigma, x, \mathbb{Q} \setminus \text{dom}(b)\right) = \pi\left(\sigma, xy, \mathbb{Q} \setminus \text{dom}(b)\right) \ . \tag{40}$$

Proof of property (35). We start with a preliminary lemma.

▶ **Lemma 33.** *For every σ-play xu',*

$$\forall p \in \mathbb{P}, \sigma_p(\partial_p(\Phi(xu')) = \sigma_p(\Phi(\partial_p(xu'))) \ . \tag{41}$$

Proof. First notice that:

$$\begin{aligned} x \sqsubseteq \partial_p(xu') &\iff \partial_p(xu') = x\,\partial_p(u') \\ &\iff b \sqsubseteq \partial_p(bu') \iff \text{dom}(b) \cap \text{dom}(\partial_p(u')) \neq \emptyset \\ &\iff \partial_p(xyu') = xy\,\partial_p(u') \\ &\iff xy \sqsubseteq \partial_p(xyu'), \end{aligned} \tag{42}$$

which comes from applications of (25) and (26) and (37) and property (9) of views.

To show (41), we consider several cases.

First case. Assume $x \sqsubseteq \partial_p(xu')$. Then according to (42), $\Phi(\partial_p(xu')) = \Phi(x\,\partial_p(u')) = xy\,\partial_p(u') = \partial_p(xyu') = \partial_p(\Phi(xu'))$ and in this case (41) holds.

Second case. Assume $x \not\sqsubseteq \partial_p(xu')$ and $\text{dom}(\partial_p(u')) \subseteq \mathbb{Q}$. Remark first that according to (42) $\text{dom}(b) \cap \text{dom}(\partial_p(u')) = \emptyset$ thus $\text{dom}(\partial_p(u')) \subseteq (\mathbb{Q} \setminus \text{dom}(b))$. Since both $x\,\partial_p(u') \sqsubseteq xu'$ and $xy\,\partial_p(u') \sqsubseteq xyu'$ are σ-plays, we can apply (40) and get $\sigma_p(x\,\partial_p(u')) = \sigma_p(xy\,\partial_p(u'))$. Thus (41) holds since

$$\begin{aligned} \sigma_p(\partial_p(\Phi(xu'))) &= \sigma_p(\partial_p(xyu')) \\ &= \sigma_p(\partial_p(xy\,\partial_p(u'))) \\ &= \sigma_p(xy\,\partial_p(u')) \\ &= \sigma_p(x\,\partial_p(u')) \\ &= \sigma_p(\partial_p(x\,\partial_p(u'))) \\ &= \sigma_p(\partial_p(xu')) \\ &= \sigma_p(\Phi(\partial_p(xu'))) \ , \end{aligned}$$

where the equalities hold because σ is a distributed strategy, according to property (15) of views and because $x \not\sqsubseteq \partial_p(xu')$ thus $\Phi(\partial_p(xu')) = \partial_p(xu')$.

Third case. Assume $x \not\sqsubseteq \partial_p(xu')$ and $\mathrm{dom}(\partial_p(u')) \not\subseteq \mathbb{Q}$. We show by contradiction that $\mathrm{dom}(\partial_p(u')) \cap \mathbb{Q} = \emptyset$. Otherwise, since $\partial_p(u')$ is prime there would exists some letter $d \in \mathrm{Alph}(\partial_p(u'))$ such that $\mathrm{dom}(d)$ intersects both $\mathbb{Q}$ and $\mathbb{P} \setminus \mathbb{Q}$. Let $w = \partial_d(\partial_p(xu'))$. Remark that $w \neq \epsilon$ and a fortiori $\partial_p(u') \neq \emptyset$.

We show that $w \not\sqsubseteq x$ by contradiction. Otherwise $wu' \sqsubseteq xu'$ hence $\partial_d(\partial_p(wu')) \sqsubseteq \partial_d(\partial_p(xu')) = w$. Let $\mathbb{R} = \mathrm{dom}(\partial_p(u'))$. Since $\partial_p(u') \neq \emptyset$ then $p \in \mathbb{R}$ thus according to (9), $\partial_p(wu') = \partial_{\mathbb{R}}(w)\,\partial_p(u')$. And since w is d-prime and $\mathrm{dom}(d) \subseteq \mathbb{R}$ then $\partial_{\mathbb{R}}(w) = w$ thus $\partial_p(wu') = w\,\partial_p(u')$. Then since w is d-prime, $\partial_d(\partial_p(wu')) = w\,\partial_d(\partial_p(u'))$. But then $\partial_d(\partial_p(wu')) \sqsubseteq w$ shown above implies $\partial_d(\partial_p(u')) = \epsilon$, a contradiction with $d \in \mathrm{Alph}(\partial_p(u'))$.

Thus $w \not\sqsubseteq x$ hence $w = \partial_d(\partial_p(xu')) \sqsubseteq xu'$ is a prime play parallel to $x \sqsubseteq xu'$ with maximal action d. However d is not $\mathbb{Q}$-safe, contradicting the hypothesis that x is a $\mathbb{Q}$-lock (cf. (36)). Thus $\mathrm{dom}(\partial_p(u')) \cap \mathbb{Q} = \emptyset$.

Since $\mathrm{dom}(\partial_p(u')) \cap \mathbb{Q} = \emptyset$ and $\mathrm{dom}(y) \subseteq \mathbb{Q}$ (cf. (38)) then $\partial_p(xyu') = \partial_p(xu')$ according to the property (9) of views. Thus (41) holds since

$$\sigma_p(\partial_p(\Phi(xu'))) = \sigma_p(\partial_p(xyu')) = \sigma_p(\partial_p(xu')) = \sigma_p(\Phi(\partial_p(xu'))) \ ,$$

where the last equality holds since $x \not\sqsubseteq \partial_p(xu')$ thus $\Phi(\partial_p(xu')) = \partial_p(xu')$. This completes the proof of (41). ◀

We prove (35) by induction on u. The base case $u = \epsilon$ holds because $\Phi(\epsilon) = \epsilon$ and ϵ is consistent with every strategy. Assume (35) holds for u and all its prefixes and let c be a letter. We show that (35) holds for uc as well.

We start with the direct implication. Assume that uc is a τ-play. We have to show

$$\Phi(uc) \text{ is a } \sigma\text{-play.} \tag{43}$$

Since uc is a τ-play then u is a τ-play thus by induction hypothesis $\Phi(u)$ is a σ-play. And since uc is a τ-play then

$$\forall p \in \mathrm{dom}(c), c \in \tau_p(u) \ . \tag{44}$$

To show (43) we distinguish between several cases.

First case: assume $x \not\sqsubseteq uc$. Then $\Phi(uc) = uc$ then a fortiori $x \not\sqsubseteq \partial_p(u)$ thus $\Phi(\partial_p(u)) = \partial_p(u)$. Hence $\forall p \in \mathrm{dom}(c), \tau_p(u) = \sigma_p(\Phi(\partial_p(u))) = \sigma_p(\partial_p(u)) = \sigma_p(u)$. Hence according to (44), $\forall p \in \mathrm{dom}(c), c \in \sigma_p(u)$. Since $u = \Phi(u)$ then u is a σ-play hence by definition of σ-plays, uc as well is a σ-play. Thus (43) holds in this case.

Second case: assume $x \sqsubseteq uc$ and $x \not\sqsubseteq u$. Then $x = uc$ and c is the maximal letter of x. Then $\Phi(uc) = \Phi(x) = xy$. Since (x, y) is a σ-repetition then xy is a σ-play thus (43) holds.

Third case: assume $x \sqsubseteq u$. Let u' such that $u = xu'$. By induction hypothesis, $\Phi(u) = xyu'$ is a σ-play thus to show that $\Phi(uc) = xyu'c$ is a σ-play, it is enough to prove

$$\forall p \in \mathrm{dom}(c), c \in \sigma_p(\partial_p(xyu')) = \sigma_p(\partial_p(\Phi(u)) \ . \tag{45}$$

We show first that

$$\forall p \in \mathrm{dom}(c), c \in \sigma_p(\Phi(\partial_p(u))) \ . \tag{46}$$

This holds because $u = xu'$ is a τ-play thus $\partial_p(u) \sqsubseteq u$ as well is a τ-play and by induction hypothesis, $\Phi(\partial_p(u))$ is a σ-play. Thus by definition of τ, $\tau_p(u) = \sigma_p(\Phi(\partial_p(u)))$ thus (46) holds according to (44). Then (46) and Lemma 33 show that (45) holds. This completes the proof of the direct implication of (35).

Now we show the converse implication of (35). Assume that $\Phi(uc)$ is a σ-play. We have to show that uc is a τ-play.

There are two cases. If $x \not\sqsubseteq u$. Then $\Phi(u) = u$ thus by induction hypothesis, u is both a τ-play and a σ-play. Moreover σ and τ coincide on u thus since uc is a σ-play then uc is a τ-play as well.

If $x \sqsubseteq u$. Let u' such that $u = xu'$. Then both $\Phi(u) = xyu'$ and $\Phi(uc) = xyu'c$ are σ-plays. By induction hypothesis, xu' is a τ-play thus to show that $xu'c$ is a τ-play we shall show

$$\forall p \in \mathrm{dom}(c), c \in \tau_p(xu') = \sigma_p(\Phi(\partial_p(xu'))) \ . \tag{47}$$

Since $xyu'c$ is a σ-play then

$$\forall p \in \mathrm{dom}(c), c \in \sigma_p(xyu') = \sigma_p(\partial_p(\Phi(xu'))) \ ,$$

hence (47) holds according to Lemma 33. This terminates the proof of the converse implication of (35). ◀

Proof that τ is winning. Since σ is winning, the set of σ-plays is finite. According to property (35) 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 (39) the global state is the same in x and xy. Since transitions are deterministic, all processes are in the same state in xw and xyw. According to (35), 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 12 have been proved. ◀

10 Decomposability

10.1 Connectedly communicating games: proof of Lemma 16

Lemma 16. *Connectedly communicating games are process decomposable.*

Proof. Let $\preceq$ be the inclusion preorder. We assume G is k-connectedly communicating and show that G is $(\preceq, k)$-process decomposable. Let xy be a prime play of G such that y is k-repeating. We set $\mathbb{Q} = \mathrm{dom}(y)$ and $z = y$ and show that both conditions in the definition of process decomposability are satisfied. Let $b = \mathrm{last}(y)$. Then $\mathrm{dom}(b) \subseteq \mathbb{Q}$ thus $(\mathbb{Q} \setminus \mathrm{dom}(b)) \subsetneq \mathbb{Q}$ and condition (4) is satisfied. To show that $\partial_b(xy)$ is a $\mathbb{Q}$-lock in G, notice first that since xy is prime then $xy = \partial_b(xy)$.

Denote G_x the game identical to G except the initial state is $\mathrm{state}(x)$. Then according to Lemma 6, xy is a $\mathbb{Q}$-lock in G: according to v) applied to G_x the play y is a $\mathbb{Q}$-lock in G_x hence according to vii) of the same lemma, xy is is a $\mathbb{Q}$-lock in G. ◀

10.2 Series-parallel games: proof of Lemma 18

Lemma 18. *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. We denote $B_\uparrow$ the smallest node of this branch and moreover we set $\emptyset_\uparrow = \emptyset$. The preorder $\preceq$ on 2^A is defined as $B \preceq B' \iff B_\uparrow \subseteq B'_\uparrow$.

Let xy be a prime play such that y is not empty. Set $\mathbb{Q} = \mathrm{dom}(\mathrm{Alph}(y)_\uparrow)$ and $z = y$. We show that both conditions in the definition of action decomposable games are satisfied. Let G_x be the game obtained by changing the initial state to state(x). According to property iv) of Lemma 6, y is a $\mathbb{Q}$-lock in G_x. Hence according to property vii) of Lemma 6, xy is a $\mathbb{Q}$-lock in G. Let $b = \mathrm{last}(y)$ and $A' = \{a \in A \mid \mathrm{dom}(a) \subseteq (\mathbb{Q} \setminus \mathrm{dom}(b))\}$, we show that $A' \prec \mathrm{Alph}(y)$.

We start with a preliminary remark. Let $C \subseteq A$. We say that C is connected if C induces a connected subset of the dependency graph of the alphabet, i.e. if the graph with nodes C and edges $D \cap C \times C$ is connected. If C is connected then $C_\uparrow$ is either a product node or a leaf of T_A, because if B is the parallel product of B_1 and B_2 and $C \subseteq B$ then either $C \subseteq B_1$ or $C \subseteq B_2$.

In particular, since y is prime then $\mathrm{Alph}(y)$ is connected 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 $A' = \emptyset \prec \mathrm{Alph}(y)$. In the second case w.l.o.g. assume that $b \in B$. Then no action of C is independent of b thus $A' \subseteq B$. Then $B_\uparrow = B \subsetneq \mathrm{Alph}(y) \preceq \mathrm{Alph}(y)_\uparrow$ thus $B \prec \mathrm{Alph}(y)$ hence $A' \prec \mathrm{Alph}(y)$. ◀

10.3 A hierarchy: proof of Lemma 19

Lemma 19. *Every structurally decomposable game is process decomposable and every process decomposable game is action decomposable.*

Proof. For the first implication, fix a preorder $\preceq$ on $\mathbb{P}$ which is monotonic with respect to inclusion and witnesses the structural decomposability of the game. We show that the game is process decomposable with parameter $(1, \preceq)$. Let xy be a prime play. Then the suffix y is prime. By definition of structural decomposability, there exists $\mathbb{Q} \supseteq \mathrm{dom}(y)$ and a letter $b \in \mathrm{Alph}(y)$ such that (H1) $(\mathbb{Q} \setminus \mathrm{dom}(b)) \prec \mathbb{Q}$ and (H2) $\forall a \in A, (a \mathbb{I} b \implies a \text{ is } \mathbb{Q}\text{-safe})$. Let z be a prime prefix of y with maximal letter b, which exists since $b \in \mathrm{Alph}(y)$ thus y factorizes as $y = y'by''$ and we can choose $z = \partial_b(y'b)$. Then (H2) implies that $\partial_b(xz)$ is a $\mathbb{Q}$-lock and (H1) is exactly condition (4) in the definition of process decomposability thus all conditions for process decomposability are met.

Now assume the game is process decomposable with parameters $(k, \preceq_{\mathbb{P}})$. We define the preorder $\preceq_A$ on 2^A by $(B \preceq_A B') \iff (\mathrm{dom}(B) \preceq_{\mathbb{P}} \mathrm{dom}(B'))$. Then every $(k, \preceq_{\mathbb{P}})$ process decomposable game is $(k, \preceq_A)$ action decomposable because, $\forall b \in A$,

$$\mathrm{dom}\left(\{a \in A \mid \mathrm{dom}(a) \subseteq \mathbb{Q} \text{ and } a \mathbb{I} b\}\right) \subseteq (\mathbb{Q} \setminus \mathrm{dom}(b))$$

and, as a consequence, for every trace y,

$$((\mathbb{Q} \setminus \mathrm{dom}(b)) \prec_{\mathbb{P}} \mathrm{dom}(y)) \implies (\{a \in A \mid \mathrm{dom}(a) \subseteq (\mathbb{Q} \setminus \mathrm{dom}(b)\} \prec_A \mathrm{Alph}(y)) \ .$$

◀

11 Decidability of decomposability: proof of Lemma 20

Lemma 20. *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.*

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

▶ **Definition 34** ($\mathbb{Q}$-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 $\mathrm{dom}(v) \cap \mathbb{Q} = \emptyset$.

We reformulate what it means for a game *not* to be decomposable, in terms of computations of asynchronous automata. For every $b \in A$, denote $\mathcal{A}_b$ the automaton identical to $\mathcal{A}$ except it is restricted to letters whose domain do not intersect $\mathrm{dom}(b)$: other letters are removed from the alphabet and the corresponding transitions are deleted.

▶ **Lemma 35.** *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 $\mathcal{A}_b$.*

Proof. Reformulation of the definition of $\mathbb{Q}$-locks. ◀

Let $\preceq$ be a preorder on 2^A compatible with inclusion.
For every letter b and subset of processes $\mathbb{Q}$ denote

$$A_{\mathbb{Q},b} = \{a \in A \mid \mathrm{dom}(a) \subseteq (\mathbb{Q} \setminus \mathrm{dom}(b)\}$$
$$\mathcal{C} = \{(B, b, (q_p)_{p\in\mathbb{P}}) \mid B \subseteq A, b \in B, \exists \mathbb{Q} \supseteq \mathrm{dom}(B), A_{\mathbb{Q},b} \prec B \text{ and } (q_p)_{p\in\mathbb{P}} \text{ is } \mathbb{Q}\text{-locked in } \mathcal{A}_b\} \quad .$$

Remark that $\mathcal{C}$ is computable because 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.

Denote $\mathcal{A}'$ the synchronous automaton reading finite words in A^* which computes on-the-fly the global state of $\mathcal{A}$ 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 $\mathcal{A}'$ accessible from the initial state by a *prime* trace. The following properties are equivalent.

i) There exists $k \in \mathbb{N}$ such that the game is $(\preceq, k)$ action decomposable.
ii) There exists $k \in \mathbb{N}$ such that for every prime play xy, if y is k-repeating there exists a prime prefix $z \sqsubseteq y$ such that $(\mathrm{Alph}(y), \mathrm{last}(z), \mathrm{state}(\partial_{\mathrm{last}(z)}(xz))) \in \mathcal{C}$.

Property ii) is actually a simple reformulation of i) based on Lemma 35 and the definition of $\mathcal{C}$. We show that it is decidable. For that we characterize it using the notion of *non-decomposability witness*.

Fix some word $x \in A^*$ and $B \subseteq A$. We say that a word $y \in A^*$ is a non-decomposability witness for (x, B) if:
a) $\mathrm{Alph}(y) = B$,
b) the trace $(xy)_\equiv$ is prime,
c) $y_\equiv$ has no prime prefix $z_\equiv \sqsubseteq y_\equiv$ such that

$$(\mathrm{Alph}(y), \mathrm{last}(z_\equiv), \mathrm{state}(\partial_{\mathrm{last}(z_\equiv)}(x_\equiv z_\equiv))) \in \mathcal{C} \quad . \tag{48}$$

We show that the language $L_{x,B}$ of non-decomposability witness for (x, B) is a regular language of finite words. Condition a) is clearly regular. Condition b) can be checked with $\mathcal{A}'$, initialized with state(x) and the list of maximal actions in x: for $(xy)_{\equiv}$ to be prime. there should be a unique maximal action in this modified version of $\mathcal{A}'$ after reading y. To show that condition c) is regular, we show that the set of mirror images of words y *not* satisfying c) is regular. While reading the mirror image of y, the automaton guesses on-the-fly the sequence of global states and transitions performed by the automaton, which should end-up in state(x) once the first letter of y has been read. The automaton picks non-deterministically at some moment the last letter c of $z_{\equiv}$ and simultaneously guesses $q = \text{state}(\partial_c(x_{\equiv} z_{\equiv}))$ under the constraint $(\text{Alph}(y), c, q) \in \mathcal{C}$. From then on the automaton keeps reading y backwards and computes on-the-fly $\partial_c(x_{\equiv} z_{\equiv})$ using the inductive definition of the view, see (8). This way the automaton can check that q is equal to state$(\partial_c(x_{\equiv} z_{\equiv}))$.

Now we show that property ii) is decidable. Note that $L_{x,B}$ actually does depend only on B (condition a)), on the set of maximal actions in $x_{\equiv}$ (condition b)) and on $(\partial_p(x_{\equiv}))_{p\in\mathbb{P}}$ (condition c)). Thus the collection of possible languages $L_{x,B}$ can be explicitely computed as a finite collection $(L_{x_i,B_i})_{1\leq i\leq M}$, together with the corresponding finite collection of automata $(\mathcal{A}_{x_i,B_i})_{1\leq i\leq M}$.

And property ii) holds if and only if for every language $L_{x,B}$,

$$f(L_{x,B}) = \sup_{y\in L_{x,B}} \min_{b\in B} |y|_b$$

is finite, in which case we can choose

$$k = 1 + \max_{L_{x,B}} f(L_{x,B})$$

to satisfy property ii) otherwise for every k we could find a k-repeating word $y \in L_{x,B}$, thus contradicting the definition of k-decomposability for the prime play xy.

Whether $f(L_{x,B}) = \infty$ is equivalent to the existence of a computation loop of the automaton $\mathcal{A}_{x,B}$ from which a final state is reachable and which at the same time contains each letter of B. If such a loop exists, there exists one of length at most $|B|$ times the number of states of the automaton. Thus we can choose $\text{decomp}(G) = 1 + |B| \max_{\mathcal{A}_{x,B}} |\mathcal{A}_{x,B}|$.

This terminates the proof of lemma 20. ◀

12 New examples

12.1 Four players games: proof of Lemma 22

Lemma 22. *Four players games are structurally decomposable.*

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

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

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

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

12.2 Five players games: proof of Lemma 23

Lemma 23. *Let G be a distributed game with five processes. 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 $\partial_{\mathrm{last}(z)}(xz)$ 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. ◀

12.3 Majority games: proof of Lemma 24

Lemma 24. *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 $2 * |\,\mathrm{dom}(y)| > |\mathbb{P}|$ then $\mathbb{Q} = \mathbb{P}$ and every action is $\mathbb{Q}$-safe. Since $|\mathbb{P} \setminus \mathrm{dom}(b)| \leq |\,\mathrm{dom}(b)|$ then $2 * |\,\mathrm{dom}(b)| \geq |\mathbb{P}|$ hence $2(|\mathbb{P}| - |\,\mathrm{dom}(b)|) \leq |\mathbb{P}| < 2 * |\,\mathrm{dom}(y)|$. Thus $(\mathbb{P} \setminus \mathrm{dom}(b)) \prec \mathrm{dom}(y)$.

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

12.4 Merging games: proof of Lemma 26

Lemma 26. *Let G be a game, and $\mathbb{P}_0, \mathbb{P}_1 \subseteq \mathbb{P}$ two set of processes such that $\mathbb{P} = \mathbb{P}_0 \cup \mathbb{P}_1$ and 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.

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 because $\preceq_0$ and $\preceq_1$ are.

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

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

Assume that $\mathrm{dom}(y) \subseteq \mathbb{P}_0 \setminus \mathbb{P}_1$ (the case $\mathrm{dom}(y) \subseteq \mathbb{P}_1 \setminus \mathbb{P}_0$ 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:

$$\forall a \in A, \mathrm{dom}(a) \cap (\mathbb{P}_0 \setminus \mathbb{P}_1) \neq \emptyset \wedge a \mathbb{I} b \implies a \text{ is } \mathbb{Q}_0\text{-safe in } G_0 \tag{49}$$

$$\mathbb{Q}_0 \setminus \mathrm{dom}(b) \prec_0 \mathrm{dom}(y) \ . \tag{50}$$

Set $\mathbb{Q} = \mathbb{Q}_0 \cup \mathbb{P}_1$. Since $\mathrm{dom}(y) \subseteq (\mathbb{P}_0 \setminus \mathbb{P}_1)$ and $\mathbb{Q} \cap (\mathbb{P}_0 \setminus \mathbb{P}_1) = \mathbb{Q}_0$ then (50) and the definition of $\prec$ implies $\mathbb{Q} \setminus \mathrm{dom}(b) \prec \mathrm{dom}(y)$. We show that every letter $a \mathbb{I} b$ is $\mathbb{Q}$-safe for that we assume $\mathrm{dom}(a) \cap \mathbb{Q} \neq \emptyset$ and we prove that $\mathrm{dom}(a) \subseteq \mathbb{Q}$ or equivalently $\mathrm{dom}(a) \cap (\mathbb{P}_0 \setminus \mathbb{P}_1) \subseteq \mathbb{Q}_0$. If $\mathrm{dom}(a) \cap (\mathbb{P}_0 \setminus \mathbb{P}_1) = \emptyset$ there is nothing to prove. Otherwise since $\mathrm{dom}(a) \cap \mathbb{Q} \neq \emptyset$ then $\mathrm{dom}(a) \cap \mathbb{Q}_0 \neq \emptyset$. Moreover according to (49), a is $\mathbb{Q}_0$-safe in G_0 thus since $\mathrm{dom}(a) \cap \mathbb{Q}_0 \neq \emptyset$ then $\mathrm{dom}(a) \cap (\mathbb{P}_0 \setminus \mathbb{P}_1) \subseteq \mathbb{Q}_0$ which terminates to prove that every action $a \mathbb{I} b$ is $\mathbb{Q}$-safe. Thus G is structurally decomposable. ◀