# On the controller synthesis problem for distributed systems with causal memory

Hugo Gimbert

# On the controller synthesis problem for distributed systems with causal memory

Hugo Gimbert

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 [12], where both the plant and the controllers are modelled as Zielonka automata [13,1] and the controllers have causal memory is a challenging open problem [8,9]. There exists 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 provide a unified proof of these three results, and designing a class of plants, called broadcast games, with a decidable controller synthesis problem. This gives new examples of decidable architectures.

## 1 Introduction

The decidability of the distributed version of the Ramadge and Wonham control problem [12], where both the plant and the controllers are modelled as Zielonka automata [13,1] and the controller have causal memory is a challenging open problem. Very good introductions to this problem are given in [8,9].

We assume that the 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 he cannot block the *uncontrollable* actions from the environment. The choice of the local controller is based on two sources of information:

- First the controller monitors the sequence of states and actions of its 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 correct controller restricts controllable actions so that every possible execution of the plant satisfies some specification. The controller synthesis problem aims at automatically computing a correct controller from its specification.

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 a one of the key aspects to get decidable synthesis problems [4]. In early work on distributed controller synthesis, for example in the setting of [11], 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 [3] 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 [4].

We adopt a modern terminology and call the plant a *distributed game* 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 *local reachability 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 local reachability condition, whatever non-deterministic choices are performed by the environment.

There exists three classes of plants for which the existence of a winning distributed strategy has been shown decidable:

1. when the dependency graph of actions is series-parallel [4],
2. when the processes are connectedly communicating [7],
3. and when the dependency graph of processes is a tree [5,10].

A series-parallel game is a game such that the dependence graph $(A, D)$ of the alphabet $A$ is a co-graph. Series-parallel games were proved decidable in [4], for a different setup than ours: in the present paper we focus on process-based control while [4] was focusing on action-based synthesis. Actually action-based control is more general than process-based control, see [4] for a proof. 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 [4] was the first discovery of a class of asynchronous distributed system with causal memory for which controller synthesis is decidable

Connectedly communicating games have been introduced [7] under the name of *connectedly communicating processes*. Intuitively, a game is connectedly communicating if there is a bound $k$ such that if a process $p$ executes $k$ steps without hearing from process $q$, directly or indirectly, then $p$ will never hear from $q$ again. The event structure of a connectedly communicating games has a decidable MSO theory [7] which implies that controller synthesis is decidable for these games.

An acyclic game as defined in [5] is a game where processes are arranged as a tree and actions are either local or synchronize a father and his son. Even in this simple setting the synthesis problem is provably non-elementary hard.

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

The new 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 threads*. Whenever a useless thread exists, we remove it using an operation called a *shortcut* in order to get a simpler strategy. Intuitively, a shortcut is a 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 thread anymore. Strategies with no useless thread have bounded size and they can be enumerated which leads to decidability.

Performing cut-and-paste in a distributed strategy is not as easy as doing it in a centralized game. In a centralized game with only one process, strategies are trees and one can cut a subtree from a point A and paste it to any other node B in A, and the operation makes sense as long as the unique process is in the same state in A and B. But in the case of a general distributed strategy, it is not sufficient that states of the processes coincide at the source and the destination, one has also to take into account the parallelism and the various information of the different processes, so that the result of the operation is still a distributed strategy. The decidability of series-parallel games established [4] 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.

This is the reason for introducing the notion of *broadcast*. A broadcast is a part of a strategy where an information is guaranteed to spread in a pool of processes before any of these processes synchronize with a process outside the pool. When two broadcasts are similar in some sense made precise in the proof of the theorem, they can be used to perform cut-and-past: upon arrival on A, a process of the pool broadcasts to other processes of the pool that they should jump to B, and play as if the path from A to B had been already taken.

The transformation of an arbitrary winning strategy to a simpler one is done by induction on the set of actions, which relies on a notion of *process ordering* of the set of actions. This notion is useful to define the new class of broadcast game and treat examples uniformly. However to our opinion the main contribution of the paper is not the notion of process ordering and broadcast games but rather the notion of useless threads and shortcuts ad their use to simplify strategies.

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 [5]. Nevertheless we think this paper sheds new light on the difficult open problem of distributed synthesis.

Missing proofs and further examples can be found in the appendix and in [6].

## 2 Definitions and basic properties

### 2.1 Mazurkiewicz traces

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

We fix an alphabet $A$ and a symmetric and reflexive dependency relation $D \subseteq A \times A$ and the corresponding independency relation $\mathbb{I} \subseteq A \times A$ defined by:

$$\forall a, b \in A, (a \mathbb{I} b) \iff (a, b) \notin D \ .$$

For $u, v \in A^*$, we denote $\mathrm{Alph}(u)$ the set of letters of $u$ and we write:

$$u \mathbb{I} v$$

whenever $\mathrm{Alph}(u) \times \mathrm{Alph}(v) \subseteq \mathbb{I}$ . For $B \subseteq A$ we write $u \mathbb{I} B$ whenever $\forall b \in B, b \mathbb{I} u$.

A Mazurkiewicz trace on $(A, \mathbb{I})$ is an equivalence class of words for the smallest equivalence relation $\approx$ on $A^*$ such that:

$$\forall u, v \in A^*, \forall a, b \in A, ((a \mathbb{I} b) \implies (uabx \approx ubax)) \ .$$

In most of the paper, a Mazurkiewicz trace is simply called a *trace*. A word in a trace is called a *linearization* of the trace. The empty trace denoted $\epsilon$ is the singleton which contains only the empty word. All words of a trace have the same alphabet, thus the notation $\mathrm{Alph}(u)$ extends to traces. The length of a trace $u$, denoted $|u|$, is the number of letters of any linearization of $u$.

For a subset $B \subseteq A$, a trace on $B$ is a trace $u$ such that $\mathrm{Alph}(u) \subseteq B$. We denote $B^*$ the set traces on an alphabet $B \subseteq A$.

The concatenation on words naturally extends to traces, given two traces $u, v \in A^*$, the trace $uv$ is the equivalence class of any word $u'v'$ such that $u' \in u$ and $v' \in v$. Also the notion of prefix extends to traces. A trace $u \in A^*$ is a prefix of a trace $v \in A^*$, denoted

$$u \sqsubseteq v$$

if there exists $w \in A^*$ such that $uw = v$. And $u$ is a suffix of $v$ is there exists $w \in A^*$ such that $v = wu$.

### 2.2 Prime traces and views

A trace $u \in A^*$ is *prime* if all its linearization have the same last letter. If this letter is $a \in A$, i.e. if $u \in A^*a$, $u$ is said to be $a$-prime. Let $B \subseteq A$. If all linearization of $u$ ends up with a letter in $B$ then $u$ is said to be $B$-prime.

Let $B \subseteq A$ and $u \in A^*$. Then there exists a shortest prefix $\partial_B(u)$ of $u$, called the *$B$-view* and denoted

$$\partial_B(u) \ .$$

such that $u$ factorizes as $u = \partial_B(u) \cdot v$ with $v \ \mathbb{I} \ B$. If $B$ is a singleton $\{b\}$ then the $B$-view is also called the $b$-view and denoted $\partial_b(u)$.

### 2.3 Processes and automata

Zielonka automata are to traces what finite automata are to finite words.

**Definition 1.** *A Zielonka automaton $\mathcal{A}$ on the alphabet $A$ and the set of processes $\mathbb{P}$ is a tuple*

$$\mathcal{A} = (A, (Q_p)_{p\in\mathbb{P}}, (i_p)_{p\in\mathbb{P}}, (F_p)_{p\in\mathbb{P}}, (A_p)_{p\in\mathbb{P}}, \Delta),$$

*where*

- *$\mathbb{P}$ is a finite set called the set of processes,*
- *$Q_p$ is the set of states of process $p$,*
- *$i_p \in Q_p$ is the initial state of $p$,*
- *$F_p \subseteq Q_p$ is the set of final states of $p$,*
- *$A_p$ is the set of actions of process $p$,*
- *$A = \bigcup_{p\in\mathbb{P}} A_p$ and for $a \in A$, the set $\{p \in \mathbb{P} \mid a \in A_p\}$ is called the domain of $a$ and denoted* $\mathrm{dom}(a)$*,*
- *$\mathcal{T} \subseteq \bigcup_{a\in A}\{a\} \times \prod_{p\in\mathrm{dom}(a)} Q_p \times Q_p$ is the set of transitions,*

*We assume that transitions are deterministic i.e. for every $a \in A$, if $(a, (q_p, q'_p)_{p\in\mathrm{dom}(a)}) \in \Delta$ and $(a, (q_p, q''_p)_{p\in\mathrm{dom}(a)}) \in \Delta$ then $q'_p = q''_p$ for every $p \in \mathrm{dom}(a)$.*

The automaton $\mathcal{A}$ defines a dependency relation $D$ and its dual commutation relation $\mathbb{I}$ on $A$: two letters can commute if and only if they have no process in common in their domains:

$$\begin{aligned} &((a,b) \in D) \iff (\mathrm{dom}(a) \cap \mathrm{dom}(b) \neq \emptyset) \ , \\ &a \ \mathbb{I} \ b \iff \mathrm{dom}(a) \cap \mathrm{dom}(b) = \emptyset \ . \end{aligned}$$

This naturally defines a notion of Mazurkiewicz trace on $A$.

We extend the notion of views and independence to processes. Let $p \in \mathbb{P}$ then the $p$-view of a trace $u \in A^*$ is

$$\partial_p(u) = \partial_{A_p}(u) \ ,$$

and since all letters of $A_p$ are dependent from each other,

$$\forall p \in \mathbb{P}, \forall u \in A^*, \partial_p(u) \text{ is prime.} \tag{1}$$

Moreover for $p \in \mathbb{P}$ and $u \in A^*$, $p \ \mathbb{I} \ u$ is a notation for $A_p \ \mathbb{I} \ u$. We extend the notion of domain to traces:

$$\mathrm{dom}(a_1 \cdots a_n) = \bigcup_{i=1}^{n} \mathrm{dom}(a_i) \ .$$

### 2.4 Plays

A play is an asynchronous computation of the automaton defined as follows.

**Definition 2 (Plays and maximal plays).** *The set of plays of the automaton $\mathcal{A}$ denoted* plays($\mathcal{A}$) *is defined inductively, together with a mapping $Q$ :* plays($\mathcal{A}$) $\to \Pi_{p\in\mathbb{P}} Q_p$. *The set* plays($\mathcal{A}$) $\subseteq A^*$ *is the smallest set of traces on $A$ such that:*

- $\epsilon$ *is a play and* $Q(\epsilon) = (i_p)_{p\in\mathbb{P}}$,
- *if* $u \in$ plays($\mathcal{A}$), $a \in A$ *and there exists* $(a, (q_p, q'_p)_{p\in \mathrm{dom}(a)}) \in \Delta$ *such that* $\forall p \in \mathrm{dom}(a), q_p = Q_p(u)$ *then* $ua \in$ plays($\mathcal{A}$) *and for every* $p \in \mathbb{P}$,

$$Q_p(ua) = \begin{cases} Q_p(u) & \text{if } p \notin \mathrm{dom}(a), \\ q'_p & \text{otherwise.} \end{cases}$$

The definition makes sense because for every $u \in$ plays($\mathcal{A}$), whatever linearization of $u$ is chosen to compute $Q(u)$ does not change the value of $Q(u)$. This holds because $\forall u \in \mathrm{plays}(\mathcal{A}), Q_p(u) = Q_p(\partial_p(u))$, which can be easily proved inductively.

### 2.5 Strategies and games

Given an automaton $\mathcal{A}$, we would like the processes to choose actions so that every play eventually terminates in a final state of $\mathcal{A}$.

Not all actions are controllable by processes, and we assume that $A$ is partitioned in $A = A_c \sqcup A_e$ where $A_c$ is the set of controllable actions and $A_e$ the set of (uncontrollable) environment actions. Intuitively, processes cannot prevent their environment to play actions in $Ae$, while they can forbid some of the actions that are in $A_c$.

The choice of actions by processes is dynamic: at every step, a process $p$ chooses a new set of controllable actions, depending on its (local) information about the way the play is (globally) going on.

This information of a process $p$ on a play $u$ is assumed to be the $p$-view $\partial_p(u)$: intuitively two processes cannot communicate together unless they synchronize on a common action. In this case they exchange as much information about the play as they want. In particular it allows them to compute at this instant their common view of the play, i.e. their causal past. Formally, for every $a$-prime play $ua \in$ plays($\mathcal{A}$), this view is:

$$\forall p, q \in \mathrm{dom}(a), \partial_p(ua) = \partial_q(ua) = ua \ .$$

We adopt a modern terminology and call the automaton $\mathcal{A}$ together with the partition $A = Ac \sqcup Ae$ a *distributed game*, or simply a game in this paper. In this game the processes play distributed strategies, defined as follows.

**Definition 3 (Distributed strategy).** *A distributed strategy $\sigma_p$ for process $p \in \mathbb{P}$ in the game $\mathcal{A}$ is a mapping $\sigma_p : A^* \to 2^A$ such that for every $u \in A^*$,*

$$A_e \subseteq \sigma_p(u) \ ,$$
$$\sigma_p(u) = \sigma_p(\partial_p(u)) \ .$$

*A distributed strategy in $\mathcal{A}$ is a tuple $\sigma = (\sigma_p)_{p\in\mathbb{P}}$ where each $\sigma_p$ is a strategy of process $p$. A play $u = a_1 \cdots a_n$ is a $\sigma$-play if $u \in \mathrm{plays}(\mathcal{A})$ and for every $i \in 1..n$ and every $p \in \mathrm{dom}(a_i)$, $a_i \in \sigma_p(a_1 \cdots a_i)$. A $\sigma$-play is maximal if it is not the strict prefix of another $\sigma$-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 occur. In particular it may happen that a process enters a final state with no outgoing transition, where no further action is possible.

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

**Definition 4 (Winning strategy).** *A strategy $\sigma$ is winning if the set of plays consistent with $\sigma$ is finite and for every maximal $\sigma$-play $u$,*

$$Q(u) \in \Pi_{p\in\mathbb{P}} F_p \ .$$

The *distributed synthesis problem* asks, given a game $G = (\mathcal{A}, Ac, Ae)$, whether the game is winning, in the sense where there is a winning strategy in $G$. If yes such a strategy should be computed.

## 3 Taking shortcuts

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 play $xy \in [\sigma]$ consistent with a strategy $\sigma$ and modify the strategy so that as soon as a process sees the play $x$ in its causal past in their causal past, they assume that not only $x$ but also $xy$ has actually occurred.

From a more formal point of view, a shortcut is a kind of *cut-and-paste* in the strategy tree: we glue on node $x$ the subtree rooted at node $xy$. This guarantees the new tree to be smaller than the previous one, thus if the new tree is a winning strategy, its duration is guaranteed to be smaller than the duration of the original strategy.

However the choice of $x$ and $y$ should be carefully performed so that the new tree is well-defined and is still a winning strategy. We provide a sufficient condition for that: $(x, y)$ should be a *useless thread*.

### 3.1 Duration and shortcuts

Winning strategies with shorter durations are better.

**Definition 5 (Duration of a strategy).** *The duration* $\mathrm{dur}(\sigma)$ *of a strategy* $\sigma$ *is an integer in* $\mathbb{N} \cup \{\infty\}$ *defined as follows. If the set of* $\sigma$*-plays is infinite then* $\mathrm{dur}(\sigma) = \infty$*. Otherwise*

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

Taking a shortcut is a convenient way to make a strategy shorter.

**Definition 6 (Shortcut).** *Let* $x, y \in A^*$ *such that* $xy$ *is a* $\sigma$*-play. Let* $\phi : A^* \to A^*$ *be the mapping:*

$$\phi_{x,y}(u) = \begin{cases} u & \text{if } x \not\sqsubseteq u \\ xyv & \text{if } u = xv \ , \end{cases} \tag{2}$$

*Then the* $(x, y, \sigma)$*-shortcut is the mapping* $\sigma_{x,y} : A^* \to A^*$ *defined by:*

$$\sigma_{x,y} = \sigma \circ \phi_{x,y} \ .$$

The mapping $\phi_{x,y}$ is well-defined: there exists at most one $v$ such that $u = xv$.

There is a priori no reason in general for $\sigma_{x,y}$ to be a distributed strategy. For that we need extra conditions on the pair $(x, y)$ and we introduce the notion of broadcasts and useless thread.

### 3.2 Broadcasts and useless threads

Intuitively, a *broadcast* is a prime play $u$ in a strategy such that the maximal action of the play and the associated information about the play is broadcasted in priority to a pool of processes $\mathbb{Q} \subseteq \mathbb{P}$, before any of these processes synchronize with a process outside the pool. During a broadcast, as long as a process of the pool $\mathbb{Q}$ plays in parallel of $u$ then he synchronizes only with processes in $\mathbb{Q}$.

**Definition 7 (broadcasts).** *Let* $\mathbb{Q} \subseteq \mathbb{P}$ *a subset of processes. We say that a prime play* $u$ *is a* $\mathbb{Q}$-broadcast *if for every play* $uv$ *and every action* $a$ *maximal in* $uv$*,*

$$(\mathrm{dom}(a) \cap \mathbb{Q} \neq \emptyset) \text{ and } (\mathrm{dom}(a) \cap \mathbb{P} \setminus \mathbb{Q} \neq \emptyset) \implies (u \sqsubseteq \partial_a(uv)) \ . \tag{3}$$

In other words, a broadcast prevents a process in $\mathbb{Q}$ to synchronize with a process out of $\mathbb{Q}$ in parallel of $u$.

For example, in an acyclic game, where processes are arranged in a tree organization, every time a process $p$ plays then it is a broadcast to its subtree: the only way for a process in the subtree to synchronize with other processes is by playing with $p$. Another example is given by series-parallel game: if the dependency alphabet $(A, D)$ is the product of two alphabets $(A_0, D_0)$ and $(A_1, D_1)$

with $D = D_0 \cup D_1 \cup A_0 \times A_1 \cup A_1 \times A_0$, then every prime play whose maximal action is $a_0 \in A_0$ is a broadcast to $\mathrm{dom}(A_0)$. The reason is that $a_0$ conflicts with every action in $a_1$.

The definition of broadcasts can be equivalently reformulated as follows.

**Proposition 1.** *A prime play $u$ is a $\mathbb{Q}$-broadcast if and only if for every $uv$ such that $v$ is prime, ($uv$ is prime) or ($v \mathbb{I} \mathbb{Q}$) or ($\mathrm{dom}(v) \subseteq \mathbb{Q}$).*

We are interested in broadcasts occuring in *threads*.

**Definition 8 (Threads).** *Let $\mathbb{Q} \subseteq \mathbb{P}$ a subset of processes (resp. $B \subseteq A$ a subset of actions). A* $\mathbb{Q}$-thread *(resp a $B$-thread) is a pair $(u, v) \in A^* \times A^*$ such that $uv$ is a play and $\mathrm{dom}(v) \subseteq \mathbb{Q}$ (resp. $v \in B^*$).*

Some threads are called *useless threads*, they can be deleted to reduce the delay of the strategy, for the same result.

**Definition 9 (Useless thread).** *Let $\sigma$ be a strategy. A* useless thread in $\sigma$ *is a $\mathbb{Q}$-thread $(x, y)$ such that there exists an action $b$ with the following properties:*

$$x \text{ and } xy \text{ are } b\text{-prime}, \tag{4}$$

$$x \text{ and } xy \text{ are } \mathbb{Q}\text{-broadcasts} \tag{5}$$

$$\text{every process } p \in \mathbb{P} \text{ has the same state in } x \text{ and } xy, \tag{6}$$

$$\forall v \in A^*, (\mathrm{dom}(v) \subseteq \mathbb{Q} \wedge v \mathbb{I} b) \implies (\sigma(xv) = \sigma(xyv)) \quad . \tag{7}$$

Taking a shortcut of a useless thread in a distributed strategy makes sense because the result is still a distributed strategy.

**Lemma 1.** *Let $(x, y)$ a useless thread in a distributed strategy $\sigma$. Then the $(x, y, \sigma)$-shortcut $\sigma_{x,y}$ is a distributed strategy.*

Taking shortcuts of useless threads is really useful for making winning strategies smaller: it transforms a *winning* distributed strategy into another, *shorter*, *winning* distributed strategy.

**Lemma 2.** *Let $(x, y)$ a useless thread in a winning distributed strategy $\sigma$. Then the $(x, y, \sigma)$-shortcut $\sigma_{x,y}$ is a winning distributed strategy as well, and*

$$\mathrm{dur}(\sigma_{x,y}) < \mathrm{dur}(\sigma) \quad . \tag{8}$$

*Moreover for every $v \in A^*$,*

$$xv \text{ is a } \sigma_{x,y}\text{-play} \iff xyv \text{ is a } \sigma\text{-play}. \tag{9}$$

In the next section we introduce a class of games, called broadcast games, where broadcasts occur quite regularly. This is the key to obtain decidability of the synthesis problem: winning strategy with long durations can be simplified into simpler one by taking shortcuts. As a consequence we obtain a computable upper bound on the duration of the longest winning strategy.

Another way to illustrate the interest of Lemma 2 is to instantiate this lemma in the case where $(x, y)$ is a $\mathbb{P}$-thread:

**Corollary 1.** *Let $\sigma$ be a strategy and $xy$ be a $\sigma$-play such that both $x$ and $y$ are $b$-prime for some letter $b \in A$. Assume that:*

- *every process $p \in \mathbb{P}$ has the same state in $x$ and $xy$,*
- $\forall v \in A^*, (v \mathbb{I} b) \implies (\sigma(xv) = \sigma(xyv))$ .

*Then the mapping $\tau$ defined by:*

$$\tau(u) = \begin{cases} \sigma(u) & \text{if } x \not\sqsubseteq u \\ \sigma(xyv) & \text{if } u = xv \end{cases} .$$

*is a distributed strategy. Moreover* $\mathrm{dur}(\tau) < \mathrm{dur}(\sigma)$ *and if $\sigma$ is winning then $\tau$ is winning as well.*

## 4 Broadcast games

We do not know whether the distributed synthesis problem is decidable in the general case, but we know it is decidable when the game is a broadcast game.

The notion of broadcast games is defined so that decidability results of [4,7,5] can be retrieved quite easily, this is done in the next section. The definition of a broadcast game is rather ad-hoc and technical. We hope further research will lead to a more general and cleaner definition.

### 4.1 Definition

The proof of decidability of broadcast games is performed inductively, and relies on the notion of an process ordering of the dependency alphabet $(A, D)$.

**Definition 10 (Process ordering).** *An process ordering of $\mathbb{P}$ is a partial order* [1] *$\preceq$ on $P$ such that for every prime trace $u$,* $\mathrm{dom}(u)$ *has a $\preceq$-maximum.*

In a distributed game, there may exist several process orderings, for example any total order on $\mathbb{P}$ is a process ordering of $\mathbb{P}$.

A broadcast game is a game where, periodically, processes create broadcasts. The pool of processes where the broadcast occurs is computed by downward-closure of the set of actions, with respect to the process ordering $\preceq$.

**Definition 11 (Process closures).** *Let $\preceq$ an process ordering of $\mathbb{P}$ and $\mathbb{Q} \subseteq \mathbb{P}$. The process-closure of $\mathbb{Q}$ is:*

$$\mathbb{Q}_{\preceq} = \{p \in \mathbb{P} \mid p \preceq q \text{ for some } q \in \mathbb{Q}\} .$$

We are especially interested in broadcasts consistent with process ordering.

**Definition 12 (Well-ordered broadcast).** *A $\mathbb{Q}$-broadcast $u$ is* well-ordered *if $\mathbb{Q} = \mathbb{Q}_{\preceq}$ and the maximal process of* $\mathrm{dom}(u)$ *plays in the last action of $u$.*

[1] i.e. a transitive, reflexive and anti-symmetric binary relation.

For every trace $u \in A^*$ and process $p \in \mathbb{P}$, $|u|_p$ denotes the number of actions of process $p$ in $u$, i.e. the number of letters in $u$ whose domain contains $p$.

**Definition 13 (Broadcast games).** *Let $N \in \mathbb{N}$. A game $G$ is a $(N, \preceq)$-*broadcast game *if, for every prime play $uv \in A^*$, whenever*

$$\forall q \in \mathrm{dom}(v), |v|_q \geq N$$

*then there exists a prefix $v' \sqsubseteq v$ such that $uv'$ is a well-ordered $\mathrm{dom}(v)_{\preceq}$-broadcast.*

In case $G$ is an $(N, \preceq)$-broadcast game for some $N$ and $\preceq$ we also say that $G$ is an $N$-broadcast game or simply a broadcast game.

The property of being a broadcast game is decidable.

**Proposition 2.** *It is decidable whether a game $G$ is a broadcast game. In case it is then $G$ is an $N$-broadcast-game for some $N \leq \Pi_{p \in \mathbb{P}} |Q_p|$.*

Actually a weaker interesting class of games are games where processes are restricted to use broadcast strategies, without any condition on the game itself.

### 4.2 Decidability

**Theorem 1.** *Whether a broadcast game is winning or not is decidable.*

The algorithm consists in enumerating all possible strategies whose plays have length less than some computable bound, and check whether any of these strategies is winning. This bound is defined by equation (10) below. The bound is quite large, which is not surprising since the problem is non-elementary [5]. We provide some examples and applications in the next section.

The proof is easy to sketch but harder to implement because distributed systems are not so easy to handle. For every subset of processes $\mathbb{Q} \subseteq \mathbb{P}$, we compute inductively a bound $K_{\mathbb{Q}}$ such that any winning strategy which has a $\mathbb{Q}$-thread of duration more than $K_{\mathbb{Q}}$ can be simplified in a shorter winning strategy, by taking a shortcut associated to a useless thread.

With every $B \subseteq A$ we associate a constant $K_B \in \mathbb{N}$ as follows. According to Ramsey theorem, for every $m, n \in \mathbb{N}$, there exists a constant $R(m, n)$ such that every undirected complete graph with at least $R(m, n)$ vertices whose edges are labelled with $m$ different colors contains a monochromatic clique of size $n$. Then we define inductively $K_\emptyset = 0$ and

$$K_{\mathbb{Q}} = |Q| \cdot R\left(2^{|\mathbb{Q}|}, N 2^{|A|} |A| |Q|^{|\mathbb{P}|} 2^{|A|^{K'_{\mathbb{Q}}}}\right) \quad . \tag{10}$$

Next lemma states that in a $(N, \preceq)$-broadcast game, very long strategies have useless threads.

**Lemma 3.** *Let $\sigma$ be a distributed strategy of a $(N, \preceq)$-broadcast game. Assume that for some $\mathbb{Q} \subseteq \mathbb{P}$, $\sigma$ has a $\mathbb{Q}$-thread of length more than $K_{\mathbb{Q}}$. Then there is a useless thread in $\sigma$.*

*Proof.* Without losing generality, we can choose $\mathbb{Q}_{\preceq}$ minimal for the inclusion so that every $\mathbb{Q}'_{\preceq}$-thread in $\sigma$ has length less than $K_{\mathbb{Q}'_{\preceq}}$ whenever $\mathbb{Q}'_{\preceq} \subsetneq \mathbb{Q}_{\preceq}$.

By hypothesis there exists $\mathbb{Q} \subseteq A$ and a $\sigma$-play $uv$ such that $\mathrm{dom}(v) \subseteq \mathbb{Q}$ and $\mid v \mid \geq K_{\mathbb{Q}}$. Without loss of generality we can assume $\mathbb{Q}_{\preceq} = \mathbb{Q}$. And w.l.o.g., since there are at most $|Q|$ maximal events in a trace, we can assume that $uv$ is prime and $\mid v \mid \geq \frac{K_{\mathbb{Q}}}{|Q|}$.

We consider the complete graph $G_c$ with vertices $1, \ldots, |v|$ and the label of the edge $\{i < j\}$ is defined as follows. We denote $v[i,j]$ the subword of $v$ between positions $i$ and $j$ (both included) and $A[i,j] = \mathrm{Alph}(v[i,j]) \subseteq \mathrm{Alph}(v)$ and $P[i,j] = \mathrm{dom}(v[i,j])_{\preceq} \subseteq \mathbb{Q}_{\preceq}$. Then for $1 \leq i_0 \leq i_1 \leq i_2 \leq |v|$ we set

$$E[i_0, i_1, i_2] = (A[i_0, i_2], a_{i_1}, Q_{i_1}, L[i_1]) \text{ with}$$

- $a_{i_1}$ the letter at position $i_1$ in $v$,
- $Q_{i_1}$ the states of the processes in $uv[1, i_1]$.
- $L[i_1] = \{w \in A^* \mid (\mathrm{dom}(w) \subseteq \mathbb{Q}_{\preceq}) \wedge (w \mathbb{I} \max_{\preceq} \mathrm{dom}(v))\}$.

Let $N_L$ the number of possible values of $E[i_0, i_1, i_2]$ and $H = N * N_L$. Notice that every trace in $L[i_1]$ is a $\mathbb{Q}'$-thread with $\mathbb{Q}' = (\mathbb{Q} \setminus \max_{\preceq} \mathrm{dom}(v))$. Since $\mathbb{Q}'_{\preceq} \subsetneq \mathbb{Q}_{\preceq}$ then we can apply the inductive hypothesis: every trace in $L[i_1]$ has length at most $|A|^{K'_{\mathbb{Q}}}$. Thus

$$H = N * N_L \leq N * 2^{|A|} |A| |Q|^{|\mathbb{P}|} 2^{|A|^{K'_{\mathbb{Q}}}} \quad .$$

By definition of $K_{\mathbb{Q}}$, $|v| \geq R(2^{\mathbb{Q}}, H)$. Hence by definition of Ramsey numbers, $G_c$ contains a monochromatic clique $i_1 < i_2 < \ldots < i_H$. Let $v = v_0 v_1 v_2 \ldots v_H v_{H+1}$ the corresponding factorization of $v$, such that $\mathrm{dom}(v_1) = \ldots = \mathrm{dom}(v_H)$.

For every $1 \leq i \leq H$, every process in $\mathrm{dom}(v_1)$ has played at least $N$ times in $v'_i = v_{i*N+1} \ldots v_{i*N+N}$. Thus, since the game is a broadcast game, for every $1 \leq i \leq N_L$, there exists a prefix $v''_i \preceq v'_i$ such that

$$uv_0 \ldots v_{i*N} v''_i \text{ is a well-ordered } \mathrm{dom}(v_1)_{\preceq}\text{-broadcast.}$$

By definition of $N_L$ we can find two indices $1 \leq i < j \leq N_L$ such that $E(i, |v''_i|, i + |v_i|) = E(j, |v''_j|, j + |v_j|)$ . Let $B = \mathrm{Alph}(v'_i) = \mathrm{Alph}(v'_j)$ and $\mathbb{Q}' = \mathrm{dom}(v_1) = \mathrm{dom}(v'_i)_{\preceq} = \mathrm{dom}(v'_j)_{\preceq}$ and $x = v_0 \ldots v_{i-1} v'_i$ and $y = v_0 \ldots v_{j-1} v'_j$. Then both $ux$ and $uy$ are $\mathbb{Q}'$-broadcasts, end up with the same letter, processes have the same state and local strategies coincide ($L[i] = L[j]$). Thus all conditions are met and $\sigma$ has a useless thread. □

*Proof (of Theorem 1).* Let $\sigma$ be a winning strategy of minimal duration. By minimality, according to Lemma 2, strategy $\sigma$ does not contain any useless thread. By Lemma 3, every play of $\sigma$ has length less than $K_{\mathbb{P}}$. There is a finite number of distributed strategies with this property, and for each such strategy $\sigma$, there is a simple algorithm that checks whether $\sigma$ is winning or not: look nondeterministically for a losing play consistent with $\sigma$. Thus the existence of a winning strategy is decidable. □

# 5 Examples of broadcast games

In this section we provide several examples of broadcast games. The first two are connectedly communicating games and series-parallel games whose decidability was already known. The third example are acyclic games whose decidability was already known in the case where all actions are local or binary. The fourth example is the class of triangulated games and the special case of three-player games. We terminate with a discussion about dynamic broadcast games.

## 5.1 Connectedly communicating games are broadcast games

Connectedly communicating games have been introduced [7] under the name of *connectedly communicating processes*, and the authors did establish the decidability of the MSO theory of the corresponding event structure, which implies that controller synthesis is decidable.

A game is *connectedly communicating* if there exists some bound $k \in \mathbb{N}$ such that whenever a process $q$ never plays while process $p$ plays $k$ times, then $p$ and $q$ will stay forever in separate threads. Formally, a game is $k$-communicating for some $k \in \mathbb{N}$ if for every processes $p, q \in \mathbb{P}$ and play $uvw$ in $G$,

$$((|v|_p \geq k) \wedge (|v|_q = 0) \wedge w \text{ prime}) \implies (|w|_p = 0 \vee |w|_q = 0) \ .$$

It is quite clear that every $k$-communicating game is a $k$-broadcast game. The process ordering is an arbitrary total order $\preceq$ on $\mathbb{P}$. Then, for every play $uv$, the hypothesis $\forall q \in \mathrm{dom}(v), |v|_q \geq k$ in the definition of broadcast games implies that no process $q \in \mathrm{dom}(v)$ will ever synchronize with a process $p \notin \mathrm{dom}(v)$ after $uv$ thus $uv$ itself is a (well-ordered) $\mathrm{dom}(v)_{\preceq}$-broadcast.

Thus every connectedly communicating is a $k$-broadcast game.

## 5.2 Series-parallel games

A series-parallel game is a game where the dependence graph $(A, D)$ of the alphabet $A$ is a co-graph i.e. it belongs to the smallest class of graphs containing singletons and closed under parallel product and complementation. Series-parallel games were proved decidable in [4], for a slightly more general setup than ours called *action-based synthesis*, which is more general than process-based control [8].

Although the series-parallel case does not fit directly in the class of broadcast games, the same techniques can be used to prove decidability in this case as well.

If $A$ is a singleton then clearly the synthesis problem is decidable in this case. In this case positional strategies are sufficient to win, hence the length of play can be bounded by $|Q|$. Otherwise either $(A, D)$ is the parallel product of two cographs or $(A, D)$ is the complement of the parallel product of two cographs. Thus there exists two cographs $G_0 = (A_0, D_0)$ and $G_1 = (A_1, D_1)$ such that $A$ is the disjoint union of $A_0$ and $A_1$ and

$$\begin{aligned} &\text{either } D = D_0 \cup D_1 \\ &\text{or } D = D_0 \cup D_1 \cup A_0 \times A_1 \ . \end{aligned}$$

In both cases, any trace whose final action is in $A_0$ is a $\mathrm{dom}(A_0)$-broadcast and the same holds for $A_1$. Thus, if a strategy $\sigma$ allows a play $xy \in A^*$ such that $x$ and $y$ have the same last letter in $A_0$, processes have the same states in $x$ and $y$ and the restriction of the strategy to $A_0$ after $x$ and $y$ is the same, then $xy$ is a useless thread. By induction, this proves that if a strategy has no useless thread then any play has length less than:

$$K_A = |A_0|^{K_{A_0}}|A_0||Q|2^{|A|^{K_{A_0}}} + |A_1|^{K_{A_1}}|A_1||Q|2^{|A|^{K_{A_1}}} \quad .$$

As a consequence winning strategies can be looked for in a finite set, which gives decidability. A more general definition of broadcast games to encompass series-parallel case is possible but at the cost of unconvenient technical notations.

### 5.3 DAG games are 1-broadcast games

An acyclic game as defined in [5] is a game where processes are arranged as a tree and actions are either local or synchronize a father and his son. Formally, the processes are arranged as a tree $T_{\mathbb{P}} = (\mathbb{P}, E_{\mathbb{P}})$, and each action is either a local action whose domain is a singleton or a binary synchronizing action such that $\mathrm{dom}(a) = \{p, q\}$ and $(p, q) \in E_{\mathbb{P}}$ i.e. $q$ is the father of $p$ in the process tree.

We extend the definition of [5] to the case of non-binary actions and we relax the assumption about the tree structure into the existence of a partial order [2] $\leq_{\mathbb{P}}$ on $\mathbb{P}$ such that for every action $a \in A$ and processes $p_0, p_1, p_2 \in \mathbb{P}$,

$$(p_0 \in \mathrm{dom}(a) \wedge p_1 \in \mathrm{dom}(a) \wedge p_0 \leq_{\mathbb{P}} p_2) \implies (p_1 \leq_{\mathbb{P}} p_2 \vee p_2 \in \mathrm{dom}(a)) \ . \quad (11)$$

This condition holds in particular in the case of trees and arbitrary actions with connected domains: if $p_0 \leq_{\mathbb{P}} p_2$ then either $p_2 \in \mathrm{dom}(a)$ or $p_2$ is an ancestor of all processes in $\mathrm{dom}(a)$ thus in particular $p_2 \geq_{\mathbb{P}} p_1$. Also this shows that four player games, ($\mathbb{P} = \{1, 2, 3, 4\}$) where all actions are allowed except those with supports $\{1, 4\}$, $\{1, 3\}$ and $\{2, 4\}$ are decidable.

There is a natural process ordering $\preceq$ of $\mathbb{P}$ associated with an acyclic game: the order coincides with the edge of the DAG.

Then every DAG game is a $(1, \preceq)$-broadcast game, whenever properties (11) are satisfied. Let $uv$ a primary play. We shall find a prefix $v' \sqsubseteq v$ such that $v'$ is a $\mathrm{dom}(v)_{\preceq}$-broadcast. Let $p$ a maximal process in $\mathrm{dom}(v)$ and $v'$ the shortest prime prefix of $v$ whose last action is an action of $p$. Then we show that $v'$ is a $\mathrm{dom}(v)_{\preceq}$-broadcast.

Let $uv'w$ a play and $c$ a maximal action of $uv'w$ such that $\mathrm{dom}(c) \cap \mathrm{dom}(v)_{\preceq} \neq \emptyset$ and $\mathrm{dom}(c) \not\subseteq \mathrm{dom}(v)_{\preceq}$. Let $p_0 \in \mathrm{dom}(c) \cap \mathrm{dom}(v)_{\preceq}$ and $p_1 \in \mathrm{dom}(c) \setminus \mathrm{dom}(v)_{\preceq}$. Then $p$ is the $\leq_{\mathbb{P}}$-maximum of $\mathrm{dom}(v)_{\preceq}$ thus $p_0 \leq_{\mathbb{P}} p$ and $\neg(p_1 \leq_{\mathbb{P}} p)$. Thus according to (11) we get $p \in \mathrm{dom}(c)$. As a consequence $uv' = \partial_c(uv') \sqsubseteq uv'w$.

Thus every DAG game with property (11) is decidable.

[2] i.e. a reflexive, anti-symmetric and transitive binary relation.

### 5.4 Triangulated games and three player games

Any 3-player game $G$ with processes $\{1, 2, 3\}$ is a 1-broadcast game, with the order $1 \preceq 2 \preceq 3$.

We show that any game $G$ with alphabet $A$ is a $(1, \preceq)$-broadcast game. Let $uv$ be a prime trace. We shall find a prefix $v' \sqsubseteq v$ such that $uv'$ is a well-ordered $\mathrm{dom}(v)_{\preceq}$ broadcast. The case where $\mathrm{dom}(v)$ is a singleton is obvious. In the remaining case we select any primary prefix $v'$ of $v$ whose last action $a$ is binary or ternary. This is a $\mathrm{dom}(a)$-broadcast because all players in $\mathrm{dom}(a)$ can immediately observe $uv'$. Consequently, all 3-player games are 1-broadcast games and are decidable.

A triangulated game is a game where processes are arranged as an undirected graph $G_{\mathbb{P}} = (\mathbb{P}, E_{\mathbb{P}})$ such that all simple cycles in the graph have length 3, and moreover we assume that

$$\forall a \in A, \mathrm{dom}(a) \text{ is connected in } G_{\mathbb{P}}. \tag{12}$$

This definition is inspired by [2]. If we shrink all triangles of a triangulated game then we get a tree, thus we can fix a root of this tree and attribute a depth to these triangles. Then we order processes with respect to the average depth of the triangles it belongs to, and there is a natural notion of descendants, that we do not detail here.

Like in an acyclic game, every action of a process $p$ is broadcasted to its descendants. The reason is every synchronisation of a descendant with a non-descendant process should include $p$, according to hypothesis (12) and by absence of simple cycles of length 4. Consequently, all triangulated games with property (12) are 1-broadcast games and are decidable.

### 5.5 Dynamic broadcast games and strategies

The condition for being a broadcast game is *dynamic*, by opposition to *static* restrictions on the architecture of processes or the dependency alphabet: a broadcast game may well behave for some time as a ring or another architecture whose decidability is unknown, as long as it performs a broadcast from time to time. Actually probably the requirement of the existence of broadcasts could be put on strategies rather than on the games themselves.

## Conclusion

We have presented a theorem and proof techniques that unifies several known decidability results for distributed games, and presented new examples of distributed games for which the existence of a winning strategy is decidable. The decidability of distributed synthesis in the general case is still open to our knowledge, even in the simple case of ring games where $G_{\mathbb{P}}$ is a simple cycle of length 5, or in the case where the automaton is synchronizing.

## References


1. Diekert, V., Rozenberg, G.: The Book of Traces. World Scientific (1995), https://books.google.co.uk/books?id=vNFLOE2pjuAC
2. Diekert, V., Muscholl, A.: A note on Métivier's construction of asynchronous automata for triangulated graphs. Fundamenta Informaticae 25(3), 241–246 (1996)
3. Finkbeiner, B., Schewe, S.: Uniform distributed synthesis. In: Logic in Computer Science, 2005. LICS 2005. Proceedings. 20th Annual IEEE Symposium on. pp. 321–330. IEEE (2005)
4. Gastin, P., Lerman, B., Zeitoun, M.: 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. pp. 275–286 (2004), http://dx.doi.org/10.1007/978-3-540-30538-5_23
5. Genest, B., Gimbert, H., Muscholl, A., Walukiewicz, I.: Asynchronous games over tree architectures. In: Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II. pp. 275–286 (2013), http://dx.doi.org/10.1007/978-3-642-39212-2_26
6. Gimbert, H.: A class of zielonka automata with a decidable controller synthesis problem. CoRR abs/1601.05176 (2016), http://arxiv.org/abs/1601.05176
7. Madhusudan, P., Thiagarajan, P.S., Yang, S.: 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. pp. 201–212 (2005), http://dx.doi.org/10.1007/11590156_16
8. Muscholl, A., Walukiewicz, I., Zeitoun, M.: A look at the control of asynchronous automata (2008), http://www.labri.fr/perso/anca/Publications/mwz08thiagu.pdf
9. Muscholl, A.: Automated synthesis of distributed controllers. In: Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II. pp. 11–27 (2015), http://dx.doi.org/10.1007/978-3-662-47666-6_2
10. Muscholl, A., Walukiewicz, I.: 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. pp. 639–651 (2014), http://dx.doi.org/10.4230/LIPIcs.FSTTCS.2014.639
11. Pnueli, A., Rosner, R.: Distributed reactive systems are hard to synthesize. In: Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on. pp. 746–757. IEEE (1990)
12. Ramadge, P.J., Wonham, W.M.: The control of discrete event systems. Proceedings of the IEEE 77(1), 81–98 (1989)
13. Zielonka, W.: Notes on finite asynchronous automata. ITA 21(2), 99–135 (1987)

# Appendix

## 6 Elementary properties of traces

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

$$\forall u, v \in A^*, ((u \sqsubseteq v) \wedge (v \sqsubseteq u) \implies u = v) \ , \tag{13}$$
$$\forall u, v, w \in A^*, (uv = uw) \implies (v = w) \ , \tag{14}$$
$$\forall u, v, w \in A^*, (uv \sqsubseteq uw) \implies (v \sqsubseteq w) \ . \tag{15}$$

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

**Lemma 4.** *For every trace $u, v, x \in A^*$ and $a \in A$ and $B \subseteq A$,*

$$(u \mathbb{I} B) \iff (\partial_B(u) = \epsilon) \tag{16}$$
$$(x \sqsubseteq uv) \implies \exists x_0 \sqsubseteq u, \exists x_1 \sqsubseteq v, x = x_0 x_1 \tag{17}$$
$$(x \sqsubseteq uv) \implies \exists x_0, x_1, x_2, x_3 \in A^*, \tag{18}$$
$$(x = x_0 x_1) \wedge (u = x_0 x_2) \wedge (v = x_1 x_3) \wedge (x_2 \mathbb{I} x_1)$$
$$uv \text{ is } B\text{-prime} \implies v \text{ is } B\text{-prime} \tag{19}$$
$$u \text{ and } v \text{ are } B\text{-prime} \implies uv \text{ is } B\text{-prime} \tag{20}$$
$$\text{If } ua \text{ is prime}, (av \text{ is } B\text{-prime} \iff uav \text{ is } B\text{-prime}) \tag{21}$$
$$(u \text{ is } B\text{-prime} \wedge \neg(a \mathbb{I} u)) \implies (au \text{ is } B\text{-prime}) \tag{22}$$
$$(u \sqsubseteq \partial_B(uv)) \iff (\partial_B(uv) = u\,\partial_B(v)) \tag{23}$$
$$(uw \sqsubseteq \partial_B(uv)) \implies (w \sqsubseteq \partial_B(v)) \tag{24}$$
$$\partial_B(\partial_B(u)) = \partial_B(u) \tag{25}$$
$$\partial_B(uv) = \partial_B(u\,\partial_B(v)) \tag{26}$$
$$\text{If } ua \text{ is prime},$$
$$uav \sqsubseteq \partial_B(uavw) \iff av \sqsubseteq \partial_B(avw) \tag{27}$$

*Proof.* The equivalence (16) is immediate from the definition of $\partial_B$.

Equation (17) is a corollary of (18) which is well-known, see [1] for example. It can be proved by induction on $|x|$.

We prove (19). If the last letter of a word $v' \in v$ is not in $B$, then the same holds for every $u'v'$ where $u' \in u$ thus $uv$ is not $B$-prime since $u'v' \in uv$.

We prove (20). Assume both $u$ and $v$ are $B$-prime. Every linearization of $uv$ is a shuffle of a linearization of $u$ and a linearization of $v$ thus it terminates with a letter in $B$. Hence $uv$ is $B$-prime.

We prove (21). Assume $ua$ prime. The converse implication follows from (19). Assume $av$ is $B$-prime. We prove that $uav$ is $B$-prime by induction on $|u|$. If $|u| = 0$ then $u = \epsilon$ and $uav = av$ is $B$-prime by hypothesis. By induction let $n \in \mathbb{N}$ and assume $u'av$ is $B$-prime for all $u'$ such that $|u'| \leq n$. Let $u$ such that $|u| = n + 1$, we prove that $uav$ is $B$-prime. Since $|u| = n + 1$, there exists

$b \in A$ and $u' \in A^*$ such that $u = bu'$ and $|u'| = n$. Using (19) and the induction hypothesis so on one hand we know that $u'av$ is $B$-prime. By definition of a trace, for any trace $w$,

$$bw = \{xbz \mid x, z \text{ words on } A \ , xz \in w, b \mathbb{I} x\} \ . \tag{28}$$

Let $y$ a linearization of $uav = bu'av$, we prove that the last letter of $y$ is in $B$. According to (28), $y$ factorizes as $y = xbz$ with $xz \in u'av$ and $x \mathbb{I} b$. Since $xz \in u'av$ and $u'av$ is $B$-prime, if $z$ is not empty then it ends with a letter in $B$ 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 $A(u'a) \subseteq A(y)$ and $A(v) \subseteq A(y)$. Since $A(y) = A(x) \cup \{b\}$ 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 $B$-prime, $a = b \in B$. Finally $b \in B$ and since $y = xb$ the last letter of $y$ is in $B$, which terminates the proof of the inductive step, and the proof of (21).

We prove (22) by contradiction. Assume $au$ is not $B$-prime then there exists a word $v'$ and a letter $c \notin B$ such that $v'c \in au$. Let $u' \in u$ then $au' \in au$ and $v'c \approx au'$ thus $A(v') \cup \{c\} = A(u') \cup \{a\}$. If $a \notin A(v')$ then $a = c$ and $v'c \approx au'$ implies $a \mathbb{I} u$ which is false by hypothesis. Thus $a \in A(v')$. Let $w'$ be the longest prefix of $v'$ which does not contain $a$ and $x'$ the suffix of $v'$ such that $v' = w'ax'c$. Then $au' \approx w'ax'c$ and $a \notin A(w')$ thus $w' \mathbb{I} a$ . Then $w'ax'c \approx aw'x'c$ thus $aw'x'c \approx au'$ hence $w'x'c \approx u'$ and $w'x'c \in u$. Since $c \notin B$, this contradicts the hypothesis $u$ is $B$-prime.

We prove (23). The converse implication in (23) is obvious so it is enough to prove the direct implication. Assume $u \sqsubseteq \partial_B(uv)$. According to (13) it is enough to prove both $\partial_B(uv) \sqsubseteq u\,\partial_B(v)$ and $u\,\partial_B(v) \sqsubseteq \partial_B(uv)$. We start with $u\,\partial_B(v) \sqsubseteq \partial_B(uv)$.. Since $u \sqsubseteq \partial_B(uv)$, then $\partial_B(uv) = uw$ for some $w \in A^*$ and $uv = uww'$ for some $w' \mathbb{I} B$. Then $v = ww'$ according to (14) and since $w' \mathbb{I} B$, then $\partial_B(v) \sqsubseteq w$, thus $u\,\partial_B(v) \sqsubseteq uw = \partial_B(uv)$ and we got the first prefix relation. Now we prove the converse prefix relation. Since $\partial_B(v) \sqsubseteq w$ then by definition of $\partial_B$ there exists $w'' \in A^*$ such that $w = \partial_B(v)w''$ and $w'' \mathbb{I} B$. Then $uv = u\,\partial_B(v)w''w'$ and $w''w' \mathbb{I} B$ thus by definition of $\partial_B$, $\partial_B(uv) \sqsubseteq u\,\partial_B(v)$. By definition of $w$ this implies $uw \sqsubseteq u\,\partial_B(v)$ thus according to (15) $w \sqsubseteq \partial_B(v)$. Finally $w = \partial_B(v)$ and $u\,\partial_B(v) = uw = u\,\partial_B(v)$ which terminates the proof of (23).

Equation (24), is a direct corollary of (23). Let $v', w'$ such that $\partial_B(uv) = uwv'$ and $uv = \partial_B(uv)w'$. Then according to (23), $uwv' = u\,\partial_B(wv'w')$ thus according to (14), $wv' = \partial_B(wv'w')$ and since $v = wv'w'$, we get$w \sqsubseteq \partial_B(v)$.

By definition $\partial_B(uv)$ is the shortest prefix of $uv$ such that $uv$=$\partial_B(uv)v'$ with $v' \mathbb{I} B$, thus by hypothesis there exists $w'$ such that $uv = uww'v'$. $v = ww'v'$ thus by definition of $\partial_B(v)$ again, $ww' \sqsubseteq \partial_B(v)$ thus $w \sqsubseteq \partial_B(v)$.

We prove (25). Since $\partial_B(u) \sqsubseteq u$, then $\partial_B((\partial_B(u)) \sqsubseteq \partial_B(u)$ according to (13) it is enough to prove $\partial_B(u) \sqsubseteq \partial_B(\partial_B(u))$. By definition of $\partial_B$, $u = \partial_B(u)u'$ with $u' \mathbb{I} B$ and $\partial_B(u) = \partial_B(\partial_B(u))u''$ with $u'' \mathbb{I} B$. Thus $u = \partial_B(\partial_B(u))u''u'$ with $u''u' \mathbb{I} B$ hence by definition of $\partial_B$, $\partial_B(u) \sqsubseteq \partial_B(\partial_B(u))$.

We prove (26). Since $u\,\partial_B(v) \sqsubseteq uv$, then $\partial_B(u\,\partial_B(v)) \sqsubseteq \partial_B(uv)$ and according to (13) it is enough to prove $\partial_B(uv) \sqsubseteq \partial_B(u\,\partial_B(v))$. By definition of $\partial_B(v)$ there exists $v' \mathbb{I} B$ such that $v = \partial_B(v)v'$ then $uv = u\,\partial_B(v)v'$ thus $\partial_B(uv) \sqsubseteq u\,\partial_B(v)$. According to (25), $\partial_B(\partial_B(uv)) = \partial_B(uv)$ thus $\partial_B(uv) \sqsubseteq \partial_B(u\,\partial_B(v))$ which terminates the proof of (26).

We prove (27). Assume $ua$ is prime. The direct implication is immediate using (24). For the converse implication, assume $av \sqsubseteq \partial_B(avw)$. Since $\partial_B(uavw) = \partial_B(uav\,\partial_B(w))$, without loss of generality we can assume $w = \partial_B(w)$ thus $\partial_B(avw) = avw$, thus we can replace $v$ with $vw$ and assume $w = \epsilon$. Then $\partial_B(av) = av$ thus according to (23) $v = \partial_B(v)$ and $a = \partial_B(a)$. Then $uav$ factorizes as $uav = w_0w_1$ with $w_0 = \partial_B(uav)$ and $w_1 \mathbb{I} B$. Then according to (18), $ua = u_0u_1$ such that $w_0 = u_0z_0$ and $w_1 = u_1z_1$. Since $ua$ is $a$-prime then either $u_1 = \epsilon$ or $u_1$ is $a$-prime. In case $u_1 = \epsilon$ then $w_0 = uaz_0$ thus $ua \sqsubseteq \partial_B(uav)$ and according to (23), $\partial_B(uav) = u\,\partial_B(av) = uav$ so the proof is done. Otherwise, $u_1$ is $a$-prime but $w_1 \in B$ thus $a \mathbb{I} B$, a contradiction with $a = \partial_B(a)$. This terminates the proof of (27). □

## 7 Taking shortcuts

**Proposition 1** A prime play $u$ is a $\mathbb{Q}$-broadcast if and only if for every $uv$ such that $v$ is prime,

$$(uv \text{ is prime}) \text{ or } (v \mathbb{I} \mathbb{Q}) \text{ or } (\mathrm{dom}(v) \subseteq \mathbb{Q}) \ . \tag{29}$$

*Proof.* For the direct implication, assume that $u$ is a $\mathbb{Q}$-broadcast, $v$ is prime and $uv$ is not prime. Let $b$ be the last action of $v$. Then $u \not\sqsubseteq \partial_b(uv)$. Thus by definition of a broadcast, either $\mathrm{dom}(b) \subseteq \mathbb{Q}$ or $\mathrm{dom}(b) \cap \mathbb{Q} = \emptyset$. And by induction the same holds for every letter $b$ of $v$. Since $v$ is prime then either $\mathrm{dom}(v) \subseteq \mathbb{Q}$ or $\mathrm{dom}(v) \cap \mathbb{Q} = \emptyset$. Which terminates the proof of the direct implication.

Conversely, assume that for every $uv$ such that $v$ is prime, condition (29) holds. Let $a$ be a maximal action of $uv$.

If $uv$ is prime then $u \sqsubseteq uv = \partial_a(uv)$ thus the right handside of the implication (3) in the definition of broadcasts is satisfied.

If $uv$ is not prime then according to (29) either $v \mathbb{I} \mathbb{Q}$ or $\mathrm{dom}(v) \subseteq \mathbb{Q}$. In the first case in particular $\mathrm{dom}(a) \cap \mathbb{Q} = \emptyset$. In the second case in particular $\mathrm{dom}(a) \subseteq \mathbb{Q}$. In both cases the implication (3) in the definition of broadcasts is satisfied because the assumption is false.

In both cases the condition defining a $\mathbb{Q}$-broadcast is satisfied. □

The first clause in the disjonction (29) can be reformulated in several ways.

**Proposition 3.** *Let $B \subseteq A$ and $a, b \in A$ and $u, v \in A^*$ such that $u$ is $a$-prime and $v$ is $b$-prime. Then the following conditions are equivalent:*

$$\begin{aligned} (uv \textit{ is } b\textit{-prime}) &\iff \neg(a \mathbb{I} v) \\ &\iff a \sqsubseteq \partial_b(av) \\ &\iff u \sqsubseteq \partial_b(uv) \ . \end{aligned}$$

*Proof.* We prove one by one all implications from the bottom to the top and finally the implication from the very top to the very bottom. Assume ($u \sqsubseteq \partial_b(uv)$) then according to (27) ($a \sqsubseteq \partial_b(av)$). Assume $a \mathbb{I} v$ then in particular $a \mathbb{I} b$ and $\partial_b(av) = \partial_b(v) = v$ because $v$ is $b$-prime thus ($a \not\sqsubseteq \partial_b(av)$). Assume $\neg(a \mathbb{I} v)$. Then $av$ is $b$-prime according to (22) thus $uv$ is $b$-prime according to (21). Assume $uv$ is $b$-prime then $uv = \partial_b(uv)$ thus $u \sqsubseteq \partial_b(uv)$. □

**Lemma 1** *Let $(x, y)$ a useless thread in a distributed strategy $\sigma$. Then the $(x, y, \sigma)$-shortcut $\sigma_{x,y}$ is a distributed strategy.*

*Proof.* We denote $\tau = \sigma_{x,y} = \sigma \circ \phi_{x,y}$ the $(x, y, \sigma)$-shortcut. To prove that $\tau$ is a distributed strategy, we take any process $p \in \mathbb{P}$ and $u \in A^*$ and prove that

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

By definition of $v$ and the shortcut $\tau$, $\tau_p(u) = \tau_p(xv) = \sigma_p(xyv)$ and since $\sigma$ is a distributed strategy $\sigma_p(xyv) = \sigma_p(\partial_p(xyv))$, thus it is enough to prove:

$$\sigma_p(\partial_p(xyv)) = \tau_p(\partial_p(xv)). \tag{30}$$

We distinguish between three cases.

First case: assume ($x \not\sqsubseteq u \wedge x \not\sqsubseteq \partial_p(u)$). Then $\tau_p(u) = \sigma_p(u) = \sigma_p(\partial_p(u)) = \tau_p(\partial_p(u))$, where the first and third equality hold by definition of a shortcut, and the second equality holds because $\sigma$ is a distributed strategy. Thus (30) holds in the first case.

Second case: assume $x \sqsubseteq \partial_p(u)$. Since $\partial_p(u) \sqsubseteq u$ this implies $x \sqsubseteq u$, hence there exists $w \in A^*$ such that $u = xw$. We start with proving

$$\partial_p(xyv) = xy\,\partial_p(v) \ . \tag{31}$$

Since $x \sqsubseteq \partial_p(xw)$, (23) implies

$$\partial_p(xw) = x\,\partial_p(w) \ . \tag{32}$$

Since $(x, y)$ is a useless thread, acccording to (4), both $x$ and $xy$ are $b$-prime. Since moreover $\partial_p(xw) = x\,\partial_p(w)$ we can apply (27) twice and get first $\partial_p(bw) = b\,\partial_p(w)$ and then (31). Now that (31) is proved we can conclude the second case:

$$\begin{aligned} \sigma_p(\partial_p(xyw)) &= \sigma_p(xy\,\partial_p(w)) && (33)\\ &= \tau_p(x\,\partial_p(w)) && (34)\\ &= \tau_p(\partial_p(xw)) \ , && (35) \end{aligned}$$

where (33) comes from (31), (34) hold by definition of shortcuts and $\tau$, and (35) comes from (32). Thus (30) holds in the second case.

We are now left with the third and last case:

$$x \not\sqsubseteq \partial_p(u) \wedge x \sqsubseteq u, \tag{36}$$

which we assume until the end of the proof. Then $u = xv$ for some $v \in A^*$.

We first take care of the special case where $\partial_p(v) = \epsilon$ Then $v \mathbb{I} p$ according to (16) thus $\partial_p(xyv) = \partial_p(xy)$. Hence,

$$\begin{aligned}
\sigma_p(\partial_p(xyv)) &= \sigma_p(\partial_p(xy)) && (37)\\
&= \sigma_p(xy) && (38)\\
&= \tau_p(x) && (39)\\
&= \sigma_p(x) && (40)\\
&= \sigma_p(\partial_p(x)) && (41)\\
&= \sigma_p(\partial_p(xv)) && (42)\\
&= \tau_p(\partial_p(xv)) \ , && (43)
\end{aligned}$$

where (37) and (42) hold because $v \mathbb{I} p$, (38) and (41) hold because $\sigma_p$ is a distributed strategy, (39) holds by definition of $\tau$, (40) holds because $(x, y)$ is a useless thread and according to (7), (43) holds by definition of $\tau$ and because by hypothesis $x \not\sqsubseteq \partial_p(u)$. This shows that (30) holds when $\partial_p(v) = \epsilon$.

Now assume that $\partial_p(v) \neq \epsilon$ (and we keep assuming (36) as well). Since $(x, y)$ is a useless thread in $\sigma$, then according to (5), $x$ is a $\mathbb{Q}$-broadcast in $\sigma$. We apply the caracterisation of broadcasts given by Proposition 1, to $x\,\partial_p(v)$, which is allowed because $\partial_p(v)$ is prime and $x\,\partial_p(v)$ is a $\sigma$-play because it is a prefix of the $\sigma$-play $xv$. Thus according to Proposition 3, one of the three following properties holds:

$$\begin{aligned}
& x \sqsubseteq \partial_p(x\,\partial_p(v)) && (44)\\
& \text{or } \ \mathrm{dom}(\partial_p(v)) \subseteq \mathbb{Q} && (45)\\
& \text{or } \ \mathrm{dom}(\partial_p(v)) \ \mathbb{I} \ \mathbb{Q} \ . && (46)
\end{aligned}$$

Since $x \not\sqsubseteq \partial_p(x\,\partial_p(v))$ by hypothesis, (44) is not possible and we are left with the two other cases (45) and (46).

We assume first that (46) holds. Since $\partial_p(v) \neq \epsilon$, it implies that $p \notin \mathbb{Q}$. Since $(x, y)$ is a $\mathbb{Q}$-thread then $\mathrm{dom}(y) \subseteq \mathbb{Q}$ thus $p \mathbb{I} y$. We can conclude the proof of (30) in the case where (46) holds:

$$\begin{aligned}
\sigma_p(\partial_p(xyv)) &= \sigma_p(\partial_p(xy\,\partial_p(v))) && (47)\\
&= \sigma_p(\partial_p(x\,\partial_p(v)y)) && (48)\\
&= \sigma_p(\partial_p(x\,\partial_p(v))) && (49)\\
&= \sigma_p(\partial_p(xv)) && (50)\\
&= \tau_p(\partial_p(xv)) \ , && (51)
\end{aligned}$$

where (47) and (50) hold according to (26), (48) holds because $\partial_p(v) \mathbb{I} \mathbb{Q}$ and $\mathrm{dom}(y) \subseteq \mathbb{Q}$, (49) holds because $p \mathbb{I} y$, and (51) holds by definition of $\tau_p$, since by hypothesis $x \not\sqsubseteq \partial_p(u)$ and $u = xv$. This proves (30) in the case where (46) holds.

Now we are left with the case where (45) holds, i.e. $\mathrm{dom}(\partial_p(v)) \subseteq \mathbb{Q}^*$ (and we keep assuming $\partial_p(v) \neq \epsilon$ and (36) as well). We first establish

$$\partial_p(v) \in (A \setminus \{b\})^*. \tag{52}$$

Since by hypothesis $x \not\sqsubseteq \partial_p(xv)$ and $x$ is $b$-prime then according to (27), $b \not\sqsubseteq \partial_p(bv)$ thus according to (22), $b \mathbb{I} \partial_p(v)$ which implies (52). Since $(x, y)$ is a useless thread in $\sigma$, we can apply (7) to $\partial_p(v)$ hence

$$\sigma_p(x\,\partial_p(v)) = \sigma_p(xy\,\partial_p(v)) \ . \tag{53}$$

Finally,

$$\begin{aligned}
\sigma_p(\partial_p(xyv)) &= \sigma_p(\partial_p(xy\,\partial_p(v))) && (54)\\
&= \sigma_p(xy\,\partial_p(v)) && (55)\\
&= \sigma_p(x\,\partial_p(v)) && (56)\\
&= \sigma_p(\partial_p(x\,\partial_p(v))) && (57)\\
&= \sigma_p(\partial_p(xv)) && (58)\\
&= \tau_p(\partial_p(xv)), && (59)
\end{aligned}$$

where equalities (54) and (58) hold according to (26), equalities (55) and (57) hold because $\sigma$ is a distributed strategy, (56) comes from (53), and finally (59) is by definition of $\tau$ and because by hypothesis $x \not\sqsubseteq \partial_p(xv)$. This terminates the proof of (30) in the last case.

As a consequence, $\tau$ is a distributed strategy. □

**Lemma 2** *Let* $(x, y)$ a useless thread in a winning distributed strategy $\sigma$. Then the $(x, y, \sigma)$-shortcut $\sigma_{x,y}$ is a winning distributed strategy as well, and

$$\mathrm{dur}(\sigma_{x,y}) < \mathrm{dur}(\sigma) \ . \tag{60}$$

Moreover for every $v \in A^*$,

$$xv \text{ is a } \sigma_{x,y}\text{-play} \iff xyv \text{ is a } \sigma\text{-play.} \tag{61}$$

*Proof.* We denote $\tau = \sigma_{x,y} = \sigma \circ \phi_{x,y}$ the $(x, y, \sigma)$-shortcut.

We first prove property (61). Let $xv \in A^*$ be a $\tau$-play, we prove that $xyv$ is a $\sigma$-play by induction on $v$. When $v = \epsilon$ then $xy$ is a $\sigma$-play because by hypothesis $(x, y)$ is a thread. For the inductive step, assume $xyv$ is a $\sigma$-play, let $c \in A$ such that $xvc$ is a $\tau$-play, and let us prove that $xyvc$ is a $\sigma$-play. Since $xvc$ is a $\tau$-play, $c \in \tau_p(xv)$ for every $p \in \mathrm{dom}(c)$. Thus by definition of $\tau$, $c \in \sigma_p(xyv)$ for every $p \in \mathrm{dom}(c)$ hence $xyvc$ is a $\sigma$-play by definition of $\sigma$-plays.

Now we prove that $\tau$ is winning. Since $\sigma$ is winning, the set of $\sigma$-plays is finite. According to property (61) and the definition of $\tau$, every $\tau$-play is either a $\sigma$-play or is a subword of a $\sigma$-play thus $K$ is also an upper bound on the length of $\tau$-plays, hence every maximal $\tau$-play is finite. Let $u$ be a maximal $\tau$-play. If $x \not\sqsubseteq u$ then $u$ is a maximal $\sigma$-play and since $\sigma$ is winning $u$ is a winning play. Assume now that $x \sqsubseteq u$ and $u = xw$. According to (61), since $xw$ is a maximal $\tau$-play, $xyw$ is a maximal $\sigma$-play, and since $\sigma$ is winning, all processes are in a

final state $xyw$. Since $(x, y)$ is a useless shell, (6) states that all processes are in the same state in $x$ and $xy$, and since transitions are deterministic, all processes are in the same state in $xw$ and $xyw$. So finally all processes are in a final state in $xw$. Thus $\tau$ is winning.

Now we prove property (60). According to (61), the mapping $\phi_{x,y}$ used to define $\tau = \sigma_{x,y}$ in (2) maps maximal $\tau$-plays to maximal $\sigma$-plays. Moreover, according to (14), $\Phi$ is an injection, and by definition it preserves the length on $\{u \mid x \not\sqsubseteq u\}$ and increases the length of $|y|$ on $\{u \mid x \sqsubseteq u\}$. This shows that $\mathrm{len}(\sigma) \geq \mathrm{len}(\tau) + q \cdot |y|$ where $q$ is the number of maximal $\tau$-plays prefixed by $x$. Since $x$ is a $\tau$-play, $q \geq 1$. According to (4), $y \neq \epsilon$ thus we get property (60).

This terminates the proof of Lemma 2. □

## 8 Broadcast games

**Proposition 2** *It is decidable whether a game $G$ is a broadcast game. In case it is then $G$ is an $N$-broadcast-game for some $N \leq \Pi_{p\in\mathbb{P}}|Q_p|$.*

*Proof.* Let $M = \Pi_{p\in\mathbb{P}}|Q_p|$. Let $\preceq$ be an inductive decomposition of $A$. A standard pumping argument of automata theory shows that the conditions in Definition 13 are satisfied for some $N$ is and only if they are satisfied when $N = M$ and both traces $u$ and $v$ have length less than $M$.

Let $b \in A$ and $\mathbb{Q} \subseteq \mathbb{P}$. Let $I_b = \{a \in A \mid a \mathbb{I} b\}$ and $I_{\mathbb{Q}} = \{a \in A \mid a \mathbb{I} \mathbb{Q}\}$. Then a $b$-primary play $u$ is a $\mathbb{Q}$-broadcast if and only if there does not exist a prime trace $v \in A^*$ such that $uv$ is a play and

$$(v \notin I_b^*) \tag{62}$$
$$\wedge\, (\mathrm{dom}(v) \not\subseteq \mathbb{Q}) \tag{63}$$
$$\wedge\, (v \notin I_{\mathbb{Q}}^*)\ . \tag{64}$$

The conjonction of these three conditions is indeed equivalent to the opposite of the definition fo broadcasts given in Proposition 1. Indeed, according to Proposition 3,

$$(uv \text{ is prime}) \iff \neg(v \mathbb{I} b) \iff (v \notin I_b^*).$$

Again, a standard pumping argument shows that if there exists $v \in A^*$ which satisfies (62), (63) and (64) and such that $uv$ is a play then $v$ can be chosen of length at most $3M$.

Thus, the proposition holds since whether $G$ is a broadcast game or not can be decided by enumerating all $N \leq M$ and every $\mathbb{Q} \subseteq \mathbb{P}$ and for each of those, enumerating all $u, v$ of length less than $M$, and for those which satisfies the conditions in Definition 13, enumerating all prefixes $v' \sqsubseteq v$ and check whether (62), (63) and (64) are satisfied with $u$ replaced by $uv'$. If a witness is found then $G$ is not a broadcast game, otherwise $G$ is a broadcast game. □