Distributed games with causal memory are decidable for series-parallel systems*

Paul Gastin, Benjamin Lerman, and Marc Zeitoun

LIAFA, Université Paris 7 & CNRS
2, pl. Jussieu, case 7014, F-75251 Paris cedex 05, France
{Paul.Gastin, Benjamin.Lerman, Marc.Zeitoun}@liafa.jussieu.fr

Abstract. This paper deals with distributed control problems by means of distributed games played on Mazurkiewicz traces. The main difference with other notions of distributed games recently introduced is that, instead of having a *local* view, strategies and controllers are able to use a more accurate memory, based on their *causal* view. Our main result states that using the causal view makes the control synthesis problem decidable for series-parallel systems for *all* recognizable winning conditions on finite behaviors, while this problem with local view was proved undecidable even for reachability conditions.

1 Introduction

This paper addresses a distributed control problem. We are given a *distributed open system* interacting with its environment. While actions of the environment cannot be controlled, actions performed by the system are controllable. We are also given a *specification* and the problem is to find, for each local process, a finite-state *local* controller such that synchronizing each local process with its local controller makes an overall system satisfying the specification.

Sequential versions of control problems have been studied for a long time [2, 16, 14] and have usually decidable answers. What makes the *distributed* control problem more difficult is that a given process and its associated local controller only have a partial view of what happened so far. For instance, a controller cannot take a decision depending on what occurred on a concurrent process, unless such information is forwarded to it (via another process or via the environment).

The problem can be modeled by a game with incomplete information. Each process is a player of the controller team and the environment is the other team. Finding a distributed controller is then equivalent to computing a distributed winning strategy for the controller team. The general situation for multiplayer games with incomplete information is undecidable [13, 12] and in this light, it is not really surprising that the distributed control problem is undecidable even for simple specifications [15, 7, 8, 10]. The aim of the present paper is to open a breach in the list of undecidable results for distributed systems. Obtaining

* Work partly supported by the European research project HPRN-CT-2002-00283 GAMES and by the ACI Sécurité Informatique 2003-22 (VERSYDIS).

efficient and broadly applicable algorithms is a long term issue, out of the scope of the paper.

We believe that there are two main reasons for undecidability results obtained in previous works and that they are both related to the fact that interleavings were used to model distributed behaviors. First, specifications were often given as regular conditions on *linearizations* and were not necessarily closed under commutations of independent actions. This is a well-known cause of undecidability, already addressed in [10]. The second reason has to do with the memory that local controllers are allowed to use. This memory is an abstraction of the part of the behavior that the controller is able to see. Previous works used a *local view*: a process can only see its own previous actions. However, the distributed control problem remains undecidable using this local view even when specification are both regular and closed under commutations [10]. For distributed games defined in [11], even reachability specifications are undecidable [1].

In our work, the local memory is based on the *causal view* of a process (a notion which already yielded positive decidability results for branching-time specifications [9]). This causal view is more accurate than the local one and includes all actions that are *causally* in the past of the current local action. Importantly, this causal memory can be implemented for reasonable communication architectures by forwarding additional informations along with usual messages. The main contribution of this paper is that, if we use causal memory, the distributed control problem becomes decidable for series-parallel systems and for *controlled reachability* conditions, encompassing specifications such as recognizability on finite behaviors, and reachability and safety conditions (on finite or infinite behaviors). Further, one can effectively compute a distributed controller when it exists. This result contrasts deeply with previous work since the problem is undecidable with local memory. Our proof is based on a structural induction that is possible for series-parallel systems.

The causal view was also considered in [6]. It was shown that distributed games with causal memory are undecidable for rational winning conditions on linearizations even for cograph dependence alphabets. This explains why we consider only recognizable winning conditions in this paper.

The distributed control problem remains open for classical conditions on infinite traces such as Büchi, liveness, parity conditions, ... We conjecture that these problems are still decidable. Another important issue is to exhibit a more direct construction of the winning strategies. Finally, the distributed control problem is still open, even for finite behaviors, on non-cograph alphabets.

Due to lack of space, most proofs had to be omitted.

2 Definitions and notation

Mazurkiewicz traces. We briefly recall definitions for our models of distributed behaviors, see [4] for details.

If $(V, \leq)$ is a poset and $S \subseteq V$, the past of S is $\downarrow S = \{x \in V \mid \exists s \in S, x \leq s\}$. If $x \in V$, we write $\downarrow x$ for $\downarrow\{x\}$ and we let $\Downarrow x = \downarrow x \setminus \{x\}$ be the strict past of x. The successor relation associated with the partial order $<$ is $\lessdot = < \setminus <^2$.

A *dependence alphabet* is a pair (Σ, D) where Σ is a finite alphabet and D is a reflexive, symmetric binary relation over Σ, called the *dependence relation*. For $A \subseteq \Sigma$, we let $D(A)$ be the set of letters that depend on some letters in A.

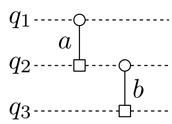
A *(Mazurkiewicz) trace* over (Σ, D) is an isomorphism class $[V, \leq, \ell]$ of a pomset such that for all $x, y \in V$: (1) $\ell(x)\ D\ \ell(y) \Rightarrow x \leq y$ or $y \leq x$, (2) $x \lessdot y \Rightarrow \ell(x) D \ell(y)$ and (3) $\downarrow x$ is finite. We denote by $\mathbb{R}(\Sigma, D)$ (resp. by $\mathbb{M}(\Sigma, D)$) the set of traces (resp. of finite traces) over (Σ, D).

If $t = [V, \leq, \ell]$ is a trace, we denote by $\max(t)$ (resp. by $\min(t)$) the set of maximal (resp. minimal) elements of t. The *alphabet* of t is $\mathrm{alph}(t) = \ell(V)$. A *prefix* of t is a trace $s = [U, \leq, \ell]$, where $U \subseteq V$ satisfies $\downarrow U = U$. We write $s \leq t$ if s is a prefix of t. In this case, we let $s^{-1}t = [V \setminus U, \leq, \ell]$. The empty trace is denoted by ε.

Distributed games. The distributed systems we want to control are based on asynchronous automata [18]. We are given a finite set of processes communicating asynchronously via shared memory variables. Each process stores a value in a register. When executing, an action reads registers of some processes, and then writes some other registers through a test-and-set instruction. Some actions are controllable. The other ones, representing the environment's actions, are uncontrollable.

We model these systems by distributed games [6] over a given *architecture* $(\Sigma, \mathcal{P}, R, W)$. Here, $\mathcal{P}$ is a finite set of *processes*, $\Sigma = \Sigma_0 \uplus \Sigma_1$ is a finite set of *players* (or *actions*), where Σ_0 is the set of players of team 0 (the controller) and Σ_1 the set of players of team 1 (the environment). Player $a \in \Sigma$ can atomically read states of processes in $R(a) \subseteq \mathcal{P}$ and write new states on processes in $W(a) \subseteq \mathcal{P}$. We require two natural restrictions also considered in [18].

$$\begin{aligned} &\forall a \in \Sigma, && \emptyset \neq W(a) \subseteq R(a) \\ &\forall a, b \in \Sigma, && R(a) \cap W(b) = \emptyset \iff R(b) \cap W(a) = \emptyset \end{aligned} \qquad (\mathcal{S})$$



These conditions encompass in particular all purely asynchronous architectures (*i.e.*, such that $R = W$) and all cellular architectures (*i.e.*, such that $|W(a)| = 1$ for all $a \in \Sigma$). In contrast, we do not treat here "one way" communication architectures, as the one depicted opposite, where circles represent processes read by the corresponding player, and squares represent processes which are both read and written. That is, $R(a) = \{q_1, q_2\}$, $W(a) = \{q_2\}$, $R(b) = \{q_2, q_3\}$, $W(b) = \{q_3\}$, which obviously violates $(\mathcal{S})$.

A *distributed game* over the architecture $(\Sigma, \mathcal{P}, R, W)$ is given by a tuple $G = (\Sigma_0, \Sigma_1, (Q_i)_{i \in \mathcal{P}}, (T_a)_{a \in \Sigma}, q^0, \mathcal{W})$, where Q_i is the set of local states (register values) of process i. Given $I \subseteq \mathcal{P}$, we let $Q_I = \prod_{i \in I} Q_i$, and if $q = (q_i)_{i \in \mathcal{P}} \in Q_{\mathcal{P}}$, we let $q_I = (q_i)_{i \in I}$. A global state of the game is then a tuple $q \in Q_{\mathcal{P}}$. Player a has a table of legal moves $T_a \subseteq Q_{R(a)} \times Q_{W(a)}$. A (sequential) play is a sequence of moves starting in the global state $q^0 \in Q_{\mathcal{P}}$, the *initial position of the game*. There is an a-move from $p \in Q_{\mathcal{P}}$ to $q \in Q_{\mathcal{P}}$ if $(p_{R(a)}, q_{W(a)}) \in T_a$ and $q_{\mathcal{P}\setminus W(a)} = p_{\mathcal{P}\setminus W(a)}$. The winning condition $\mathcal{W}$ describes a set of desired plays and will be discussed later on.

Note that, if $R(a) \cap W(b) = \emptyset = R(b) \cap W(a)$ then in any global state p the two moves a and b can be executed simultaneously or in any order without affecting the resulting global state q: they are independent. Therefore, a (distributed) play of a distributed game is more accurately defined by an equivalence class of sequential plays, or equivalently, by a Mazurkiewicz trace over a suitable dependence alphabet.

Distributed plays will be defined as traces with doubly labeled vertices: the first label is the player's name, and the second one is a vector of local states representing what was written by the player. Formally, we consider a new symbol $\# \notin \Sigma$, with $R(\#) = W(\#) = \mathcal{P}$. Let then $\Sigma' = \{(a,p) \mid a \in \Sigma \uplus \{\#\} \text{ and } p \in Q_{W(a)}\}$. We define the dependence relation D over $\Sigma \uplus \{\#\}$ by $a \, D \, b \Leftrightarrow R(a) \cap W(b) \neq \emptyset \Leftrightarrow R(b) \cap W(a) \neq \emptyset$ and D' over Σ' by $(a,p) \; D' \; (b,q) \Leftrightarrow a \; D \; b$. We write a trace of $\mathbb{R}(\Sigma', D')$ as $[V, \leq, \ell, \sigma]$, where $\ell : V \to \Sigma \uplus \{\#\}$ and $\sigma : V \to \bigcup_{a \in \Sigma \uplus \{\perp\}} Q_{W(a)}$ together define the labeling: a vertex x is labeled by $(\ell(x), \sigma(x))$. A trace $t = [V, \leq, \ell, \sigma] \in \mathbb{R}(\Sigma', D')$ is *rooted* if $\ell^{-1}(\#) = \{x_\#\}$ is a singleton and $x_\# \leq y$ for all $y \in V$. The global state reached on a finite rooted trace $t \in \mathbb{M}(\Sigma', D')$ is $\overline{q}(t) = (\overline{q}_i(t))_{i \in \mathcal{P}} \in Q_{\mathcal{P}}$ where:

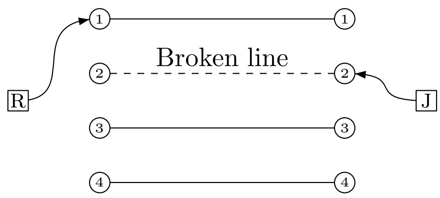
$$\overline{q}_i(t) = (\sigma(y))_i \text{ with } y = \max\{x \in V \mid i \in W(\ell(x))\}.$$

In other words, we retain the last write action performed on each process.

A (distributed) play is a rooted trace $t = [V, \leq, \ell, \sigma] \in \mathbb{R}(\Sigma', D')$ which obeys the rules given by $(T_a)_{a \in \Sigma}$, *i.e.*, $\sigma(x_\#) = q^0$ and

$$\forall x \in V, \quad \ell(x) = a \neq \# \implies (\overline{q}(\Downarrow x)_{R(a)}, \sigma(x)) \in T_a$$

Note that after the beginning of a play, several moves may be enabled, concurrently or not, from Σ_0 or from Σ_1. Thus, we do not see a distributed play as turn-based. The winning condition $\mathcal{W}$ can then formally be defined as a subset of $\mathbb{R}(\Sigma', D')$. Team 0 wins the play t if $t \in \mathcal{W}$.



Example. Romeo and Juliet are in two separate houses and they want to set up an appointment. There are four communication lines of which exactly one is broken. At any time, Romeo (or Juliet) may look at the status of the communication lines to see which one is broken and then chooses to connect to one line (the whole operation is atomic). The environment tries to prevent the communication. For this, at any time, it might look at which line Romeo and Juliet are connected, and then decide to change the broken line (again this operation is atomic). The actions of Romeo and Juliet are independent but they both depend on the action of the environment. The

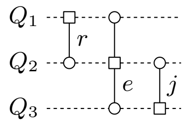
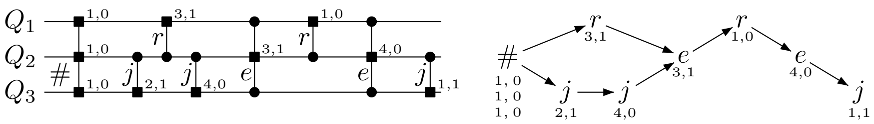


Fig. 1. A simple cograph architecture

problem is to find two strategies, one for Romeo and one for Juliet, so that they end up communicating whatever the environment does. If there is no restriction on the environment then it might monopolize the system by constantly changing the broken line, thus preventing any action by Romeo or Juliet due to the dependence of actions. Therefore we restrict the environment so that it cannot act twice consecutively.

We formalize this system using three processes with states $Q_1 = Q_2 = Q_3 = \{1, 2, 3, 4\} \times \{0, 1\}$ and three players r, e, j whose read and write domains are depicted in Figure 1, where circles represent processes read by the corresponding player, and squares represent processes which are both read and written. State $(1, 0)$ for process 1 means that Romeo is connected to the first line and has played an even number of times. The situation is similar for process 3 and Juliet. State $(2, 1)$ for process 2 means that line number 2 is broken and the environment has played an odd number of times. The environment is allowed to play only if the total number of moves is odd. A process based picture and a Hasse diagram representation of a distributed play are given below. Between moves (whose reads are $\bullet$ and read-writes are $\blacksquare$), we draw local states which get modified by the test-and-set legal moves. For instance, the first e reads (3,1), (1,0), (4,0) on processes 1, 2 and 3 and writes (3,1) on process 2. The global state reached at the end is (1,0), (4,0), (1,1) which is winning for Romeo and Juliet. The interested reader might check that Romeo and Juliet have memoryless strategies to win this game.



Strategies and memory. Intuitively, player a of team 0 can restrict its set of potential moves depending on its own history of the play. In the distributed setting, it would not make sense to define this history on *sequential* plays. Indeed, the strategy of player a should not depend on the ordering of independent moves that are in its past and it should not depend either on concurrent moves that happen to occur before it in some linearization.

A first solution is to define the history of some move a as the sequence of moves that have written on process $W(a)$ (assuming $W(a)$ is a singleton). This is the minimal reasonable amount of information we want to provide to players. This defines strategies with *local memory* [11, 8, 10]. Unfortunately, even games with reachability conditions on the simple architecture given in Figure 1 are undecidable with the local view [1].

The representation of plays by traces provides another natural solution. In order to choose which move to take, player a may look at all the causal past (in the partial order) of the last write-events on the processes in $R(a)$. This is intuitively the *maximal* amount of information we can provide to players. This is technically a bit more complicated since this memory information has to be computed in a distributed way via states. The idea is that any player a can compute, in addition to the values $q_{W(a)}$, a memory value that he also writes in all locations of $W(a)$.

Let $t = [V, \leq, \ell, \sigma] \in \mathbb{R}(\Sigma', D')$ be a rooted trace. For $A \subseteq \Sigma$, the trace $\partial_A t$ is the smallest prefix of t containing all vertices labeled in A under ℓ. For $I \subseteq \mathcal{P}$, the trace $\partial_I t$ is the smallest prefix of t containing all vertices x such that $W(\ell(x)) \cap I \neq \emptyset$.

An asynchronous mapping [3] is a function $\mu : \mathbb{M}(\Sigma', D') \to M$ such that $\mu(\partial_{A \cup B} t)$ only depends on $\mu(\partial_A t)$ and $\mu(\partial_B t)$, and $\mu(\partial_{D(a)} t.a)$ only depends on $\mu(\partial_{D(a)} t)$ and a. Asynchronous mappings can be computed in a distributed way [3]. A *distributed memory* is a computable abstraction of an asynchronous mapping. Formally, $\mu : \mathbb{M}(\Sigma', D') \to M$ is a *distributed memory* if there is a computable asynchronous mapping $\nu : \mathbb{M}(\Sigma', D') \to N$ and a computable function $\pi : N \to M$ such that $\pi \circ \nu = \mu$. The function μ is the information actually needed for a strategy, and the asynchronous mapping ν represents an asynchronous implementation of this memory. Property $(\mathcal{S})$ makes it possible to *implement* causal memory. Indeed, if $x \lessdot y$ in a trace, then by definition $\ell(x) \, D \, \ell(y)$ and therefore, by $(\mathcal{S})$, there is at least one process where x writes and y reads. Hence, information computed by player $\ell(x)$ can be forwarded to player $\ell(y)$. Observe that the environment's team participates to the computation of the causal view. This is not unrealistic: one cannot know when an environment's action will occur, but some systems may be designed so that events of the environment forward the necessary information to compute the needed abstraction of the causal view.

Intuitively, a given memory will be used by players of team 0 as an abstraction (computed in M) of their past in a play. (This is why we call these memories *causal*.) For instance, $\mu(t) = t$ is the largest possible memory and would provide for each player a full view of its past.

The distributed memory $\mu : \mathbb{M}(\Sigma', D') \to M$ is said to be *finite* if it is realized by a finite asynchronous mapping $\nu : \mathbb{M}(\Sigma', D') \to N$. In this case, its *size* is defined as the number of elements of a minimal such N.

A *distributed strategy with memory* $\mu : \mathbb{M}(\Sigma', D') \to M$ for team 0 is a function $f : \bigcup_{a \in \Sigma_0} Q_{R(a)} \times M \times \{a\} \to Q_{W(a)} \cup \{\text{stop}\}$ such that if $f(p, m, a) = q \neq \text{stop}$, then $(p, q) \in T_a$. Intuitively, if $f(p, m, a) = q \neq \text{stop}$, then the strategy f dictates an a-move to $q \in Q_{W(a)}$ on any distributed play $t \in \mathbb{R}(\Sigma', D')$ such that $\partial_{R(a)} t$ is finite, $p = \overline{q}(\partial_{R(a)} t)_{R(a)}$ and $m = \mu(\partial_{R(a)} t)$. If $f(p, m, a) = \text{stop}$, the a-move is disabled by the strategy. Note that several players of team 0 may be simultaneously enabled by f during a play. A distributed play $t = [V, \leq, \ell, \sigma] \in \mathbb{R}(\Sigma', D')$ is an *f-play* if for all $x \in V$ with $\ell(x) \in \Sigma_0$, we have $\sigma(x) = f(\overline{q}(\Downarrow x)_{R(a)}, \mu(\Downarrow x), a)$.

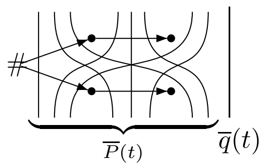
A play t is *f-maximal* if $f(\overline{q}(\partial_{R(a)}t)_{R(a)}, \mu(\partial_{R(a)}t), a) = \text{stop}$ for all $a \in \Sigma_0$ such that $\partial_{R(a)}t$ is finite. The *maximality* condition is natural: if the distributed strategy of team 0 dictates some a-moves at some f-play t, then the f-play t is not over. This applies also if t is infinite and corresponds to some fairness condition: along an infinite f-play, a move of team 0 cannot be ultimately continuously enabled by f without being taken. Note that any f-play t is the prefix of some f-maximal f-play. If each f-maximal f-play is in $\mathcal{W}$ then f is a *winning distributed strategy* (WDS) for team 0.

3 Controlled reachability games

In this section, we introduce controlled reachability games and we prove their decidability on cograph dependence alphabets.
Define the set of global states seen along a rooted (possibly infinite) trace t as

$$\overline{P}(t) = \{\overline{q}(s) \mid s \text{ finite and } \varepsilon < s < t\}$$

Observe that $\overline{q}(t)$ is not necessarily in the set $\overline{P}(t)$.



Define $(\overline{P}, \overline{q})(t) = (\overline{P}(t), \overline{q}(t))$ with $\overline{q}(t) = \infty$ if t is infinite.

Let $G = (\Sigma_0, \Sigma_1, (Q_i)_{i\in\mathcal{P}}, (T_a)_{a\in\Sigma}, q^0, \mathcal{W})$ be a distributed game. Say that G is a *controlled reachability game* if there is a set $\mathcal{F} \subseteq 2^{Q_\mathcal{P}} \times (Q_\mathcal{P} \uplus \{\infty\})$ such that a play t is winning for team 0 iff $(\overline{P}, \overline{q})(t) \in \mathcal{F}$. One will then write $G = (\Sigma_0, \Sigma_1, (Q_i)_{i\in\mathcal{P}}, (T_a)_{a\in\Sigma}, q^0, \mathcal{F})$. Note that we get classical reachability or safety conditions as special cases of controlled reachability conditions.

An undirected graph is a *cograph* if it belongs to the smallest class of graphs containing singletons and closed under parallel product and complementation. Therefore, if (A, D_A) and (B, D_B) are cographs with $A \cap B = \emptyset$, then so are $(A \cup B, D_A \cup D_B)$ and $(A \cup B, D_A \cup D_B \cup A \times B \cup B \times A)$ and vice versa, every cograph can be decomposed using these two operations. All Mazurkiewicz traces on a cograph alphabet are *series-parallel*, that is, they can be described by an expression using only single-vertex traces, and parallel and sequential product of directed graphs. It is well-known that cographs are undirected graphs with no P_4, i.e., no induced subgraph of the form $a{-}b{-}c{-}d$. We can now state the main result of the paper.

Theorem 1. *Given a controlled reachability game on a cograph alphabet, one can decide if team* 0 *has a WDS on this game. One can effectively compute such a strategy if it exists.*

We would like to stress that this theorem might be applied to more general settings than series-parallel systems by adding dependencies (communication channels) and thus turning an arbitrary dependence alphabet into a cograph.

Any recognizable winning condition on finite traces can be reduced to a (controlled) reachability condition by building a product of the game with an asynchronous automaton on the same architecture for this recognizable winning condition.

Corollary 1. *Given a distributed game on a cograph alphabet, with a recognizable set of finite traces as winning condition, one can decide if team* 0 *has a WDS on this game. One can effectively compute such a strategy if it exists.*

To prove Theorem 1, we will build from an arbitrary WDS f with memory μ another WDS f' whose memory is bounded by an effectively computable function depending only on $|\Sigma|$ and $|Q_{\mathcal{P}}|$. By [6], given a distributed memory μ, one can then effectively transform a distributed game G into a new game G^{μ} such that team 0 has a winning distributed strategy with memory μ in G iff it has a memoryless strategy in G^{μ}, which is decidable, again by [6].

We build f' from f by induction on the cograph alphabet. For technical reasons, one proves in the induction the following additional property on f'.

Definition 1. *Let f, f' be two distributed strategies. Then f' is* f-compatible *if for all finite f'-play t', there exists an f-play t such that $(\overline{P}, \overline{q})(t) = (\overline{P}, \overline{q})(t')$.*

Obviously, the compatibility relation is transitive. The following result we shall prove is more accurate than Theorem 1.

Theorem 2. *Let G be a controlled reachability game. There exists a computable function $M : \mathbb{N}^2 \to \mathbb{N}$, such that, for any WDS f over G, there exists a WDS f' which is* f-compatible *and whose memory is bounded by $M(|\Sigma|, |Q_{\mathcal{P}}|)$.*

We start with an intuitive sketch of the proof of Theorem 2. We build the f-compatible WDS f' using strategies obtained by induction over smaller alphabets. The parallel case is easy: the game defines two subgames, one on each alphabet of the parallel product, and f induces WDS's on these subgames. The induction provides compatible WDS's with bounded memory, which we recombine into a new strategy on the parallel product.

The sequential case is more complex. To simplify notation, we write $\max(r) \subseteq A$ instead of $\ell(\max(r)) \subseteq A$ and we use similar a notation for min. We also write $\mathrm{alph}(t)$ instead of $\ell(t)$. On $\Sigma = A \uplus B$, where $A \times B \subseteq D$ (with A, B cographs), f-plays have the form $(\#, q^0)s_1 s_2 \cdots$ where $\mathrm{alph}(s_i) \subseteq A$ iff $\mathrm{alph}(s_{i+1}) \subseteq B$. Each block s_i can be seen as a play $(\#, q^{i-1})s_i$ over A or B where $q^i = \overline{q}((\#, q^0)s_1 \cdots s_i)$. From f, one derives restricted strategies over A or B to go from q^{i-1} to q^i, visiting the same set of states. We then replace, using the induction, these restricted strategies by strategies with bounded memory. This is where controlled reachability is used: the induction hypothesis ensures that states visited by the new strategies are the same as for original strategies. We need this information to ensure we won't reach new states from which team

1 could escape in the other alphabet. By simplifying the strategy f (removing unwanted loops from all f-plays), this makes it possible to recombine the strategies over the smaller alphabets to obtain the desired strategy f'. We also have to prove that players of team 0 can detect with a distributed memory if they are minimal in some block s_i, to know which restricted strategy they have to play. The rest of this section is devoted to the formal proof of Theorem 2.

Induction basis: $|\Sigma| = 1$.

In this case, G is a 1-player sequential game. If we do not insist on getting an f-compatible strategy for f' it would be enough to observe that the winning condition is recognizable (alphabetic condition on states seen along the play and reachability on the possible last state) and computable by an automaton of size $2^{|Q_{\mathcal{P}}|}$. The existence of a winning strategy is therefore equivalent to the existence of a winning strategy with memory less than $2^{|Q_{\mathcal{P}}|}$. However, the strategy f' we have to build must be f-compatible, and we cannot use directly this result. For our proof, we distinguishes two cases.

1. $\Sigma = \{a\} = \Sigma_1$. Then, the set of plays does not depend on the strategy of team 0, since team 0 has no choice. Hence, if team 0 has a winning strategy, this winning strategy is memoryless.
2. $\Sigma = \{a\} = \Sigma_0$. Then, if player a has a winning strategy f, there exists a unique f-maximal f-play r and this play is winning. It is possible to show that one can build from r a new play t satisfying the following three conditions:

$$\forall s, s' \leq t, \quad (\overline{P}, \overline{q})(s) = (\overline{P}, \overline{q})(s') \Rightarrow s^{-1}t = s'^{-1}t \tag{1}$$

$$(\overline{P}, \overline{q})(t) = (\overline{P}, \overline{q})(r) \tag{2}$$

$$\forall t' \leq t, \exists r' \leq r, \quad (\overline{P}, \overline{q})(t') = (\overline{P}, \overline{q})(r') \tag{3}$$

Observe that property (1) guarantees that t is played according to a strategy f' with memory $\mu : s \to (\overline{P}, \overline{q})(s)$ which is indeed distributed since Σ is a singleton. Property (2) ensures that f' is winning, while (3) implies that f' is f-compatible. It follows that $M(1, |Q_{\mathcal{P}}|) \leq |Q_{\mathcal{P}}| \cdot 2^{|Q_{\mathcal{P}}|}$.

Induction, first case: $\Sigma = A \uplus B$ **with** $(A \times B) \cap D = \emptyset$

Without loss of generality, we may assume that $\mathcal{P} = R(A) \cup R(B)$ and that $R(A) \cap R(B) = \emptyset$. Indeed, since $R(A) \cap W(B) = R(B) \cap W(A) = \emptyset$, we have $i \notin W(A) \cup W(B)$ if $i \in R(A) \cap R(B)$. In other terms, such a component i remains constant along a run, and does not play any role during the moves.

Abusing notation we write q_A instead of $q_{R(A)}$ for $q \in Q_{\mathcal{P}}$. Let $q_A \in Q_{R(A)}$ and $q_B \in Q_{R(B)}$. One defines $q = q_A \parallel q_B$ by $q_i = (q_A)_i$ if $i \in R(A)$ and $q_i = (q_B)_i$ if $i \in R(B)$. Further, $\infty \parallel q_B = q_A \parallel \infty = \infty \parallel \infty = \infty$. One extends this definition to pairs of $2^{Q_{\mathcal{P}}} \times (Q_{\mathcal{P}} \uplus \{\infty\})$ by

$$(P, q) \parallel (P', q') = \big((((P \cup \{q\} \setminus \{\infty\}) \parallel P') \cup (P \parallel (P' \cup \{q'\} \setminus \{\infty\}))), q \parallel q'\big).$$

Let $A' = \{(a, p) \in \Sigma' \mid a \in A\} \cup (\{\#\} \times Q_{R(A)})$, and B' be defined similarly. Let $r_A = (\#, q_A^0) \cdot s_A$ be a rooted trace over A' and $r_B = (\#, q_B^0) \cdot s_B$ a rooted trace over B'. Define $r_A \parallel r_B = (\#, q_A^0 \parallel q_B^0) \cdot s_A \cdot s_B = (\#, q_A^0 \parallel q_B^0) \cdot s_B \cdot s_A$.

Lemma 1. *Let r_A and r_B be two rooted traces on the alphabets A' and B' respectively. Then $(\overline{P}, \overline{q})(r_A \parallel r_B) = (\overline{P}, \overline{q})(r_A) \parallel (\overline{P}, \overline{q})(r_B)$.*

A rooted trace t over Σ' can be uniquely factorized as $t = t_A \parallel t_B$ where t_A and t_B are rooted traces over alphabets A' and B' respectively.

If $r = r_A \parallel r_B$ and $s = s_A \parallel s_B$ are f-plays on G then $r_A \parallel s_B$ is again an f-play of G. Indeed, since $A \times B \cap D = \emptyset$, the strict past of a vertex of $r_A \parallel s_B$ is either $(\#, q^0)$, or the same as that of the corresponding vertex in r or in s (depending on whether $\ell(x) \in A$ or $\ell(x) \in B$). If r and s are f-maximal, then $r_A \parallel s_B$ is also f-maximal since, if $c \in A$ for instance, $\partial_{R(c)}(r_A \parallel s_B) = \partial_{R(c)}(r_A \parallel (\#, q_B^0)) = \partial_{R(c)}(r)$.

The set S of f-maximal f-plays is therefore of the form $S = S_A \parallel S_B$. Let $\mathcal{F}_A = (\overline{P}, \overline{q})(S_A)$ and $\mathcal{F}_B = (\overline{P}, \overline{q})(S_B)$. Let us show that $\mathcal{F}_A \parallel \mathcal{F}_B \subseteq \mathcal{F}$. Let $r_A \in S_A$ and $s_B \in S_B$. By definition, there exists r_B and s_A such that $r = r_A \parallel r_B \in S$ and $s = s_A \parallel s_B \in S$. We have seen that this implies $t = r_A \parallel s_B \in S$. Using Lemma 1, one gets $(\overline{P}, \overline{q})(r_A) \parallel (\overline{P}, \overline{q})(s_B) = (\overline{P}, \overline{q})(t) \in \mathcal{F}$, since the strategy f is winning and $t \in S$.

Let $G_A = (\Sigma_0 \cap A, \Sigma_1 \cap A, (Q_i)_{i \in R(A)}, (T_a)_{a \in A}, q_A^0, \mathcal{F}_A)$ on the architecture $(A, R(A), R_{|A}, W_{|A})$. This is again a distributed game. Define G_B symmetrically.

Define $f_A(\overline{q}(r_A), \mu(r_A), a) = f(\overline{q}(r_A \parallel (\#, q_B^0)), \mu(r_A \parallel (\#, q_B^0)), a)$. Let us show that f_A is a WDS for G_A. First, f_A is a distributed strategy with memory μ in G_A, since one can associate to any play r_A of G_A the play $r_A \parallel (\#, q_B^0)$ of G. It remains to show that f_A is winning. Consider an f_A-maximal f_A-play r_A and let r_B be an f_B-maximal f_B-play of G_B. Then, $r_A \parallel r_B$ is an f-maximal f-play. Hence, $(\overline{P}, \overline{q})(r_A) \in \mathcal{F}_A$, and r_A is a winning play of G_A.

By induction, there exists an f_A-compatible winning strategy f'_A for team $\Sigma_0 \cap A$ in G_A with memory μ_A of size less that $M(|A|, |Q_{\mathcal{P}}|)$, and dually for B. We define the memory μ on $\mathbb{M}(\Sigma', D')$ by $\mu(t) = (\mu_A(t_A), \mu_B(t_B))$ for $t = t_A \parallel t_B$. We build from f'_A and f'_B an f-compatible winning strategy f' for Σ_0 in G as follows. For $a \in A$ and $q_{R(a)} \in Q_{R(a)}$, we define $f'(q_{R(a)}, (m_A, m_B), a) = f'_A(q_{R(a)}, m_A, a)$ and similarly, we let $f'(q_{R(b)}, (m_A, m_B), b) = f'_B(q_{R(b)}, m_B, b)$ for $b \in B$ and $q_{R(b)} \in Q_{R(b)}$.

Using the next statement, one can bound the memory of f' by a function depending only on $M(|A|, |Q_{\mathcal{P}}|)$ and $M(|B|, |Q_{\mathcal{P}}|)$, which finishes the induction for the parallel case.

Lemma 2. *The strategy f' is an f-compatible WDS for team 0 on G.*

Induction, second case: $\Sigma = A \uplus B$ with $(A \times B) \subseteq D$.

We define the product $r \cdot_A s$ by $r \cdot_A s = rs$ if $\max(r) \nsubseteq A$ and $\min(s) \subseteq A$. The product is undefined otherwise. Let f be a WDS for team 0 on G and let S the set of all f-plays. If t is a finite f-play, we let $t^{-1}S = \{t^{-1}S \mid t \leq s \text{ and } s \in S\}$ and $\mathrm{From}_A(t) = t^{-1}S \cap (\min \subseteq A)$. We also define

$$\mathrm{Cut}_{A,P,q} = \{t \in S \mid t \text{ is finite}, \ \max(t) \nsubseteq A \text{ and } (\overline{P}, \overline{q})(t) = (P, q)\}.$$

A distributed strategy f is (A, P, q)-*uniform* if for all $r_1, r_2 \in \mathrm{Cut}_{A,P,q}$, we have $\mathrm{From}_A(r_1) = \mathrm{From}_A(r_2)$. Say that f is *uniform* if it is (A, P, q)-*uniform* and (B, P, q)-*uniform* for all $(P, q) \in 2^{Q_{\mathcal{P}}} \times Q_{\mathcal{P}}$.

Lemma 3. *For any winning distributed strategy f on G, there exists a winning f-compatible distributed strategy on G, which in addition is uniform.*

Thanks to Lemma 3 and using the transitivity of the compatibility relation, we may assume that f is uniform for the rest of the proof of Theorem 2. Let then

$$\mathrm{Next}_A(t) = \mathrm{From}_A(t) \cap (\mathrm{alph} \subseteq A)$$

A play r is (f, A)-maximal if for all $a \in A \cap \Sigma_0$, $f(\overline{q}(r)_{R(a)}, \mu(r), a) = \{\mathrm{stop}\}$. If $\mathrm{Cut}_{A,P,q} \neq \emptyset$, we choose $r \in \mathrm{Cut}_{A,P,q}$ and define a winning condition $\mathcal{F}_{A,P,q}$:

$$\mathcal{F}_{A,P,q} = \big\{(\overline{P}, \overline{q})((\#, q)s) \mid s \in \mathrm{Next}_A(r) \text{ and } rs \text{ is } (f, A)\text{-maximal}\big\}$$

Since f is uniform, $\mathrm{From}_A(r)$ and $\mathrm{Next}_A(r)$ do not depend on r. One shows that if rs is (f, A)-maximal, then for all $r' \in \mathrm{Cut}_{A,P,q}$, $r's$ is also f_A-maximal. One deduces that $\mathcal{F}_{A,P,q}$ does not depend on the choice of r.

If $\mathrm{Cut}_{A,P,q} \neq \emptyset$, define $G_{A,P,q} = (\Sigma_0 \cap A, \Sigma_1 \cap A, (Q_i)_{i \in \mathcal{P}}, (T_a)_{a \in A}, q, \mathcal{F}_{A,P,q})$. From f, one can derive a distributed strategy for the distributed game $G_{A,P,q}$:

$$f_{A,P,q}(\overline{q}((\#, q)s), \mu((\#, q)s), a) = f(\overline{q}(rs), \mu(rs), a) \text{ where } r \in \mathrm{Cut}_{A,P,q}$$

Since f is uniform, $f_{A,P,q}$ does not depend on r, and by construction of $f_{A,P,q}$, the set of $f_{A,P,q}$-plays is exactly $(\#, q)\mathrm{Next}_A(r)$. By construction of $G_{A,P,q}$ and $\mathcal{F}_{A,P,q}$, all $f_{A,P,q}$-maximal $f_{A,P,q}$-plays are winning in G_A, so $f_{A,P,q}$ is winning.

Moreover, $G_{A,P,q}$ is a controlled reachability game on the alphabet A, smaller than Σ. By induction, there exists a winning strategy $f'_{A,P,q}$ on $G_{A,P,q}$ which is $f_{A,P,q}$-compatible and whose memory is of size at most $M(|A|, |Q_{\mathcal{P}}|)$. One easily transforms $f'_{A,P,q}$ to ensure that if $(\emptyset, q) \in \mathcal{F}_{A,P,q}$, then $f'_{A,P,q}((\#, q), a) = \{\mathrm{stop}\}$ for all $a \in \Sigma_0 \cap A$. This modification does not change the amount of memory necessary for $f'_{A,P,q}$. Further, $f'_{A,P,q}$ is still $f_{A,P,q}$-compatible and winning.

We now have WDS on smaller games $G_{A,P,q}, G_{B,P,q}$ whose memories have a controlled size. It remains to glue them suitably to reconstruct the f-compatible WDS f'. For this, we need to know on which subgame (A, P, q) or (B, P, q) to play. To this aim, we have to compute necessary information with a distributed memory: The *lb-factorization* (for last-block factorization) of a rooted trace $t \neq (\#, q^0)$ is defined (in a unique way) as the factorization $t = rs$ such that

$$t = \begin{cases} r \cdot_A s \text{ with } \emptyset \neq \mathrm{alph}(s) \subseteq A \\ r \cdot_B s \text{ with } \emptyset \neq \mathrm{alph}(s) \subseteq B. \end{cases}$$

One can write an $\mathrm{MSO}_{\Sigma'}(\leq)$-formula $\mathrm{Lastcut}_{P,q}$ which is satisfied by a trace t if and only if $(\overline{P}, \overline{q})(r) = (P, q)$ where $t = rs$ is the lb-factorization of t. Now, an $\mathrm{MSO}_{\Sigma'}(\leq)$-definable trace language can be accepted by an asynchronous mapping [17, 5, 3]. Hence, the mapping $t \mapsto (\overline{P}, \overline{q})(r)$ where $t = rs$ is the lb-factorization of t is a distributed memory. Similarly, one can show that a mapping indicating to a player if its move (if played) would change the alphabet from A to B or from B to A, is also a distributed memory. These informations give exactly the needed information to players of team 0 to know in which game they are playing. Hence, they make it possible to glue strategies $f'_{A,P,q}, f'_{B,P,q}$ to obtain the desired f-compatible WDS f'. For lack of space, we cannot provide details for this construction. Since we have bounded the sizes of the memories used by the small strategies, this gives us a bound for the memory needed for f'.

Acknowledgements The authors wish to thank the anonymous referees for their careful reading of the submitted version of the paper, which helped us improve its presentation. We also thank J. Bernet, D. Janin and I. Walukiewicz for fruitful discussions.

References

1. J. Bernet, D. Janin, and I. Walukiewicz. Private communication. 2004.
2. J. R. Büchi and L. H. Landweber. Solving sequential conditions by finite-state strategies. *Trans. Amer. Math. Soc.*, 138:295–311, 1969.
3. R. Cori, Y. Métivier, and W. Zielonka. Asynchronous mappings and asynchronous cellular automata. *Inform. and Comput.*, 106:159–202, 1993.
4. V. Diekert and G. Rozenberg, editors. *The Book of Traces*. World Scientific, 1995.
5. W. Ebinger and A. Muscholl. Logical definability on infinite traces. *Theoret. Comput. Sci.*, 154(1):67–84, 1996. Conference version in ICALP '93.
6. P. Gastin, B. Lerman, and M. Zeitoun. Distributed games and distributed control for asynchronous systems. In *LATIN04*, volume 2976 of *LNCS*, pages 455–465. Springer, 2004.
7. O. Kupferman and M. Y. Vardi. Synthesizing distributed systems. In *LICS '01*, pages 389–398. Computer Society Press, 2001.
8. P. Madhusudan and P. S. Thiagarajan. Distributed controller synthesis for local specifications. In *ICALP '01*, volume 2076 of *LNCS*. Springer, 2001.
9. P. Madhusudan and P. S. Thiagarajan. Branching time controllers for discrete event systems. *Theor. Comput. Sci.*, 274(1-2):117–149, 2002.
10. P. Madhusudan and P. S. Thiagarajan. A decidable class of asynchronous distributed controllers. In *CONCUR '02*, volume 2421 of *LNCS*. Springer, 2002.
11. S. Mohalik and I. Walukiewicz. Distributed games. In *FSTTCS '03*, volume 2914 of *LNCS*, pages 338–351. Springer, 2003.
12. G. Peterson, J. Reif, and S. Azhar. Lower bounds for multiplayer noncooperative games of incomplete information. *Comput. Math. Appl.*, 41(7-8):957–992, 2001.
13. G. L. Peterson and J. H. Reif. Multiple-person alternation. In *20th Annual Symposium on Foundations of Computer Science (San Juan, Puerto Rico, 1979)*, pages 348–363. IEEE, New York, 1979.
14. A. Pnueli and R. Rosner. On the synthesis of an asynchronous reactive module. In *ICALP '89*, volume 372 of *LNCS*, pages 652–671. Springer, 1989.
15. A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthetize. In *31th IEEE Symp. FOCS*, pages 746–757, 1990.
16. P. Ramadge and W. Wonham. The control of discrete event systems. In *IEEE*, volume 77, pages 81–98, 1989.
17. W. Thomas. On logical definability of traces languages. In *workshop of ESPRIT BRA 3166, ASMICS*, pages 172–182, Kochel am See, 1990.
18. W. Zielonka. Asynchronous automata. In G. Rozenberg and V. Diekert, editors, *Book of Traces*, pages 175–217. World Scientific, Singapore, 1995.