Distributed games and distributed control for asynchronous 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. We introduce distributed games over *asynchronous* transition systems to model a distributed controller synthesis problem. A game involves two teams and is not turn-based: several players of both teams may simultaneously be enabled. We define distributed strategies based on the *causal* view that players have of the system. We reduce the problem of finding a winning distributed strategy with a given memory to finding a memoryless winning distributed strategy in a larger distributed game. We reduce the latter problem to finding a strategy in a classical 2-players game. This allows to transfer results from the sequential case to this distributed setting.
Keywords. Distributed game, distributed control, distributed strategy.

1 Introduction

The controller synthesis problem has been widely investigated by many authors for different system types (sequential, concurrent, timed, probabilistic) and different specification languages (linear or branching temporal logics for instance). The variant addressed here, called distributed controller synthesis problem, is the following. We are given a *distributed reactive system* executing a program in some environment, modeled by an *asynchronous* transition system [13] made up of several processes. It can perform local actions and synchronization actions involving several processes. Such an action first reads states of the participating processes and, depending on what is read, chooses a transition changing the states of the involved processes. Interpreting processes as memory locations, this suggests communication via shared memory, and explains the terminology adopted in the paper. However, one can as well simulate with these asynchronous systems other communication paradigms, such as point-to-point channels. Another advantage of this model is to handle actions of the environment just as actions of the reactive system. We are also given a *specification*, *i.e.*, a property expressing behaviors one wants to ensure for the system. The distributed controller synthesis problem is then to compute a distributed controller on the same process set (actions of the controller observe local states of some processes).

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

With this information, the controller has to enable or disable controllable actions of the system, so that the *overall* system behaves correctly according to the specification.

The problem is known to be decidable and to have an optimal solution in the sequential case [10], *i.e.*, when there is a single process. For distributed systems however, the situation is more involved. Several models have been considered so far (formulated in control or in game theory terminology). For synchronous processes communicating via buffers, the problem is undecidable [9] for LTL specifications except for very few communication architectures. Recent works [5,4] extend this result, *e.g.*, for local specifications (talking only of actions of single processes). The approach of [7] unifies [9,5,4]. In all settings, a major reason for undecidability is when the specification language makes it possible to express properties of an *observed linearization* of process actions, ignoring their possible concurrency. Another distributed model is studied in [6], and it is shown that the existence of specific controllers is decidable for specification languages making no distinction between linearizations of the same concurrent execution.

Systems used in this paper subsume those of [9,5,4,7] and [6]. In [6], global transitions of the system are obtained by synchronizing local transitions of processes, and transitions of the environment are local. Here in contrast, a transition of a synchronizing action also depends on the states of other involved processes, so that transition functions of actions are not necessarily a cartesian product of local transition functions. Further, environment moves can be defined globally. Systems of [7], in which the environment is global and transitions of the system are purely local (process communication flows through the environment) can also be modeled naturally in our framework. Another difference is that [6,7] use *local* memory controllers, based on the history of process local states (cf. Sec. 5). We use *causal* memory: a controller can remember information collected from other processes along the computation. The existence of a distributed controller in the settings of [6,7] implies the existence of a controller in our setting. As the converse does not hold, one cannot transfer immediately the undecidability results of [6,7] to our case.

Our primary goal is to model the distributed controller synthesis problem by *games*. Sequential 2-players games already provide a natural and widely used context to model sequential reactive systems [8,11,14,12]. Player 0 represents the system and player 1 represents the environment. The rules of the game describe the possible interactions between them, and the winning condition for player 0 expresses the specification that the system should meet. Thus, deciding whether player 0 has a winning strategy corresponds to deciding whether the system can be controlled to meet the specification, and computing a winning strategy for player 0 corresponds to solving the controller synthesis problem.

Distributed games proposed in this paper fit suitably to the model of asynchronous systems and supply a natural framework for studying the distributed controller problem. Two teams play one against the other. Players of team 0 may be viewed as controllable actions of a distributed system which cooperate in order to meet the specification, no matter how the environment (team 1) behaves. All players use a pool of shared variables to transmit information. The game is not turn-based: in each position of the game, several players of team 0 or team 1 may be simultaneously enabled. Thus, the game is asynchronous, contrary to the setting of [7] where at each stage players act synchronously. Assuming that dependencies between actions are fixed (*i.e.*, do not depend on the context), a play is then a Mazurkiewicz trace.

We next define the notion of distributed strategy for a team in such a game. Roughly speaking, a strategy is distributed if any move it predicts for a player only depends on the causal view of that player. In this context, there exist games in which neither team 0 nor team 1 have a winning distributed strategy.

We first show that, as in the sequential framework, one can transform a game G for which team 0 has a distributed strategy with memory μ into a game G^μ for which team 0 has a memoryless strategy. If G and the memory are finite, then so is G^μ. We further transform a distributed game G into a classical 2-players game $\widetilde{G}$, such that team 0 has a memoryless distributed strategy in G if and only if player 0 has a memoryless winning strategy in $\widetilde{G}$. This result is effective: $\widetilde{G}$ can be effectively constructed and from a winning strategy of $\widetilde{G}$ we can effectively construct a winning distributed strategy for G and vice versa. We then show that if the winning condition is a recognizable trace language, then one can decide whether team 0 has a memoryless distributed winning strategy, and compute it. The restriction to recognizable specifications is not artificial and makes it possible to express a relationship between the architecture of the game and the specification. Moreover, in practice, recognizable languages cover most interesting properties. As in [6,7], specifications depending on the order of independent actions lead to undecidability.

Finally, we explain how to simulate distributed games of [7] in our context. The goals of the two kinds of games are quite different. The aim of [7] is to unify different approaches and to find generic transformations on distributed games (*e.g.*, the reduction of the number of players) to get decidability results, while we first focused on a natural model which we then reduced to sequential games. Due to space constraints, proofs are omitted.

2 Preliminaries

In this section, we briefly recall definitions of pomsets and Mazurkiewicz traces. The reader is referred to [3,2] for details.

If $(V, \leqslant)$ is a poset and $S \subseteq V$, we let $\downarrow S = \{e \in V \mid \exists s \in S, e \leqslant s\}$. When $e \in V$ then we simply write $\downarrow e$ for $\downarrow\{e\}$ and we let $\Downarrow e = \downarrow e \setminus \{e\}$. The successor relation associated with the partial order $<$ is $\lessdot \; = \; < \setminus <^2$.

A *pomset* over an alphabet Σ is a tuple $(V, \leqslant, \ell)$ where $(V, \leqslant)$ is a poset, and $\ell : V \to \Sigma$ is a mapping called the *labeling*. Elements of V are called *events* or *vertices*. Two pomsets $t = (V, \leqslant, \ell)$, and $t' = (V', \leqslant', \ell')$ are *isomorphic*, written $t \sim t'$ if there exists a bijection $\varphi : V \to V'$ such that $\ell' \circ \varphi = \ell$ and for all $e, f \in V$ $e \leqslant f$ iff $\varphi(e) \leqslant' \varphi(f)$. If $\Sigma = \Sigma_1 \times \Sigma_2$ and $\ell(e) = (\ell_1(e), \ell_2(e))$, we write (ℓ_1, ℓ_2) (or even ℓ_1, ℓ_2) instead of ℓ. If $t = (V, \leqslant, \ell)$ is a pomset, we denote by $\max(t)$ (resp. by $\min(t)$) the set of maximal (resp. minimal) elements of t. The *alphabet* of t is $\mathrm{alph}(t) = \ell(V)$. We let $\mathrm{alphinf}(t) = \big\{a \in \mathrm{alph}(t) \mid \ell^{-1}(a) \text{ is infinite}\big\}$.

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*. We let $\mathrm{I} = \Sigma \times \Sigma \setminus \mathrm{D}$ be the independence relation. A *(Mazurkiewicz) trace* over (Σ, D) is an isomorphism class of a pomset $(V, \leqslant, \ell)$ such that, for all $e, f \in V$: (1) $\ell(e) \,\mathrm{D}\, \ell(f) \Rightarrow e \leqslant f$ or $f \leqslant e$, (2) $e \lessdot f \Rightarrow \ell(e) \,\mathrm{D}\, \ell(f)$ and (3) $\downarrow e$ is finite. Two traces t, t' are *independent* if $(\mathrm{alph}(t) \times \mathrm{alph}(t')) \cap \mathrm{D} = \emptyset$. We denote by $\mathbb{R}(\Sigma, \mathrm{D})$ (resp. by $\mathbb{M}(\Sigma, \mathrm{D})$) the set of traces (resp. of finite traces) over (Σ, D). It is well-known that $\mathbb{M}(\Sigma, \mathrm{D})$ is a monoid. The free monoid (resp. the free semigroup) over Σ is denoted by Σ^* (resp. by Σ^+).

A *prefix* of $t = (V, \leqslant, \ell)$ is a trace $(U, \leqslant, \ell)$, where $U \subseteq V$ satisfies $\downarrow U = U$. We write $s \leqslant t$ is s is a prefix of t. A *linearization* of t is a labeled total order $(V, \preccurlyeq, \ell)$ such that $e \leqslant f$ implies $e \preccurlyeq f$. For any $w \in \Sigma^*$, there exists a unique trace $[w]$ of which w is a linearization.

3 Distributed games

A distributed system made up of asynchronous processes interacting together and with the environment may be viewed as a single asynchronous model having controllable actions (the system's ones) and uncontrollable actions (the environment's ones). In the game setting, one views actions as players, which are split in two teams Σ_0 (actions of the system) and Σ_1 (actions of the environment). An execution of the system inside the environment corresponds then to a play, a property of the executions to a winning condition, and a distributed controller to a winning distributed strategy for team 0.

If X and I are sets and $J \subseteq I$, then for $x = (x_i)_{i \in I} \in X^I$ we let $x_J = (x_i)_{i \in J} \in X^J$. Given sets $(X_i)_{i \in I}$, and $J \subseteq I$, we let $X_J = \prod_{i \in J} X_i$.

An *architecture* is a tuple $(\Sigma, \mathcal{P}, R, W)$ such that Σ is a finite set of *actions* or *players*, $\mathcal{P}$ is a finite set of *processes*, $R : \Sigma \to 2^{\mathcal{P}}$ assigns to each $a \in \Sigma$ its read domain $R(a)$, $W : \Sigma \to 2^{\mathcal{P}}$ assigns to each $a \in \Sigma$ its write domain $W(a)$. We only consider architectures satisfying the following natural restriction, already considered in [13], and sufficient to get a dependence alphabet on actions.

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

We define the dependence relation over Σ as $\mathrm{D} = \{(a, b) \mid R(a) \cap W(b) \neq \emptyset\}$.

Let $(\Sigma, \mathcal{P}, R, W)$ be an architecture. A *distributed game* over $(\Sigma, \mathcal{P}, R, W)$ 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 Σ_0 and Σ_1 are the players of teams 0 and 1 respectively and we have $\Sigma = \Sigma_0 \uplus \Sigma_1$, $\forall i \in \mathcal{P}$, Q_i is the set of local states for process i, $\forall a \in \Sigma$, $T_a \subseteq Q_{R(a)} \times Q_{W(a)}$ gives the local moves of player a, $q^0 \in Q = \prod_{i \in \mathcal{P}} Q_i$ is the *starting position* of G, and $\mathcal{W}$ defines the winning condition of G.

The easiest way to define the semantics of the distributed game is via its sequential game graph whose set of positions is Q, the initial position is q^0 and there is an a-move from $p \in Q$ to $q \in Q$ (denoted $p \xrightarrow{a} q$) if $(p_{R(a)}, q_{W(a)}) \in T_a$

and $q_{\mathcal{P}\setminus W(a)} = p_{\mathcal{P}\setminus W(a)}$. A sequential play is a sequence $q^0 \xrightarrow{a_1} q^1 \xrightarrow{a_2} q^2 \cdots$. Note that in a position $p \in Q$, several players of team 0 and of team 1 may be simultaneously enabled, hence this sequential game graph does not correspond to a conventional (sequential) game in which each position is either a position of player (team) 0 or a position of player (team) 1.

We consider a new symbol $\perp \notin \Sigma$ with $R(\perp) = W(\perp) = \mathcal{P}$ and the alphabet $\Sigma' = \{(a,p) \mid a \in \Sigma \text{ and } p \in Q_{W(a)}\} \cup \{(\perp, q^0)\}$ with the dependence relation $\mathrm{D}' = \{((a,p),(b,q)) \mid R(a) \cap W(b) \neq \emptyset\}$. The winning condition of the game is a set of words $\mathcal{W} \subseteq \Sigma'^{\infty}$ which is closed under the usual trace equivalence (see [2,3]). With the sequential play $\pi = q^0 \xrightarrow{a_1} q^1 \xrightarrow{a_2} q^2 \cdots$ of G we associate the word $w = (\perp, q^0)(a_1, q^1_{W(a_1)})(a_2, q^2_{W(a_2)}) \cdots$ over Σ'. Note that the word w faithfully encodes the sequential play π and team 0 wins the play π if $w \in \mathcal{W}$.

A better semantics of these distributed games is to view a play directly as a *rooted* trace over the alphabet Σ'. A finite or infinite trace $t = (V, \leqslant, (\ell, \sigma)) \in \mathbb{R}(\Sigma', \mathrm{D}')$ is *rooted* if $\ell^{-1}(\perp) = \{x_\perp\}$ is a singleton and $x_\perp \leqslant y$ for all $y \in V$. If $s = (U, \leqslant, (\ell, \sigma))$ is a nonempty prefix of t then $\bar{\sigma}(s) = (\bar{\sigma}(s)_i)_{i \in \mathcal{P}} \in Q$ is defined by $\bar{\sigma}(s)_i = \sigma(y)_i$ where y is the maximal vertex in $\{x \in U \mid i \in W(\ell(x))\}$.

A distributed play of G is a finite or infinite rooted trace $t = (V, \leqslant, (\ell, \sigma)) \in \mathbb{R}(\Sigma', \mathrm{D}')$ such that for each $a \in \Sigma$ and $x \in \ell^{-1}(a)$, we have $(\bar{\sigma}(\Downarrow x)_{R(a)}, \sigma(x)) \in T_a$. The winning condition $\mathcal{W}$ is now a subset of $\mathbb{R}(\Sigma', \mathrm{D}')$ and team 0 wins the distributed play t if $t \in \mathcal{W}$. The two definitions above are indeed equivalent but the second one is better suited to distributed games and allows a natural definition of a distributed strategy. In sequential games, one often considers infinite plays only. We also consider finite plays because it can be more convenient.

Let $t = (V, \leqslant, \ell, \sigma) \in \mathbb{R}(\Sigma, \mathrm{D})$ be a rooted trace and let $J \subseteq \mathcal{P}$. The trace $\partial_J t$ is the prefix of t defined by the set of vertices $U = \downarrow\{x \in V \mid W(\ell(x)) \cap J \neq \emptyset\}$.

An asynchronous mapping [1] is a function $\mu : \mathbb{M}(\Sigma, \mathrm{D}) \to M$ such that $\mu(\partial_{A \cup B} t)$ only depends on $\mu(\partial_A t)$ and $\mu(\partial_B t)$, and $\mu(\partial_{R(a)} t.a)$ only depends on $\mu(\partial_{R(a)} t)$ and a. Asynchronous mappings can be computed by deterministic asynchronous transition systems. A *distributed memory* on a game G is an asynchronous mapping $\mu : \mathbb{M}(\Sigma', \mathrm{D}') \to M$. It will be used by players of team 0 as an abstraction (computed in M) of their causal view of the play. A *distributed strategy with memory* μ for team 0 (μ-DS) is a pair (f, μ) where f is a partial function $f : \bigcup_{a \in \Sigma_0} Q_{R(a)} \times M^{R(a)} \times \{a\} \to Q_{W(a)}$ such that if $f(p, m, a) = q$, then $(p, q) \in T_a$. Intuitively, if $f(p, m, a) = q$, then the strategy f dictates an a-move to $q \in Q_{W(a)}$ when the memory of the play that a can observe using μ is $m \in M^{R(a)}$. If $f(p, m, a)$ is undefined, the a-move is disabled by the strategy. Note that several players of team 0 may be simultaneously enabled by f during a play. In the sequel we write f instead of (f, μ), μ being understood.

Let f be a μ-DS. Let $\bar{\mu}(t) = (\mu(\partial_i(t)))_{i \in \mathcal{P}}$. A distributed play $t = (V, \leqslant, \ell, \sigma) \in \mathbb{R}(\Sigma', \mathrm{D}')$ is an *f-play* if

$$\forall x \in V,\ \sigma(x) = f(\bar{\sigma}(\Downarrow x)_{R(a)}, \bar{\mu}(\Downarrow x)_{R(a)}, a)$$

The play t is *f-maximal* if $f(\bar{\sigma}(\partial_{R(a)} t)_{R(a)}, \bar{\mu}(\partial_{R(a)} t)_{R(a)}, a)$ is undefined for all $a \in \Sigma_0$ such that $\partial_{R(a)} t$ is finite. The *maximality* condition is natural: if the DS of team 0 dictates some a-moves at some f-play t, then the f-play t is not over and we do not have to decide whether it is winning or not for team 0. Note that 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 enabled by f. Observe 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.

A distributed game is not necessarily determined in the sense that it is possible that neither team 0 nor team 1 have a WDS, even with perfect memory. For instance, consider $G = (\Sigma_0, \Sigma_1, (Q_i)_{i\in\mathcal{P}}, (T_a)_{a\in\Sigma}, q^0, \mathcal{W})$ with $\Sigma_0 = \{a\}$, $\Sigma_1 = \{b\}$, $\mathcal{P} = \{1,2\}$, $R(a) = W(a) = \{1\}$, $R(b) = W(b) = \{2\}$, $Q_1 = Q_2 = \{1\}$, $T_a = Q_1^2$, $T_b = Q_2^2$, $q^0 = (1,1)$, and $\mathcal{W} = \mathbb{M}(\Sigma', \mathrm{D}') \cup \{(\bot, q^0)(a,1)^\omega(b,1)^\omega\}$. Assume that team 0 has a DS. If $f((\bot, q^0)(a,1)^n, a) \neq \emptyset$ for all $n \geqslant 0$, then team 0 loses if team 1 does not play at all, yielding the play $(\bot, q^0)(a,1)^\omega$. Conversely, if there exists $n \geqslant 0$ such that $f((\bot, q^0)(a,1)^n, a) = \emptyset$, then team 0 loses if team 1 makes infinitely many moves. Symmetrically, team 1 does not have a WDS.

Actually, this non-determinacy is not a problem. For the distributed control problem, we are looking for a WDS allowing controllable events (team 0) to enforce good behaviors but we are not interested in a winning *distributed* strategy for the uncontrollable events: uncontrollable events are played by an environment, and there is no reason to consider only distributed environments.

A *memoryless distributed strategy (MDS)* is a μ-DS with $|\mu(\mathbb{M}(\Sigma', \mathrm{D}'))| = 1$, that is, the memory does not record any information. In this case, we write $f(p,a)$ instead of $f(p,m,a)$. A *perfect-memory distributed strategy* is a μ-DS with $\mu(t) = t$. It provides for a move to x with $\ell(x) = a$ the full causal view $\bar{\mu}(\Downarrow x)_{R(a)} = (\partial_i \Downarrow x)_{i\in R(a)}$. Since $\bar{\sigma}(\Downarrow x)_{R(a)}$ can be computed from $(\partial_i \Downarrow x)_{i\in R(a)}$, one can drop the state component in f and write $f(m,a)$ instead of $f(p,m,a)$. As in the sequential case, one can embed a given memory into the game.

Proposition 1. *Let G be a distributed game and let μ be a distributed memory on G. One can construct a distributed game G^μ such that there exists a μ-WDS for G iff there exists a WMDS in G^μ. Moreover, if G is finite and μ is realized by a finite asynchronous automaton, then G^μ is finite.*

4 Global game

In order to use known results of game theory, we want to define a classical two-players global game $\widetilde{G} = (Z, T)$ such that team 0 has a WMDS in the distributed game G iff player 0 has a winning memoryless strategy in the global game $\widetilde{G}$.

The positions of the global game are $Z = Z_0 \cup Z_1$ where $Z_0 = Q \times \Sigma_0$ are the positions of player 0 and $Z_1 = Q \times (\Sigma_1 \cup \{0,1,2\})$ are the positions of player 1. The initial position is $(q^0, 0) \in Z_1$. In a position (q,a), the first component describes the current global state of the play and the second component is used both to determine whose turn it is and which action should be executed. The set $T \subseteq (Z_0 \times Z_1) \cup (Z_1 \times Z)$ of moves is defined as follows:

- $(p,b) \to (p,a)$ with $b \in \{0,1,2\}$ and $a \in \Sigma$. Player 1 decides that the next move should be an *a-move*. In this global game, player 1 is in charge of deciding which actions are used and in which order. This allows him to investigate all possible linearizations of distributed plays.

– $(p, a) \to (q, 1)$ with $a \in \Sigma$, $(p_{R(a)}, q_{W(a)}) \in T_a$ and $q_{\mathcal{P}\setminus W(a)} = p_{\mathcal{P}\setminus W(a)}$. This *a-move* is executed by player 0 or player 1 depending on whether $a \in \Sigma_0$ or $a \in \Sigma_1$.
– $(p, a) \to (p, 2)$ with $a \in \Sigma_0$. Player 0 *refuses* to make an a-move.
– $(q, b) \to (q^0, 0)$ with $b \in \{0, 1, 2\}$. These *reset-moves* are used by player 1 to show that player 0 is not following a *distributed* strategy.

Note that player 1 may perform several consecutive moves.

A global play is a finite or infinite sequence $z = z_0z_1z_2 \cdots \in Z^\infty$ starting from the initial position $z_0 = (q^0, 0)$ and such that $z_n \to z_{n+1}$ is a move for all $n \geqslant 0$. Let $z = z_0z_1z_2 \cdots \in Z^\infty$ be a global play and let $z_n = (q^n, a_n) \in Z$ for $n \geqslant 0$. We define by induction the sequence $(t_n)_{n\geqslant 0} \in \mathbb{M}(\Sigma', \mathrm{D}')^{\mathbb{N}}$ associated with z. If $z_n = (q^0, 0)$ then $t_n = (\perp, q^0)$. If $a_{n+1} \in \Sigma \cup \{2\}$ then $t_{n+1} = t_n$. Finally, if $a_n = a \in \Sigma$ and $a_{n+1} = 1$ then $t_{n+1} = t_n \cdot (a, q^{n+1}_{W(a)})$. We prove by induction that t_n is a distributed play and $\bar{\sigma}(t_n) = q^n$ for all $n \geqslant 0$. The only non trivial case is when $a_n = a \in \Sigma$ and $a_{n+1} = 1$. By induction, t_n is a distributed play and $\bar{\sigma}(t_n) = q^n$. We have $(q^n_{R(a)}, q^{n+1}_{W(a)}) \in T_a$ and $q^n_{R(a)} = \bar{\sigma}(\partial_{R(a)}t_n)_{R(a)}$. Therefore, t_{n+1} is a distributed play and using $q^{n+1}_{\mathcal{P}\setminus W(a)} = q^n_{\mathcal{P}\setminus W(a)}$ we get $\bar{\sigma}(t_{n+1}) = q^{n+1}$.

The global play z is *consistent* if for all $j, k \geqslant 0$ with $a_j = a_k = a \in \Sigma_0$ and $q^j_{R(a)} = q^k_{R(a)}$ we have $a_{j+1} = a_{k+1}$ and $q^{j+1}_{W(a)} = q^{k+1}_{W(a)}$. The global play z is *fair* if $\{n \geqslant 0 \mid a_n = 0\}$ is finite and for all $a \in \Sigma_0$, $\{n \geqslant 0 \mid a_n = a\}$ is infinite. If z is both consistent and fair then we let $N(z) = \max\{n \mid a_n = 0\}$. The sequence $(t_n)_{n\geqslant N(z)}$ is increasing and admits a least upper bound $t(z)$ which is a distributed play of G.

The winning condition $\widetilde{\mathcal{W}}$ of $\widetilde{G}$ only involves infinite plays $z \in Z^\omega$. If z is not consistent then player 0 loses the game since this reveals that he does not mimic a memoryless *distributed* strategy. If z is not fair then player 1 loses the game. Finally, if z is both consistent and fair then player 0 wins the game iff $t(z) \in \mathcal{W}$.

A (global) *strategy* (S) for player 0 in $\widetilde{G}$ is a mapping $g : Z^*Z_0 \to Z_1$ such that $g(z(p, a)) = (q, b)$ implies $(p, a) \to (q, b)$. A global play $z = z_0z_1z_2 \cdots \in Z^\infty$ is *played according to g* (g-play) if each move of player 0 is done according to g: $z_k \in Z_0$ implies $z_{k+1} = g(z_0 \cdots z_k)$. If player 0 wins all infinite g-plays then g is a winning strategy (WS) for player 0. A strategy g is *memoryless* if for all $x, x' \in Z^*$ and $y \in Z_0$, we have $g(xy) = g(x'y)$. We write MS and WMS for *memoryless strategy* and *winning memoryless strategy*.

We can now state the main result of this section.

Theorem 1. *The following conditions are equivalent for a distributed game G:*

1. *There exists a WMDS for team 0 in the distributed game G.*
2. *There exists a WMS for player 0 in the global game $\widetilde{G}$.*
3. *There exists a WS for player 0 in the global game $\widetilde{G}$.*

The following proposition gives the construction used for the implication $(1 \Rightarrow 2)$.

Proposition 2. *Let f be a deterministic WMDS for team 0 in G. For $(p,a) \in Z_0$, we define $g((p,a)) = (p,2)$ if $f(p_{R(a)},a) = \emptyset$ and $g((p,a)) = (q,1)$ with $q_{\mathcal{P}\setminus W(a)} = p_{\mathcal{P}\setminus W(a)}$ and $f(p_{R(a)},a) = \{q_{W(a)}\}$ otherwise. Then, g is a WMS for player 0 in the global game $\widetilde{G}$.*

To prove the implication $(2 \Rightarrow 1)$ of Theorem 1, we exploit reset-moves.

Lemma 1. *Let g be a WMS of player 0 in the global game $\widetilde{G}$. Let $(p^1,a) \in Z_0$ and $(p^2,a) \in Z_0$ be accessible in g-plays and such that $p^1_{R(a)} = p^2_{R(a)}$. Then, $g(p^1,a) = (p^1,2)$ iff $g(p^2,a) = (p^2,2)$ and if $g(p^1,a) = (q^1,1)$ and $g(p^2,a) = (q^2,1)$ then $q^1_{W(a)} = q^2_{W(a)}$.*

Using this lemma, we can now transform a WMS in $\widetilde{G}$ into a WMDS in G.

Proposition 3. *Let g be a WMS of player 0 in the global game $\widetilde{G}$. For $(p,a) \in Z_0$ accessible by a g-play, we define $f(p_{R(a)},a) = \emptyset$ if $g(p,a) = (p,2)$ and $f(p,a) = \{q_{W(a)}\}$ if $g(p,a) = (q,1)$. Then, f is a WMDS of team 0 in the distributed game G.*

Even if $\mathcal{W}$ is rational ($\mathcal{W} = [L]$, where $L \in \mathrm{Rat}(\Sigma^*)$), determining if team 0 has a W(M)DS is undecidable. Indeed, on $\mathbb{M}(\Sigma,\mathrm{D}) = A^* \times B^*$, determining if a rational trace language $\mathcal{L}$ is $[\Sigma^*]$ is undecidable [2]. From such a language $\mathcal{L}$, we construct a 2-processes game in which team 0 has a WMDS iff $\mathcal{L} = [\Sigma^*]$: $\Sigma_0 = \emptyset$, $\Sigma_1 = A \uplus B$, $R(a) = W(a) = \{1\}$ for $a \in A$ and $R(b) = W(b) = \{2\}$ for $b \in B$. Finally, $|Q| = 1$ (so that we identify Σ' and Σ), and $\mathcal{W} = \mathcal{L} \cup (\mathbb{R}(\Sigma,\mathrm{D}) \setminus \mathbb{M}(\Sigma,\mathrm{D}))$. Players of team 1 nondeterministically choose a move in some finite local game, so that any possible trace is a play. Now, team 0 has a WMDS iff he has a WDS iff team 1 cannot generate a finite trace outside $\mathcal{L}$, that is, iff $\mathcal{L} = [\Sigma^*]$.

We now explain how to use Theorem 1 to decide if team 0 has a WDS. Denote by $\mathrm{Lin}(t)$ the set of all linearizations of $t \in \mathbb{R}(\Sigma,\mathrm{D})$. Properties considered in practice are recognizable (*i.e.*, $\mathrm{Lin}(\mathcal{W})$ is rational), and as noted in [6], there are many temporal logics expressing only recognizable specifications. To determine whether team 0 has a WMDS in G with $\mathcal{W}$ recognizable, we could enumerate all memoryless distributed strategies for player 0, and check whether one is winning. This amounts to testing an inclusion between recognizable trace languages. Theorem 1 provides a better algorithm. The principle is to build the global game $\widetilde{G}$, to transform it into a parity game and to apply known algorithms.

Let $\mathcal{A}$ be a parity automaton accepting $\mathrm{Lin}(\mathcal{W})$. The winning condition $\widetilde{\mathcal{W}}$ for player 0 on the global game $\widetilde{G}$ can be defined by $\widetilde{\mathcal{W}} = \widetilde{\mathcal{W}}_c \cap (\mathrm{Lin}(\mathcal{W}) \cup \widetilde{\mathcal{W}}_{\mathrm{nf}})$, where $\widetilde{\mathcal{W}}_c = \{z \in Z^\omega \mid z \text{ is consistent}\}$, and $\widetilde{\mathcal{W}}_{\mathrm{nf}} = \{z \in Z^\omega \mid z \text{ is not fair}\}$. We describe informally how to construct a parity automaton accepting $\widetilde{\mathcal{W}}$. One can build a Büchi automaton accepting consistent plays in Z^ω: it records all transitions $(p_{R(a)}, q_{W(a)})$ performed by player 0, as well as the refused transitions. It falls into the unique rejecting state as soon as an inconsistent move is detected. One can also build a parity automaton checking that a play, supposed consistent, is not fair, by checking that there is an infinite number of reset moves (a Büchi condition) or that, for some $a \in \Sigma_0$, there is only a finite number of states of the form (p, a) (co-Büchi conditions). From the parity automaton $\mathcal{A}$ and from these automata, it should now be clear how to build a parity automaton for $\widetilde{\mathcal{W}}$.

Observe that using Proposition 1, one can also determine whether team 0 has a μ-WDS for a given finite distributed memory μ.

5 Related approaches

A distributed game $G = \langle P, E, \mathrm{Tr}, \mathrm{Acc}, q^0 \rangle$ as in [7] is built from n local games $G_1, \ldots, G_n$, where $G_i = \langle P_i, E_i, \mathrm{Tr}_i, q_i^0 \rangle$ with $\mathrm{Tr}_i \subseteq (P_i \times E_i)$. Positions of the environment are $E = \prod_i E_i$. The position set of the players is $P = \prod_i (P_i \cup E_i) \setminus E$. Transitions of the players are defined with a cartesian product: $\mathrm{Tr}_p = (\prod_i (\mathrm{Tr}_i \cup \Delta_i)) \cap (P \times E)$ where $\Delta_i = \{(x_i, x_i) \mid x_i \in E_i\}$ is the diagonal. Transitions of the environment are simply given by a subset Tr_e of $E \times P$, and $\mathrm{Tr} = \mathrm{Tr}_e \uplus \mathrm{Tr}_p$. A play of G starts in position $q^0 \in E$, and moves from the environment and from the players alternate. Hence, any infinite play is in $(E \cdot P)^\omega$, and the winning condition is a subset $\mathrm{Acc} \subseteq (E \cdot P)^\omega$.

There is a natural translation from these games to our setting. With G, we associate the distributed game $\bar{G} = (\Sigma_0, \Sigma_1, (T_a)_{a \in \Sigma}, q^0, \mathcal{W})$ as follows. The set of processes is $\mathcal{P} = \{1, \ldots, n\}$ and the local states are $Q_i = P_i \cup E_i$ for $i \in \mathcal{P}$. Team 0 is defined by $\Sigma_0 = \{1, \ldots, n\}$ with $R(i) = W(i) = \{i\}$ for all $i \in \Sigma_0$. The transitions for player i are simply $T_i = \mathrm{Tr}_i$. Team 1 consists of a single player e (the environment) with $R(e) = W(e) = \mathcal{P}$. Its transitions are $T_e = \mathrm{Tr} \cap (E \times P)$.

To define the winning condition $\mathcal{W}$, we associate with each infinite play $w = e^0 x^1 e^1 \cdots \in (E \cdot P)^\omega$ of G with $e^0 = q^0$ a distributed play $\mathrm{trace}(w) \in \mathbb{R}(\Sigma', \mathrm{D}')$ as the least upper bound of $(t_i)_{i \geqslant 0}$ where the increasing sequence $(t_n)_{n \geqslant 0}$ is defined inductively by $t_0 = (\perp, q^0)$ and $t_{n+1} = t_n \cdot (e, x^{n+1}) \cdot \prod_{i \mid x_i^{n+1} \in P_i} (i, e_i^{n+1})$. Finally, the winning condition is $\mathcal{W} = \mathrm{trace}(\mathrm{Acc}) \cup \{t \in \mathbb{M}(\Sigma', \mathrm{D}') \mid \bar{\sigma}(t) \in E\}$.

The distributed games of [7] are thus a special case of our games in which all players of team 0 are completely local and the environment consists of a single global player. Note that, in the game G, information between players can only flow through environment moves and the environment can decide which information is exchanged between players.

However, the crucial difference between games of [7] and the ones presented here concerns the definition of strategies. In [7], a strategy is a tuple of mappings $f_i : (E_i P_i)^+ \to E_i$. Hence a move of player i only depends on its *local* view consisting of the history on its process only. In our setting, the strategy of player i which is the mapping $f(-, i)$ is based on its causal view $\partial_{R(i)} t$. Since actions of the environment are global, the causal view is almost the complete global view of the game. It is therefore clear that if there exists a WS for the players in G, then there exists a WDS for team 0 in $\bar{G}$. The converse is false: it is easy to find a game G not determined while there is a winning strategy for team 0 in $\bar{G}$. Yet, one can prove that if a game G of [7] is determined, then there is a WDS for team 0 in $\bar{G}$ iff players have a WS in G.

If we want to get an equivalence, we have to restrict the memory used by our strategies to the local view, that is, to change the notion of memory used by the strategies. The i-projection of $t \in \mathbb{M}(\Sigma, \mathrm{D}')$ is $\Pi_i(t)$ where Π_i is the morphism from $\mathbb{M}(\Sigma', \mathrm{D}')$ to Q_i^* defined by $\Pi_i(x,q) = q_i$ if $x = e$ or $x = i$ and $\Pi_i(x,q) = \varepsilon$ otherwise. Since player i is only aware of move he takes part in, we have to abstract away from unobservable stuttering of the environment. For this we use the congruence on Q_i^* generated by $p_i^3 = p_i$ for $p_i \in E_i$ and we write $w_\equiv$ for the equivalence class of $w \in Q_i^*$. We say that a distributed strategy f is local if for all $i \in \Sigma_0$, $f(t,i)$ depends only on i and $\Pi_i(t)_\equiv$.

Proposition 4. *The players have a WS in G iff team 0 has a local WDS in* $\bar{G}$*.*

In the distributed control problem presented in [6], the environment performs local moves and the transitions of a controllable action is defined by a cartesian product of local transition functions. It is therefore straightforward to translate as above these games to our framework. Since local strategies are used in [6], Propositions 4 also holds.

Acknowledgment We thank Madhavan Mukund for fruitful discussions.

References

1. R. Cori, Y. Métivier, and W. Zielonka. Asynchronous mappings and asynchronous cellular automata. *Inform. and Comput.*, 106:159–202, 1993.
2. V. Diekert and Y. Métivier. *Handbook of Formal languages*, volume 3, chapter Partial Commutations and Traces, pages 457–533. Springer, 1997.
3. V. Diekert and G. Rozenberg, editors. *The Book of Traces.* World Scientific, 1995.
4. O. Kupferman and M. Y. Vardi. Synthesizing distributed systems. In *Proc. of the LICS '01.* Computer Society Press, 2001.
5. P. Madhusudan and P. S. Thiagarajan. Distributed controller synthesis for local specifications. In *Proc. of the ICALP '01*, volume 2076 of *Lect. Notes Comp. Sci.*, pages 396–407. Springer, 2001.
6. P. Madhusudan and P. S. Thiagarajan. A decidable class of asynchronous distributed controllers. In *Proc. of the CONCUR '02*, volume 2421 of *Lect. Notes Comp. Sci.*, pages 145–160. Springer, 2002.
7. S. Mohalik and I. Walukiewicz. Distributed games. In *FSTTCS '03*, Lect. Notes Comp. Sci. Springer, 2003.
8. J. Pin and D. Perrin. *Infinite words. Automata, Semigroups, Logic and Games.* Elsevier, To appear.
9. A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthetize. In *Proc. of the 31th IEEE Symp. FOCS*, pages 746–757, 1990.
10. P. Ramadge and W. Wonham. The control of discrete event systems. In *Proceedings of the IEEE*, volume 77, pages 81–98, 1989.
11. W. Thomas. Infinite games and verification. In *Proc. of CAV'02*, volume 2404 of *Lect. Notes Comp. Sci.*, pages 58–64. Springer, 2002.
12. J. Vöge and M. Jurdziński. A discrete strategy improvement algorithm for solving parity games. In *Proc. of CAV '00*, volume 1855 of *Lect. Notes Comp. Sci.*, pages 202–215. Springer, 2000.
13. W. Zielonka. Asynchronous automata. In G. Rozenberg and V. Diekert, editors, *Book of Traces*, pages 175–217. World Scientific, Singapore, 1995.
14. W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1–2):135–183, 1998.