Distributed Games*

Swarup Mohalik[1] and Igor Walukiewicz[2]

[1] Hewlett-Packard, India Software Operations, Bangalore, India, **
[2] LaBRI, CNRS, 351, cours de la Libration, 33405 Talence Cedex, France

Abstract. We propose a notion of distributed games as a framework to formalize and solve distributed synthesis problems. In general the problem of solving distributed games is undecidable. We give two theorems allowing to simplify, and possibly to solve, some distributed games. We show how several approaches to distributed synthesis found in the literature can be formalized and solved in the proposed setting.

1 Introduction

Consider a *system* consisting of a *process*, an *environment* and possible ways of *interaction* between them. The *synthesis problem* is stated as follows: given a specification S, find a finite state program P for the process such that the overall behaviour of the system satisfies S, no matter how the environment behaves.

In a *distributed system*, in general, one can have multiple processes. The system specifies possible interactions between the processes and the environment and also the interactions among the processes themselves. The synthesis problem here is to find a program for each of the processes such that the overall behaviour of the system satisfies a given specification. We call this *distributed synthesis problem (DSP)*.

In this paper we study DSP in a setting where there is a fixed set of processes with no interaction among themselves; they interact only with the environment. Thus, any communication among processes in the system must be channeled through the environment. This is the typical scenario, for example, in any communication network where processes are peer protocols and the environment is the stack of lower layers (including the communication medium) below them. Typical metaphors of communication and synchronization like channels, rendezvous, handshakes can be easily presented in our model.

Earlier approaches. The distributed synthesis problem has been considered by Pnueli and Rosner in the setting of an architecture with fixed channels of communication among processes [29]. They have shown that distributed synthesis is undecidable for most classes of architectures. They have obtained decidability

* This research has been partially supported by the European Community Research Training Network "Games and Automata for Synthesis and Validation" (GAMES)
** The work was carried out at LaBRI, France when the author was a post-doctoral fellow with leave from Sasken Communication Technologies Ltd., Bangalore, India.

for a special class of hierarchical architectures called *pipelines*. It must be noted that the basic undecidability and lower bounds (in case of decidability) follow from much earlier results on the multi-player games of Peterson and Reif [27]. After the work of Pnueli and Rosner, the decidability results have been extended to branching time specifications over two-way pipelines and one-way rings [13]. These are essentially the only architectures for which the problem is decidable. Madhusudan and Thiagarajan [17] considered local specifications, i.e., a conjunction of specifications for each of the processes. For such specifications the class of decidable architectures is slightly larger and includes doubly-flanked pipelines.

The other approach to distributed synthesis, initiated roughly at the same time as the work of Pnueli and Rosner, comes from control theory of discrete event systems [30, 15, 32]. The system is given as a plant (deterministic transition system) and the distributed synthesis problem is to synthesize a number of controllers, each being able to observe and control only a specific subset of actions of the plant. While the original problem refers only to safety properties, an extension to the μ-calculus specifications has also been considered [4]. Except for some special cases, the problem turns out to be undecidable ([4, 34]). It is one of the important goals of the area of decentralized control synthesis to identify conditions on a plant and a specification such that DSP is decidable.

A different approach was suggested in [18]. The authors consider a setting were processes communicate via handshaking, i.e., common actions. This setting can easily encode undecidable architectures from Pnueli and Rosner setting so the synthesis problem, even for local specifications, is undecidable. To get decidability results the authors propose to restrict the class of allowed controllers.

Our approach. Game theory provides an approach to solving the (centralized) synthesis problem. An interaction of a process with its environment can be viewed as a game between two players [1, 28, 33]. Then the synthesis problem reduces to finding a finite-state winning strategy for the process. The winning strategy can then be implemented as the required program. This approach does not extend to DSP because there we have more than two parties.

In this paper, we suggest an approach to DSP by directly encoding the problem game-theoretically. We extend the notion of games to n players playing a game against a single hostile environment. We call this model *distributed games*. In this model, there are no explicit means of interaction among processes. Any such interaction must take place through the environment. Moreover, each player has only a local view of the global state of the system. Hence, a *local strategy* for a player is a function of its local history (of player's own states and the partial view of the environment's states). A *distributed strategy* is a collection of local strategies; one for each of the players. The environment in distributed games, on the other hand, has access to the global history. Any play in a distributed game consists of alternating sequence of moves of (some of) the players and of the environment.

Distributed synthesis in this model amounts to finding a distributed winning strategy. This means finding a collection of local strategies that can win against the global environment. A side effect of the requirement that the players need to win together is that they need to implicitly communicate when they make their moves. The card game of bridge is a good example of the kind of implicit communication we have in mind. When $n = 1$, distributed games reduce to the usual two-party games.

The main technical contribution of the paper are two theorems allowing simplification of distributed games. In general it is not decidable to check whether there is a distributed winning strategy in a finite distributed game. The simplification theorems allow to reduce the number of players and to reduce nondeterminism in the game. In some cases, by repetitive application of these theorems we can simplify the game to one with only one player against the environment (where the existence of a winning strategy is decidable). The other possibility is that after simplification we get a game where environment has no choice of moves. We show that in this case the existence of a distributed strategy is also decidable. This technique is enough to solve all decidable cases of distributed control problems mentioned above.

Related works. Readers may find the model of distributed games very close to the models of distributed systems in [25, 11]. The closeness is not accidental: the model was partly motivated by these and the later works of Halpern et al. [12, 10] which explore the issue of knowledge in distributed systems.

Distributed multi-player games have been studied extensively in classical game theory, both in the settings of cooperation and non-cooperation among the players [23, 21]. There have been attempts to model and logically reason about cooperation as in [24, 26]. Distributed games can be seen as a special type of concurrent processes – the models of Alternating Temporal Logic – with incomplete information [2], and the distributed systems model of Bradfield [7] (which generalizes ATL and integrates incomplete information). We consider our proposal as being something between these models and concrete synthesis problems, like the one for pipeline architectures. Our model is less general but this permits us to obtain stronger results that allow to solve concrete synthesis problems.

Amroszkiewicz and Penczek [3] study a class of games, also called distributed games, but the framework, questions and approaches are closer to classical game theory and different from ours.

Organization of the paper We start with a definition of games and distributed games. We give some simple properties of our model. In Sections 4 and 5 we formulate the two main theorems allowing to simplify distributed games. In Section 6 we show how to use these theorems to solve the synthesis problem for pipelines. In the full version of the paper [20], we consider other distributed synthesis problems and provide all the proofs.

2 Games

A *game* G is a tuple $\langle P, E, T \subseteq V \times V, Acc \subseteq V^\omega \rangle$ where $\langle P, E, T \rangle$ is a graph with the vertices $V = P \cup E$ and $Acc \subseteq V^\omega$ is a set defining the winning condition.

We say that a vertex x' is a *successor* of a vertex x if $T(x, x')$ holds. We call P the set of *player vertices* and E the set of *environment vertices*.

A *play* between player and environment from some vertex $v \in V$ proceeds as follows: if $v \in P$ then player makes a choice of a successor, otherwise environment chooses a successor; from this successor the same rule applies and the play goes on forever unless one of the parties cannot make a move. If a player cannot make a move he loses; similarly for the environment. The result of an infinite play is an infinite path $v_0 v_1 v_2 \ldots$ This *path is winning* for player if the sequence belongs to Acc. Otherwise environment is the winner.

A *strategy* σ for player is a function assigning to every sequence of vertices $\boldsymbol{v}$ ending in a vertex v from P a vertex $\sigma(\boldsymbol{v})$ which is a successor of v. The strategy is *memoryless* iff $\sigma(\boldsymbol{v}) = \sigma(\boldsymbol{w})$ whenever $\boldsymbol{v}$ and $\boldsymbol{w}$ end in the same vertex.

A *play respecting* σ is a sequence $v_0 v_1 \ldots$ such that $v_{i+1} = \sigma(v_i)$ for all i with $v_i \in P$. The *strategy* σ *is winning* from a vertex v iff all the plays starting in v and respecting σ are winning. A *vertex is winning* if there exists a strategy winning from it. The strategies for the environment are defined similarly.

In this paper all acceptance conditions $Acc \subseteq V^\omega$ will be regular: that is, there will be a colouring $\lambda : V \to Colours$ of the set of vertices with a finite set of colours and a regular language $L \subseteq Colours^\omega$ that define the accepting sequences by: $Acc = \{v_0 v_1 \ldots \in V^\omega : \lambda(v_0)\lambda(v_1) \ldots \in L\}$

An important type of regular winning condition is a *parity condition*. It is a condition determined by a function $\Omega : V \to \{0, \ldots, d\}$ in the following way:

$$Acc = \{v_0 v_1 \ldots \in V^\omega : \liminf_{i \to \infty} \Omega(v_i) \text{ is even}\}$$

Hence, in this case, the colours are natural numbers and we require that the smallest among those appearing infinitely often is even. The main results about games that we need are summarized in the following theorem

Theorem 1 ([19, 9, 22]). *Every game with regular winning conditions is determined, i.e., every vertex is winning for the player or for the environment. In a parity game a player has a memoryless winning strategy from each of his winning vertices. It is decidable to check if a given vertex of a finite game with a regular winning condition is winning for the player.*

3 Distributed Games

A *local game* is any game $G = \langle P, E, T\rangle$ as above but without a winning condition and with the restriction that it is bipartite, i.e., a successor of a player move is always an environment move and vice versa.

Let $G_i = \langle P_i, E_i, T_i\rangle$, for $i = 1, \ldots, n$, be local games. A *distributed game* constructed from $G_1, \ldots, G_n$ is $\mathcal{G} = \langle P, E, T, Acc \subseteq (E \cup P)^\omega\rangle$ where:

1. $E = E_1 \times \cdots \times E_n$.
2. $P = (P_1 \cup E_1) \times \ldots (P_n \cup E_n) \setminus E$.
3. From a player's position, we have $(x_1, \ldots, x_n) \to (x'_1, \ldots, x'_n) \in T$ if and only if $x_i \to x'_i \in T_i$ for all $x_i \in P_i$ and $x_i = x'_i$ for all $x_i \in E_i$.

4. From environment's position, if we have $(x_1, \ldots, x_n) \to (x'_1, \ldots, x'_n) \in T$ then for every x_i, either $x_i = x'_i$ or $x'_i \in P_i$ and moreover $(x_1, \ldots, x_n) \neq (x'_1, \ldots, x'_n)$
5. Acc is any winning condition.

Observe that a distributed game is a bipartite game. Notice that there is an asymmetry in the definition of environment's and player's moves. In a move from player's to environment's position, all components which are players' positions must change, and the change respects transitions in local games. In the move from environment's to player's position, all components are environment's positions but only some of them need to change; moreover these changes need not to respect local transitions. Hence, while global moves of the player are a kind of free product of moves in local games, it is not the case for the environment. The moves from environment positions are the only part of a distributed game that is not determined by the choice of components, i.e., of local games. This freedom makes it possible to encode different communication patterns and other phenomena.

We interpret a distributed game as a game of n players against environment. This intuition will become clear when we will define the notions of views and local strategies.

For an n-tuple η and $i = 1, \ldots, n$, let $\eta[i]$ denote the i-th component of η. Similarly, for a sequence $\boldsymbol{v} = \eta_1\eta_2 \ldots$ of n-tuples, let $\boldsymbol{v}[i] = \eta_1[i]\eta_2[i] \ldots$ denote the projection of the sequence on the i-th component.

From the definition of the moves it is easy to observe that given a play $\boldsymbol{v}$ in a distributed game $\mathcal{G}$, the projection of $\boldsymbol{v}$ to the positions of the i-local game, $\boldsymbol{v}[i]$, is of the form $e_0^+ p_0 e_1^+ p_1 \ldots$ Note that the player's positions do not repeat since as soon as the local game moves to a player position, it reacts immediately with an environment position.

Definition 1. *Consider a play $\boldsymbol{v}$ and let $e_0^+ p_0 e_1^+ p_1 \ldots$ be the projection of $\boldsymbol{v}$ on i-th component. The* view of process i *of $\boldsymbol{v}$ is $view_i(\boldsymbol{v}) = e_0 p_0 e_1 p_1 \ldots$*

Definition 2. *An i-*local strategy *is a strategy in the game G_i. A* distributed (player) strategy *σ is a tuple of local strategies $\langle \sigma_1, \ldots, \sigma_n \rangle$.*

A distributed strategy σ defines a strategy in $\mathcal{G}$ by $\sigma(\boldsymbol{v} \cdot (x_1, \ldots, x_n)) = (e_1, \ldots, e_n)$ where $e_i = x_i$ if $x_i \in E_i$ and $e_i = \sigma_i(view_i(\boldsymbol{v} \cdot x_i))$ otherwise. We call σ the *global strategy* associated with the given distributed player strategy.

Remark 1. It is important to note that thanks to the definition of distributed game any tuple of local strategies indeed defines a (global) strategy, i.e., it always suggests a valid move.

Examples and easy observations. Consider the two local games G_1 and G_2, presented in Figure 1 where players' positions are marked by squares and environment's positions by circles. Observe that in the second game the moves of the player are more restricted than in the first game.

Fig. 1. Local games

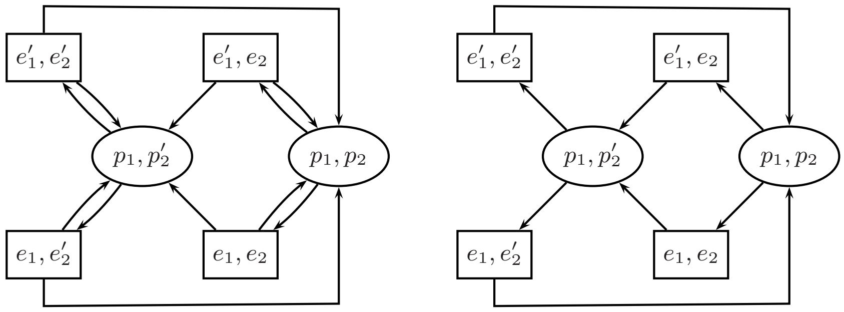


Fig. 2. Global games

Consider a (part of) distributed game built from G_1 and G_2 presented on the left part of Figure 2. Observe that in this game environment has less possibilities than it would have in the free product of G_1 and G_2. For example, from the position e'_1 in G_1 environment can move to p'_1, similarly from e'_2 in G_2 it can move to p'_2; but there is no move from (e'_1, e'_2) to (p'_1, p'_2) in the distributed game.

Suppose that the winning condition in this game is to avoid environment's positions where the components have different polarities, i.e., vertices (e_1, e'_2) and (e'_1, e_2). It is clear that there is a global winning strategy in this game. In position (p_1, p_2) players should go to (e_1, e_2) and in position (p_1, p'_2) they should go to (e'_1, e'_2). We claim that there is no distributed strategy. Suppose conversely that we have a distributed strategy $\langle \sigma_1, \sigma_2 \rangle$ which is winning from the vertex (e_1, e_2). If environment moves to the position (p_1, p'_2) then player 1 should respond with e'_1. Hence $\sigma_1(e_1 p_1) = e'_1$. But now, if environment moves to (p_1, p_2) then the view of player 1 of the play is the same, so he moves also to $\sigma_1(e_1 p_1) = e'_1$, which is a losing move.

As another example consider the game on the right of Figure 2. This is almost the same game as before but for some more suppressed moves of the environment. Once again there is a global winning strategy in this game. But this time there is also a distributed strategy. Define $\sigma_1(\boldsymbol{v} p_1)$ to be e'_1 if the number of p_1 in $\boldsymbol{v}$ is even and to be e_1 otherwise. Let $\sigma_2(\boldsymbol{v} p_2) = e_2$ and $\sigma_2(\boldsymbol{v} p'_2) = e'_2$. It is easy to verify that $\langle \sigma_1, \sigma_2 \rangle$ is a distributed strategy winning from (e_1, e_2). Observe that strategy σ_1 is not memoryless. It is easy to check that there is no distributed strategy which is a memoryless strategy for player 1.

The following fact summarizes the observations we have made above.

Proposition 1. *There exist distributed games with a global winning strategy for the players but no distributed winning strategy. There exist distributed games with a memoryless global strategy but where all distributed strategies require memory.*

It is not difficult to show that it is not decidable to check if there is a distributed winning strategy in a given distributed game. The argument follows the same lines as, for example, in [29, 16, 14, 4].

Proposition 2. *The following problem is undecidable: Given a finite distributed game check if there is a distributed winning strategy from a given position.*

Recall that by Theorem 1 it is decidable if there is a global winning strategy in a finite game. There are two cases when existence of a global winning strategy guarantees existence of a distributed winning strategy. The first is when we just have one player. The second is when a game is *environment deterministic*, i.e., if each environment position has exactly one successor (like in the second example above).

Proposition 3. *If there is a global winning strategy in an environment deterministic game then there is a distributed winning strategy from a given position.*

4 Division Operation

Let us assume that we have a distributed game $\mathcal{G} = \langle P, E, T, Acc\rangle$ with $n+1$ players constructed from local games $G_i = \langle P_i, E_i, T_i\rangle$. We would like to construct an equivalent game with n players. This will be possible if some of the players can deduce the global state of the game.

Definition 3. *A game $\mathcal{G}$ is i-*deterministic *if for every environment position η of $\mathcal{G}$ and every $(\eta, \pi_1), (\eta, \pi_2) \in T_e$, if $\pi_1 \neq \pi_2$ then $\pi_1[i], \pi_2[i] \in P_i$ and $\pi_1[i] \neq \pi_2[i]$.*

Intuitively, the definition implies that player i can deduce the global position of the game from its local view.

We use two functions for rearranging tuples: $\mathit{flat}((x_0, x_n), x_1, \ldots, x_{n-1}) = (x_0, x_1, x_2, \ldots, x_n)$ and $flat^{-1}(x_0, x_1, x_2, \ldots, x_n) = ((x_0, x_n), x_1, \ldots, x_{n-1})$. We extend these functions point-wise to sequences.

Division operation For the game $\mathcal{G}$, we define $\textsc{divide}(\mathcal{G}) = \langle \widetilde{P}, \widetilde{E}, \widetilde{T}, \widetilde{Acc}\rangle$. It consists of the local games $\widetilde{\mathcal{G}}_i = \langle \widetilde{P}_i, \widetilde{E}_i, \widetilde{T}_i\rangle$ $(i = 0, \ldots, n-1)$ where:

- $\widetilde{P}_i = P_i$, $\widetilde{E}_i = E_i$ and $\widetilde{T}_i = T_i$ for $i = 1, \ldots, n-1$;
- $\widetilde{E}_0 = E_0 \times E_n$ and $\widetilde{P}_0 = (P_0 \cup E_0) \times (P_n \cup E_n) \setminus \widetilde{E}_0$;
- In $\widetilde{T}_0$ we have transitions from (p_0, e_n), (e_0, p_n), (p_0, p_n) to (e_0, e_n) provided $(p_0, e_0) \in T_0$ and $(p_n, e_n) \in T_n$.

The global moves from the environment positions in $\textsc{divide}(\mathcal{G})$ and the acceptance condition are defined by:

- $\eta \to \pi \in \widetilde{T}$ if $\mathit{flat}(\eta) \to \mathit{flat}(\pi) \in T$,
- $\widetilde{Acc} = \{\boldsymbol{v} : \mathit{flat}(\boldsymbol{v}) \in Acc\}$.

Theorem 2. *Let $\mathcal{G}$ be a 0-deterministic and n-deterministic distributed game of $n+1$ players. For every position η of $\mathcal{G}$, there is a distributed winning strategy from η iff there is one from $\mathit{flat}^{-1}(\eta)$ in $\widetilde{\mathcal{G}}$.*

5 Gluing Operation

Let us assume that we have a game $\mathcal{G} = \langle P, E, T, Acc \rangle$ constructed from $n+1$ local games $G_0, \ldots, G_n$. We are going to define an operation GLUE which is like determinizing the behaviour of the environment for one of the players. This is sometimes a necessary step before being able to apply the division operation. As

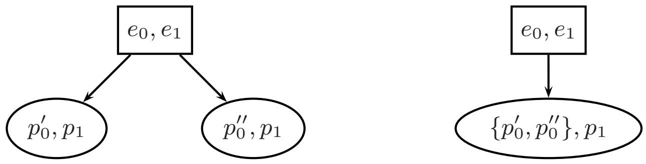


Fig. 3. Gluing operation

an example suppose that in a distributed game we have the moves from (e_0, e_1) as on the left side of Figure 3. There are two transitions with the same player 1 positions, so the game is not 1-deterministic and we cannot apply the division operation. Gluing together the possibilities for player 0 (as depicted on the right) we make this part of the game deterministic for both players.

For gluing operation to work, the game should satisfy certain conditions. There are in fact two sets of conditions describing different reasons for the operation to work. The first is when, the player being glued has almost complete information about the whole game. In the second, he almost does not influence the behaviour of other components.

Definition 4. *A game $\mathcal{G}$ is* I-gluable *if it satisfies the following conditions.*

1. *$\mathcal{G}$ is 0-deterministic;*
2. *$\mathcal{G}$ has no 0-delays: if $(e_0, e_1, \ldots, e_n) \to (x_0, x_1, \ldots, x_n)$ then $x_0 \in P_0$;*
3. *The winning condition Acc is a parity condition on player 0: there is a map $\Omega : (P_0 \cup E_0) \to \mathbb{N}$ such that $\boldsymbol{v} \in Acc$ iff $\liminf_{i \to \infty} \Omega(view_0(\boldsymbol{v}))$ is even.*

Definition 5. *A game $\mathcal{G}$ is* II-gluable *if it satisfies the following conditions:*

1. *The moves of other players are not influenced by player 0 moves, i.e., if $(e_0, e_1, \ldots, e_n) \to (x_0, x_1, \ldots, x_n)$ then for every other environment position e'_0 we have $(e'_0, e_1, \ldots, e_n) \to (x'_0, x_1, \ldots, x_n)$ for some x'_0.*
2. *The moves of player 0 are almost context independent: there is an equivalence relation $\sim \subseteq (E_0 \times P_0)^2$ s.t. if $(e_0, e_1, \ldots, e_n) \to (p_0, x_1, \ldots, x_n)$ then for every (e'_0, p'_0): $(e'_0, e_1, \ldots, e_n) \to (p'_0, x_1, \ldots, x_n)$ iff $(e'_0, p'_0) \sim (e_0, p_0)$.*
3. *$\mathcal{G}$ has no 0-delays.*
4. *The winning condition is a conjunction of the winning condition for players 1 to n and the condition for player 0. Additionally, the condition for player 0 is the parity condition.*

Glue operation We define the game $\widetilde{G} = \text{GLUE}(\mathcal{G})$ of $n+1$ players as follows (to make the notation lighter we will use abbreviated notation for tuples, for example we will write $(e_0, \overline{e})$ instead of $(e_0, e_1, \ldots, e_n)$):

- $\widetilde{P}_i = P_i$, $\widetilde{E}_i = E_i$ and $\widetilde{T}_i = T_i$ for all $i = 1, \ldots, n$;
- $\widetilde{P}_0 = \mathcal{P}(E_0 \times P_0)$ and $\widetilde{E}_0 = \mathcal{P}(P_0 \times E_0)$;
- $\widetilde{p} \rightarrow_0 \widetilde{e}$ if for every $(e,p) \in \widetilde{p}$ there is $(p, e') \in \widetilde{e} \cap T_0$;
- $(\widetilde{e}_0, \overline{e}) \rightarrow (\widetilde{x}_0, \overline{x}) \in \widetilde{T}$ for $\widetilde{x}_0 \neq \emptyset$, where $\widetilde{x}_0 = \{(e_0, x_0) : \exists (p', e_0) \in \widetilde{e}_0.\ (e_0, \overline{e}) \rightarrow (x_0, \overline{x})\}$.
- $\widetilde{Acc}$ will be defined shortly.

Consider $\boldsymbol{u} = u_1, \ldots u_{2k} \in (\widetilde{E}_0 \cdot \widetilde{P}_0)^+$. It is a sequence of sets of pairs of nodes of the game G_0. A *thread in* $\boldsymbol{u}$ is any sequence $e_1 p_1 \cdots e_k p_k \in (E_0 P_0)^+$ such that $(p_{i-1}, e_i) \in u_{2i-1}$ and $(e_i, p_i) \in u_{2i}$ for all $i = 1, \ldots, k$. Similarly we define threads for infinite sequences. Let $\mathit{threads}(\boldsymbol{u})$ be the set of threads in $\boldsymbol{u}$. We put:

$$\begin{aligned} \boldsymbol{u} \in \widetilde{Acc} \quad \text{iff} \quad & \text{every } \boldsymbol{v} \in \mathit{threads}(\mathit{view}_0(\boldsymbol{u})) \text{ satisfies the parity condition } \Omega \\ & \text{and } \boldsymbol{u} \text{ satisfies the conditions for players } 1, \ldots, n \end{aligned}$$

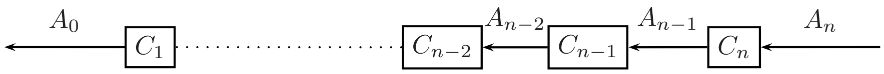
Observe that if a game is I-gluable then the winning condition is only on 0-th player and the second clause in the definition of $\widetilde{Acc}$ is automatically true.

Theorem 3. *Let $\mathcal{G}$ be a I-gluable or II-gluable game. There is a distributed winning strategy from a position η in $\mathcal{G}$ iff there is a distributed winning strategy from the position $\widetilde{\eta}$ in* $\text{GLUE}(\mathcal{G})$.

The proof of this theorem is relatively long and it is not presented here. While in principle it uses similar methods as in determinization of automata on infinite words, some arguments need to be refined [20].

6 Synthesis for Pipeline Architecture

A pipeline is a sequence of processes communicating by unidirectional channels:



We assume that the alphabets $A_0, \ldots, A_n$ are disjoint. The execution follows in rounds. Within a round, processes get inputs and produce outputs in a step-wise fashion. At the beginning of a round, process C_n gets input $a_n \in A_n$ from the environment and gives an output $a_{n-1} \in A_{n-1}$. In the next step, this output is given as input to process C_{n-1} and so on. When C_1 has given an output, the round finishes and another round starts.

A *local controller* for the i-th component is a function $f_i : (A_i)^* \to A_{i-1}$. A sequence $a_0 b_0 a_1 b_1 \cdots \in (A_i \cdot A_{i-1})^\omega$ *respects* f_i if $f_i(a_0 a_1 \dots a_j) = b_j$ for all j.

A *pipeline controller* P is a tuple of local controllers $\langle f_1, \dots, f_n \rangle$, one for each component. An execution of the pipeline is a string in $(A_n A_{n-1} \cdots A_0)^\omega$. An execution $\boldsymbol{v}$ respects P if $\boldsymbol{v}|(A_i \cup A_{i-1})$ respects f_i, for all $i = 1, \dots, n$.

Let $\Sigma = \bigcup_{i=0\dots,n} A_i$. A controller P defines a set of Σ-labeled paths $\mathcal{L}(P)$ which is the set of all the executions respecting P.

The *pipeline synthesis problem* is: given a pipeline over alphabets $A_0, \dots, A_n$ and a deterministic parity word automaton $\mathcal{A}$ over the alphabet $\Sigma = \bigcup_{i=0\dots,n} A_i$, find a pipeline controller $P = \langle f_1, \dots, f_n \rangle$ such that $\mathcal{L}(P) \subseteq L(\mathcal{A})$.

We would like to remark that in the proof presented here there is no difficulty to consider branching specifications, i.e., tree automata [13]. We restrict to word automata because we have no space to give the definition of automata on trees with nodes of varying degrees.

Encoding into a game A pipeline synthesis problem is coded as a distributed game $\mathcal{G} = \langle P, E, T, Acc \rangle$ constructed from local games $G_0, \dots, G_n$, with G_0 taking the role of the automaton $\mathcal{A} = \langle Q, \Sigma, q^0, \delta : Q \times \Sigma \to Q, \Omega : Q \to \mathbb{N} \rangle$ and G_i the role of the i-th component C_i. The game G_0 is: (1) $P_0 = Q \times \Sigma^{n+1}$; $E_0 = Q$; (2)$(q, w) \to q' \in T_0$ if $q' = \delta(q, w)$; and $q \to (q, w) \in T_0$ for all $w \in \Sigma^{n+1}$.

For each component $i = 1, \dots, n$ we have the game G_i which is defined by: $P_i = A_i$; $E_i = (A_i \to A_{i-1})$; and there is a complete set of transitions between P_i and E_i.

From an environment position $(q, f_1, \dots, f_n)$, for a letter $a_n \in A_n$ we have a move to $((q, w(a_n)), a_1, \dots, a_n)$ where $w(a_n) = a_n a_{n-1} \dots a_0$ is a word such that $a_{i-1} = f_i(a_i)$.

The winning condition Acc is the set of sequences such that the projection on the states in the first component satisfies the the parity condition of the automaton $\mathcal{A}$. Here we need to assume some special form of $\mathcal{A}$. This is because we "jump" over the states by using $\delta(q, w)$ for w a word of $n+1$ letters. We need to be sure that while doing this we do not jump over states of small priority. As the length of w is fixed we can easily guarantee this. The initial position is $\eta_0 = (q^0, a_1, \dots, a_n)$ for some arbitrarily chosen letters $a_1, \dots, a_n$.

Lemma 1. *There is a distributed winning strategy in $\mathcal{G}$ from η_0 iff the pipeline synthesis problem has a solution. A distributed winning strategy gives a controller for the pipeline.*

Decidability We will abstract some properties of a pipeline game and show that for any game with these properties it is decidable to establish if there exists a distributed winning strategy.

Definition 6. *A game $\mathcal{G}$ is* i-sequential *if for all environment positions η_1 and η_2: if $\eta_1 \to \pi_1$, $\eta_2 \to \pi_2$, $\eta_1[1, i] = \eta_2[1, i]$ and $\pi_1[1, i-1] \neq \pi_2[1, i-1]$ then $\pi_1[i] \neq \pi_2[i]$ and $\pi_1[i], \pi_2[i] \in P_i$. Here we use $\eta[1, i-1]$ to denote the subsequence of the sequence η consisting of elements on positions from 1 to $i-1$; note that tuple η has also 0 position.*

Definition 7. *We call a game* $\langle 0, n \rangle$-proper *if it satisfies the following:*

P1 $\mathcal{G}$ *is* 0*-deterministic, has no* 0*-delays, and its winning condition is a parity condition on player* 0*;*
P2 $\mathcal{G}$ *is* n*-deterministic;*
P3 $\mathcal{G}$ *is* i*-sequential for all* $i \in \{1, \ldots, n\}$.

A game is $\langle 0, n \rangle$*-almost proper* if it is proper except that it may not satisfy P2. Observe that the condition P3 does not imply P2, as sequentiality does not say anything about player 0.

The following results are direct consequences of the definitions.

Lemma 2. *The following are true about (almost) proper games:*

- *The pipeline game* $\mathcal{G}$ *is* $\langle 0, n \rangle$*-proper.*
- *A* $\langle 0, n \rangle$*-proper game* $\mathcal{G}$ *is dividable and* DIVIDE($\mathcal{G}$) *is a* $\langle 0, n-1 \rangle$*-almost proper game.*
- *A* $\langle 0, n \rangle$*-almost proper game* $\mathcal{G}$ *is I-gluable and* GLUE($\mathcal{G}$) *is a* $\langle 0, n \rangle$*-proper game.*

Corollary 1. *Existence of a distributed strategy in a* $\langle 0, n \rangle$*-proper game is decidable. The synthesis problem for pipelines is decidable.*

Even though our definition of pipeline architecture is slightly different from those in Pnueli and Rosner [29] or Kupferman and Vardi [13], one can see that they can be captured by our definition. One can also see that two-way pipelines and one-way rings of [13] give rise to $\langle 0, n \rangle$-proper games. Hence, we get decidability results for all these classes of architectures at one go. The translation of two way rings does not give $\langle 0, n \rangle$-proper games. Indeed the synthesis problem for this architecture is undecidable.

7 Conclusions

We have introduced a notion of distributed games as a framework for solving distributed synthesis problems (DSP). We have tried to make the model as specific as possible but still capable to easily encode the particular instances of DSP found in the literature. This deliberate restriction of the model is the main difference between our approach and the general models proposed in the literature [23, 2, 7].

Having decided on a model we have looked for a set of tools that can be applied to different instances of DSP. We have given two theorems allowing to simplify distributed games. We have shown that they can be used to solve the pipeline synthesis problem. In the full version of the paper [20] we consider three more problems: local specifications and double-flanked pipelines [17, 16], synthesis for communicating state machines of Madhusudan and Thiagarajan [18], and decentralized control synthesis of Rudie and Wonham [32]. We give these examples in order to show that the framework of distributed games is rich enough to model different synthesis problems proposed in the literature. This way we also show that the two simplification theorems we propose are powerful enough to solve known decidable cases of the distributed synthesis problem. The advantage of our approach is that we separate the proofs into two steps: coding an instance in question as a distributed game model and use of simplification theorems. While both steps are not completely straightforward, they nevertheless allow some modularization of the proof and reuse of the general results on distributed games.

We hope that distributed games will be useful in exploring the borderline between decidability and undecidability of DSP. For example, the only architectures for which the DSP problem with local specifications is decidable are doubly flanked pipelines. The undecidability arguments for other architectures use quite unnatural specifications that require a process to guess what will be its next input. We hope to find interesting classes of pairs (architecture, specification) for which the DSP problem is decidable. We want to do this by encoding the problem into games and looking at the structure of resulting games. In a similar way we want to find decidable instances of DSP in the discrete control synthesis framework.

Acknowledgments: The authors are very grateful to Julien Bernet and David Janin for numerous discussions and valuable contributions.

References

1. M. Abadi, L. Lamport, and P. Wolper. Realizable and unrealizable specifications of reactive systems. In *ICALP'89*, volume 372 of *Lecture Notes in Computer Science*, pages 1–17, 1989.
2. R. Alur, T. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *FOCS'97*, pages 100–109, 1997.
3. S. Ambroszkiewicz and W. Penczek. Local interactions, explicit communication and causal knowledge in games and multi-agent systems. In *CEEMAS'99*, St. Petersburg, 1999.
4. A. Arnold, A. Vincent, and I. Walukiewicz. Games for synthesis of controllers with partial observation. *Theoretical Computer Science*, 303(1):7–34, 2003.
5. H. Attiya and J. Welch. *Distributed Computing : Fundamentals, Simulations, and Advanced Topics*. McGraw-Hill Publishing Company, UK, 1998.
6. A. Bergeron. A unified approach to control problems in discrete event processes. *RAIRO-ITA*, 27:555–573, 1993.
7. J. Bradfield. Independence: logic and concurrency. In *LCS'00*, volume 1682 of *Lecture Notes in Computer Science*, 2000.
8. V. Diekert and G. Rozenberg, editors. *The Book of Traces*. World Scientific, 1995.
9. E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *Proc. FOCS 91*, pages 368–377, 1991.
10. R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about knowledge*. MIT Press, 1995.
11. J. Y. Halpern and R. Fagin. A formal model of knowledge, action and communication in distributed systems. In *Proc. of ACM Symp. on Principles of Distributed Computing*, pages 224–236, 1985.

12. J. Y. Halpern and L. Zuck. A little knowledge goes a long way: Simple knowledge based derivations and correctness proofs for a family of protocols. *Journal of the ACM*, 39(3):449–478, 1992.
13. O. Kupferman and M. Vardi. Synthesizing distributed systems. In *Proc. 16th IEEE Symp. on Logic in Computer Science*, July 2001.
14. H. Lamouchi and J. G. Thistle. Effective control synthesis for DES under partial observations. In *Proc. 39th IEEE Conf. on Decision and Control*, December 2000.
15. F. Lin and M. Wonham. Decentralized control and coordination of discrete-event systems with partial observation. *IEEE Transactions on automatic control*, 33(12):1330–1337, 1990.
16. P. Madhusudan. *Control and Synthesis of Open Reactive Systems*. PhD thesis, University of Madras, 2001.
17. P. Madhusudan and P. Thiagarajan. Distributed control and synthesis for local specifications. In *ICALP'01*, volume 2076 of *Lecture Notes in Computer Science*, 2001.
18. P. Madhusudan and P. Thiagarajan. A decidable class of asynchronous distributed controllers. In *CONCUR'02*, volume 2421 of *Lecture Notes in Computer Science*, 2002.
19. D. Martin. Borel determinacy. *Ann. Math.*, 102:363–371, 1975.
20. S. Mohalik and I. Walukiewicz. Distributed games: full version with complete proofs. http://www.labri.fr/∼igw/publications.html.
21. P. Morris. *Introduction to game theory.* Springer-Verlag, 1994.
22. A. W. Mostowski. Games with forbidden positions. Technical Report 78, University of Gdansk, 1991.
23. M. J. Osborne and A. Rubinstein. *A Course in Game Theory.* The MIT Press, 1994.
24. R. Parikh. Social software. Synthese, 2002 (To appear).
25. R. Parikh and R. Ramanujam. Distributed processing and logics of knowledge. In *Proc. Workshop on logics of programs*, Volume 193 of *Lecture Notes in Computer Science*, pages 256–268, 1985.
26. M. Pauly. *Logic for social software.* PhD thesis, Institute for Logic, Language and computation, Universiteit van Amsterdam, 2001. ILLC Dissertation Series 2001-10.
27. G. L. Peterson and J. H. Reif. Multi-person alternation. In *Proc. IEEE FOCS*, pages 348–363, 1979.
28. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. ACM POPL*, pages 179–190, 1989.
29. A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *31th IEEE Symposium Foundations of Computer Science (FOCS 1990)*, pages 746–757, 1990.
30. P. J. G. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77, 1989.
31. S. Ricker and K. Rudie. Incorporating knowledge into discrete-event control systems. *IEEE Trans. on Automatic Control*, 45(9):1656–1668, 2000.
32. K. Rudie and W. Wonham. Think globally, act locally: Decentralized supervisory control. *IEEE Trans. on Automat. Control*, 37(11):1692–1708, 1992.
33. W. Thomas. On the synthesis of strategies in infinite games. In *STACS '95*, volume 900 of *Lecture Notes in Computer Science*, pages 1–13, 1995.
34. S. Tripakis. Undecidable problems of decentralized observation and control. In *IEEE Conference on Decision and Control*, 2001.