Distributed Games

Swarup Mohalik[1] and Igor Walukiewicz[2]

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.

[1]Sasken Communication Technologies Ltd., Bangalore

[2]LaBRI, CNRS, 351, cours de la Libration, 33405 Talence Cedex, France, igw@labri.fr

Earlier approaches. The distributed synthesis problem have been considered by Pnueli and Rosner in the setting of an architecture with fixed channels of communication among processes [27]. They have shown that distributed synthesis is undecidable for most classes of architectures. They have obtained decidability 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 [25]. After the work of Pnueli and Rosner, the results has 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 process. 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 [28, 15, 30]. 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 events of the plant. While the original problem refers only to safety properties, an extension to the μ-calculus specifications has been also considered [4]. Except for some special cases, the problem turns out to be undecidable ([4, 32]). 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, 26, 31]. 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 repetitively applying these theorems we can simplify the game to a game 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. These two theorems are 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 [11]. The closeness is not accidental: the model was partly motivated by this 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 ([22, 20]). There have been attempts to model and logically reason about cooperation as in [23, 24]. 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 the rest of the paper we give four example applications. In Section 6 we solve the synthesis problem for pipelines. Then, we consider doubly-flanked pipelines with local specifications. Next, we apply our setting to the synthesis problem for communicating state machines of Madhusudan and Thiagarajan [18]. Finally, we consider decentralized control synthesis of Rudie and Wonham [30]. 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.

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 $v_0v_1v_2\ldots$ 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 $\vec{v}$ ending in a vertex v from P a vertex $\sigma(\vec{v})$ which is a successor of v. The strategy is *memoryless* iff $\sigma(\vec{v}) = \sigma(\vec{w})$ whenever $\vec{v}$ and $\vec{w}$ end in the same vertex.

A *play respecting* σ is a sequence $v_0v_1\ldots$ 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 $L \subseteq Colours^\omega$, such that

$$Acc = \{v_0v_1\ldots \in V^\omega : \lambda(v_0)\lambda(v_1)\ldots \in L\}$$

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:

$$Acc = \{v_0v_1\ldots \in V^\omega : \liminf_{i\to\infty} \Omega(v_i) \text{ is even}\}$$

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, 21])
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. Similarly for the environment. 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 $x_i \to x'_i \in T_i$.
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. In the move from environment's to player's position, all components are environment's positions but only some of them need to change. The other difference is that the set of moves from the environment's position need not be a free product of local moves, i.e., some moves of the environment in local games may be blocked in the global game (cf. the following subsection with examples). Because of this, it is not necessary to specify moves from environment's to player's positions in local games. We can always assume that in local games environment can move to any player's position, then we can put restrictions in the definition of the global game.

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 $l = 1, \ldots, n$, let $\eta[l]$ denote the l'th component of η. Similarly, for a sequence $\vec{v} = \eta_1\eta_2\ldots$ of n-tuples, let $\vec{v}[l] = \eta_1[l]\eta_2[l]\ldots$ denote the projection of the sequence on the l'th component.

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

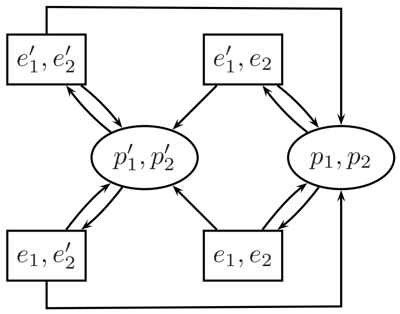
Definition 2 Consider a play $\vec{v}$ and let $e_0^+ p_0 e_1^+ p_1 \dots$ be the projection of $\vec{v}$ on i-th component. The *view of process* i of $\vec{v}$ is $view_i(\vec{v}) = e_0 p_0 e_1 p_1 \dots$

- An *i-local strategy* is a strategy in the game G_i.
- A *distributed strategy* σ is a tuple of local strategies $\langle \sigma_1, \dots, \sigma_n \rangle$. It defines a strategy in $\mathcal{G}$ by $\sigma(\vec{v}(x_1, \dots, x_n)) = (e_1, \dots, e_n)$ where $e_i = x_i$ if $x_i \in E_i$ and $e_i = \sigma_i(view_i(\vec{v} \cdot x_i))$ otherwise. We call σ the *global strategy* associated with the given distributed strategy.

Remark: 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 games, where players' positions are marked by squares and environments' positions by circles:

The difference between the two games is that in the second the moves of the player are restricted. Consider a distributed game build from G_1 and G_2:

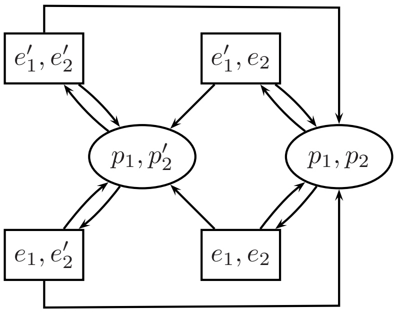


Observe that in this game environment has less possibilities than it would have in the free product of G_1 and G_2. For example, from the position e_1' in G_1 environment can move to p_1, similarly from e_2' in G_2 it can move to p_2'; but there is no move from (e_1', e_2') to (p_1, p_2') in the distributed game.

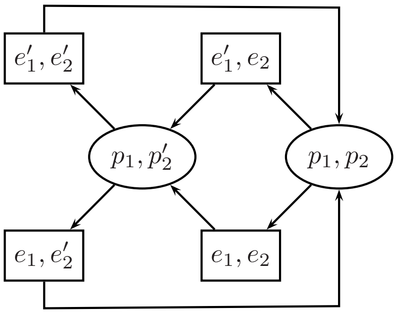
Suppose that the winning condition in this game is to avoid environments' positions where the components have different polarities, i.e., vertices (e_1, e_2') and (e_1', e_2).

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). This strategy is also a distributed strategy: player 1 chooses e_1 in p_1 and e'_1 in p'_1; similarly for player 2.

Now let us modify the example a little.



This is a global game build from the same local games G_1 and G_2 but the moves of the environment are different. Instead to the position (p'_1, p'_2) we have a move to the position (p_1, p'_2). It is clear that there is a global winning strategy in this game. 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



This is almost the same game as before but for some suppressed moves of the environment. Once again there is a global winning strategy in this game. But there is also a distributed strategy. Define $\sigma_1(\vec{v}p_1)$ to be e'_1 if the number of p_1 in $\vec{v}$ is even and to be e_1 otherwise. Let $\sigma_2(\vec{v}p_2) = e_2$ and $\sigma_2(\vec{v}p'_2) = e'_2$. 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.

Fact 3 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 [27, 16, 14, 4].

Fact 4 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 strategy guarantees existence of a local 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).

Fact 5 If there is a global winning strategy in an environment deterministic game then there is a distributed winning strategy from a given position.

Proof
A global winning strategy in an environment deterministic game determines a single path in the game. Take some player i nad a prefix $\vec{v}$ of this path such that $view_i(\vec{v})$ is a player's position. It is easy to see that $view_i(\vec{v})$ determines $\vec{v}$ uniquely. This is because each time i-th component of a global position is a player's position, in the next step it changes to environments position. Hence, given $view_0(\vec{v})$ player i can make exactly the move that is suggested by the global strategy from $\vec{v}$. We define this way all the local strategies. Resulting distributed strategy will choose exactly the same path in the game as the initial global strategy. □

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 6 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.

Fact 7 In an i-deterministic game $\mathcal{G}$, if $(\eta, \pi) \in T_e$ and $\eta[i] = \pi[i] \in E_i$, then π is the only successor of η.

Proof
If η had more than one successors, then by the definition of i-deterministic games, each one of them would have positions from P_i. This contradicts the assumption that $\pi[i] \in E_i$. $\square$

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

Theorem 8
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 $\widetilde{\mathcal{G}}$.

The theorem follows from the two lemmas below. We will use $view_{01}(\vec{v})$ as an abbreviation of $view_1(view_0(\vec{v}))$ for $\vec{v}$ a path in $\widetilde{\mathcal{G}}$. Similarly for $view_{00}$. Observe that $view_{01}(\vec{v})$ is a sequence of positions form $(E_n \cup P_n)$ and $view_{00}(\vec{v})$ is a sequence of positions form $(E_0 \cup P_0)$

Lemma 9 If there is a distributed winning strategy in $\mathcal{G}$ from η_0 then there is one in $\widetilde{\mathcal{G}}$ from $\widetilde{\eta}_0 = \mathit{flat}^{-1}(\eta_0)$.

Proof
Let us assume that there is a strategy in $\mathcal{G}$. This means that there are strategies $\sigma_i : (E_i P_i)^+ \to E_i$ for all $i = 0, \ldots, n$. We define strategies in $\widetilde{\mathcal{G}}$. For $i = 1, \ldots, n-1$ the strategies do not change, i.e., $\widetilde{\sigma}_i = \sigma_i$. Strategy $\widetilde{\sigma}_0 : (\widetilde{E}_0 \widetilde{P}_0)^+ \to \widetilde{E}_0$ is defined by

$$\widetilde{\sigma}_0(\vec{v}) = \begin{cases} (\sigma_0(view_{00}(\vec{v})), e_n) & \text{if } last(\vec{v}) = (p_1, e_n) \\ (e_0, \sigma_n(view_{01}(\vec{v}))) & \text{if } last(\vec{v}) = (e_0, p_n) \\ (\sigma_0(view_{00}(\vec{v})), \sigma_n(view_{01}(\vec{v}))) & \text{if } last(\vec{v}) = (p_0, p_n) \end{cases}$$

It is clear from the above definition that $\widetilde{\sigma}$ is defined for all plays in $\widetilde{\mathcal{G}}$.

By induction on the length we show that for $\vec{u}$ in $\widetilde{\mathcal{G}}$ respecting $\widetilde{\sigma}$, $flat(\vec{u})$ is a play in $\mathcal{G}$ and $flat(\vec{u})$ respects σ. The base case is when $\vec{u} = \widetilde{\eta}_0$ and it holds trivially. The induction step follows directly from the definitions.

Finally, we show that the distributed strategy $\langle \widetilde{\sigma}_0, \widetilde{\sigma}_1, \ldots, \widetilde{\sigma}_{n-1} \rangle$ is winning in $\widetilde{\mathcal{G}}$. Take an infinite play $\vec{u}$ in $\widetilde{\mathcal{G}}$ respecting $\widetilde{\sigma}$. All the finite prefixes of $\vec{u}$ respect $\widetilde{\sigma}$ so all the prefixes of $flat(\vec{u})$ respect σ. Hence $\mathit{flat}(\vec{u})$ respects σ, the distributed strategy in $\mathcal{G}$. Therefore, $\mathit{flat}(\vec{u}) \in \mathit{Acc}$ which implies $\vec{u} \in \widetilde{\mathit{Acc}}$. $\square$

Lemma 10 If there is a distributed winning strategy in $\widetilde{\mathcal{G}}$ from $\widetilde{\eta}_0$ then there is one in $\mathcal{G}$ from $\eta_0 = \mathit{flat}(\widetilde{\eta}_0)$.

Proof

Suppose that there is a distributed winning strategy in $\widetilde{\mathcal{G}}$. We define strategies in $\mathcal{G}$. Once again the strategies for players $1, \ldots, n-1$ do not change. From the strategy $\sigma_0 : (\widetilde{E}_0 \cdot \widetilde{P}_1)^+ \to \widetilde{E}_1$ we need to define two strategies σ_0 and σ_n. In order to do this, we need the following construction.

Let $\widetilde{\mathcal{G}}(\widetilde{\sigma}_0, \ldots, \widetilde{\sigma}_n)$ be the graph of the game $\widetilde{\mathcal{G}}$ where the moves of the players are fixed by the respective strategies. The main observation is that for a path $\vec{v} \in (E_0 \cdot P_0)^+$ there is at most one path $\vec{u}$ in $\widetilde{\mathcal{G}}(\widetilde{\sigma}_0, \ldots, \widetilde{\sigma}_{n-1})$ such that $\mathit{view}_{00}(\vec{u})) = \vec{v}$. Assuming conversely that there exists two such paths, say $\vec{u}_1$ and $\vec{u}_2$. We have either:

Case 1 : There is the earliest environment position η and two player positions $\pi_1 \neq \pi_2$ such that $\vec{u}_1 = \vec{u} \cdot \eta \cdot \pi_1 \cdot \vec{w}_1$ and $\vec{u}_2 = \vec{u} \cdot \eta \cdot \pi_2 \cdot \vec{w}_2$. Let $\pi_1 = ((x_0, x_n), x_1, \ldots, x_n)$ and $\pi_2 = ((x'_0, x'_n), x'_1, \ldots, x'_n)$. Then, $x_0 \neq x'_0$ since $\mathcal{G}$ is 0-deterministic. This gives $\mathit{view}_{00}(\vec{u}_1)) \neq \mathit{view}_{00}(\vec{u}_2))$. A contradiction because π is of the form $((p_0, x_n), x_1, \ldots, x_{n-1})$ for p_0 a player's position.

Case 2 : $\vec{u}_1$ is a proper prefix of $\vec{u}_2$ or vice versa. In this case, $\vec{u}_2 = \vec{u}_1 \cdot \vec{w} \cdot \pi$. Hence, $\mathit{view}_{00}(\vec{u}_2))$ is strictly longer than $\mathit{view}_{00}(\vec{u}_1))$. This is also a contradiction.

The above observation allows us to define a partial function that for a path $\vec{v} \in (E_0 P_0)^+$ gives the corresponding unique path (if it exists) $\widetilde{\mathit{path}}_0(\vec{v})$ in $\widetilde{\mathcal{G}}(\widetilde{\sigma}_0, \ldots, \widetilde{\sigma}_{n-1})$. By an analogous argument we can define the partial function $\widetilde{\mathit{path}}_n(\vec{v})$ for $\vec{v} \in (E_n P_n)^+$.

Now we define strategies:

1. $\sigma_0(\vec{v}_0) = \mathit{proj}_1(\widetilde{\sigma}_0(\mathit{view}_0(\widetilde{\mathit{path}}_0(\vec{v}_0))))$ for all $\vec{v}_0 \in (E_0 \cdot P_0)^+$;

2. $\sigma_n(\vec{v}_n) = \mathit{proj}_2(\widetilde{\sigma}_0(\mathit{view}_0(\widetilde{\mathit{path}}_n(\vec{v}_n))))$ for all $\vec{v}_n \in (E_n \cdot P_n)^+$.

where proj_1 and proj_2 are the projections on the first and second component respectively.

As in the previous claim, to show that $\langle \sigma_0, \ldots, \sigma_n \rangle$ is winning in $\mathcal{G}$, it suffices to prove by induction on length that for all finite plays $\vec{v}$ in $\mathcal{G}$ respecting σ, $flat^{-1}(\vec{v})$ is a play respecting $\widetilde{\sigma}$.

The base case is when $\vec{v} = \eta_0$. Then $\widetilde{\eta}_0 = flat^{-1}(\eta_0)$ satisfies the conditions. The induction step when $last(\vec{v}) \in P$ follows directly from the definition of environment transitions. Slightly more interesting case is when $\vec{v}$ ends in an environment position preceded by a players position, say $\vec{v} = \vec{w}(x_0, \dots, x_n)(e_0, \dots, e_n)$. Consider $\widetilde{\sigma}(flat^{-1}(\vec{v})) = ((e'_0, e'_n), e'_1, \dots, e'_{n-1})$. We want to show that $e_i = e'_i$ for all i.

First, consider any $i = 1, \dots, n-1$. By the definition of *flat* function: $view_i(\vec{v}) = view_i(flat^{-1}(\vec{v}))$. If $x_i \in E_i$ the by the definition of distributed games $e_i = x_i$ and $e'_i = x_i$. If not then

$$e_i = \sigma(view_i(\vec{w})x_i) = \widetilde{\sigma}(view_i(flat^{-1}(\vec{w}))x_i) = e'_i$$

Next, we consider the case of $i = 0$. If $x_0 \in E_0$ then $e_0 = x_0$ and $e'_0 = x_0$ by the definition of the game $\widetilde{G}$. If not then we have by the definition of σ_0:

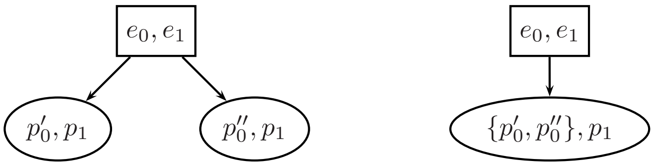
$$e_0 = proj_1(\widetilde{\sigma}_0(view_0(\widetilde{path}_0(view_0(\vec{v}))))) \quad \text{and} \quad e'_0 = proj_1(\widetilde{\sigma}_0(view_0(flat^{-1}(\vec{v}))))$$

By the definition of flat function we have that $view_{00}(flat^{-1}(\vec{v}))) = view_0(\vec{v})$. Hence, $flat^{-1}(\vec{v}) = \widetilde{path}_0(view_0(\vec{v}))$ which shows that $e_0 = e'_0$. The case of $i = n$ is analogous.

□

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 $G_0, \dots, G_n$. 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 an example suppose that in a distributed game we have the moves from (e_0, e_1) as on the left side of the picture:



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 11 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 $(e_0, e_1, \dots, e_n) \to (x_0, x_1, \dots, x_n)$ then $x_0 \in P_0$;
3. The winning condition *Acc* is a parity condition on P0: there is a map $\Omega : (P_0 \cup E_0) \to \mathbb{N}$ such that $\vec{v} \in Acc$ iff $\liminf_{i\to\infty} \Omega(view_0(\vec{v}))$ is even.

Definition 12 A game $\mathcal{G}$ is *II-gluable* if it satisfies the following conditions:

1. The moves of other players are not influenced by P0 moves: for every transition $(e_0, e_1, \dots, e_n) \to (x_0, x_1, \dots, x_n)$ and every other environment position e'_0 we have $(e'_0, e_1, \dots, e_n) \to (x'_0, x_1, \dots, x_n)$ for some x'_0.
2. The moves of payer 0 are almost context independent: there is an equivalence relation $\sim \subseteq (E_0 \times P_0)^2$ s.t. if $(e_0, e_1, \dots, e_n) \to (p_0, x_1, \dots, x_n)$ then for every (e'_0, p'_0): $(e'_0, e_1, \dots, e_n) \to (p'_0, x_1, \dots, x_n)$ 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 $\widetilde{G} = \textsc{glue}(\mathcal{G})$ 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 $(e_0, e_1, \dots, e_n)$):

- $\widetilde{P}_i = P_i$, $\widetilde{E}_i = E_i$ and $\widetilde{T}_i = T_i$ for all $i = 1, \dots, n$;
- $\widetilde{P}_0 = \mathcal{P}(E_0 \times P_0)$ and $\widetilde{E}_0 = \mathcal{P}(P_0 \times E_0)$;
- $\widetilde{p} \to_0 \widetilde{e}$ if for every $(e, p) \in \widetilde{p}$ there is $(p, e') \in \widetilde{e} \cap T_0$;
- $(\widetilde{e}_0, \overline{e}) \to (\widetilde{x}_0, \overline{x}) \in \widetilde{T}$ for $\widetilde{x}_0 \neq \emptyset$, where $\widetilde{x}_0 = \{(e_0, x_0) : \exists (p', e_0) \in \widetilde{e}_0.\ (e_0, \overline{e}) \to (x_0, \overline{x})\}$.
- $\widetilde{Acc}$ will be defined shortly.

Consider $\vec{u} = u_1, \dots u_{2k} \in (\widetilde{E}_0 \cdot \widetilde{P}_0)^+$. It is a sequence of sets of pairs of nodes of the game G_0. A *thread in* $\vec{u}$ 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 $i = 1, \dots, k$. Similarly we define threads for infinite sequences. Let $threads(\vec{u})$ be the set of threads in $\vec{u}$. We put:

$$\vec{u} \in \widetilde{Acc} \quad \text{iff} \quad \begin{array}{l} \text{every } \vec{v} \in threads(view_0(\vec{u})) \text{ satisfies the parity condition } \Omega \\ \text{and } \vec{u} \text{ satisfies the conditions for players } 1, \dots, n \end{array}$$

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 13
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 proofs are different for I-gluable and II-gluable games. Their are presented in the following two sections.

Corollary 14 Directly from the definition of the glue operation it follows that if from an environment position $(\widetilde{e}_0, \overline{e})$ we have transitions to $(\widetilde{x}_0, \overline{x})$ and to $(\widetilde{x}'_0, \overline{x}')$ then $\overline{x} \neq \overline{x}'$.

Remark: The condition "every $\vec{v} \in \mathit{threads}(\mathit{view}_0(\vec{u}))$ satisfies the parity condition Ω" is a regular winning condition. This means that there is a deterministic parity automation recognizing sequences over $(\widetilde{E}_0 \cdot \widetilde{P}_0)$ with this property. By adding this automaton as a component for player 0 positions we can convert $\text{GLUE}(\mathcal{G})$ to the game with parity winning conditions for player 0.

5.1 I-gluable games

In this section we prove Theorem 13 for I-gluable games. Fix such a game $\mathcal{G} = \langle P, E, T, Acc \rangle$ constructed from $n+1$ local games $G_0, \dots, G_n$ where $G_i = \langle P_i, E_i, T_i \rangle$ for $i = 0, \dots, n$. Since $\mathcal{G}$ is 0-deterministic, we get the following fact immediately.

Fact 15 Let $\langle \sigma_0, \dots, \sigma_n \rangle$ be a distributed strategy in $\mathcal{G}$. For a sequence $\vec{v} \in (E_0 \cdot P_0)^+$ there is at most one path $\vec{u}$ in $\mathcal{G}$ such that $\mathit{view}_0(\vec{u}) = \vec{v}$ and $\vec{u}$ respects $\langle \sigma_0, \dots, \sigma_n \rangle$.

The above fact allows us to make a definition.

Definition 16 For $\vec{v} \in (E_0 P_0)^+$, let $\mathit{path}(\vec{v})$ be the unique play respecting $\langle \sigma_0, \dots, \sigma_n \rangle$ in $\mathcal{G}$ such that $\mathit{view}_0(\mathit{path}(\vec{v})) = \vec{v}$. This function may be undefined for some $\vec{v}$ because such a play may not exist. The notation does not mention $\mathcal{G}$ and the strategy, but these will be always clear from the context.

The next definition and observation is essential for the construction. It says that when the winning conditions are of the special form that we have assumed then the strategy for player 0 can be in some sense memoryless.

Definition 17 A distributed strategy $\langle \sigma_0, \dots, \sigma_n \rangle$ is called *semi-positional for process* 0 if the local strategy σ_0 is such that $\sigma_0(\vec{u}_1) = \sigma_0(\vec{u}_2)$ whenever $\mathit{last}(\vec{u}_1) = \mathit{last}(\vec{u}_2)$ and $\mathit{view}_i(\mathit{path}(\vec{u}_1)) = \mathit{view}_i(\mathit{path}(\vec{u}_2))$ for all $i = 1, \dots, n$.

Lemma 18 Given a I-gluable distributed game $\mathcal{G}$, if there is a distributed winning strategy from a position η then there is a semi-positional one for process 0.

Proof
Let $\langle \sigma_0, \dots, \sigma_n \rangle$ be a distributed strategy in $\mathcal{G}$. Let $\mathcal{G}(\sigma_1, \dots, \sigma_n)$ be the same game but where the players $1, \dots, n$ use their strategies. Let us define this game formally. We will use notation $V_i = P_i \cup E_i$. We have that $\mathcal{G}(\sigma_1, \dots, \sigma_n) = \langle \widehat{P}, \widehat{E}, \widehat{T}, \widehat{\Omega} \rangle$ where

- $\widehat{E} = E_0 \times (V_1^* \cdot E_1) \times \cdots \times (V_n^* \cdot E_n)$,
- $\widehat{P} = (P_0 \times \Pi_{i=1,\dots,n} V_i^+) \setminus E$,
- $(e_0, \vec{v}_1 e_1, \dots, \vec{v}_n e_n) \to (x_0, \vec{w}_1 x_1, \dots, \vec{w}_n x_n) \in \widehat{T}$ when $(e_0, e_1, \dots, e_n) \to (x_0, x_1, \dots, x_n) \in T$, and $\vec{w}_i = \vec{v}_i$ if $x_i = e_i$ or $\vec{w}_i = \vec{v}_i e_i$ otherwise.
- $(x_0, \vec{v}_1 x_1, \dots, \vec{v}_n x_n) \to (e_0, \vec{w}_1 e_1, \dots, \vec{w}_n e_n)$ if
  - for e_0: $x_0 \to e_0 \in T_0$;
  - for every $\vec{w}_i e_i$: either $x_i \in E_i$ and $\vec{w}_i e_i = \vec{v}_i x_i$, or $x_i \in P_i$ and $\vec{w}_i e_i = \vec{v}_i x_i \cdot \sigma_i(\vec{v}_i x_i)$.
- $\Omega(x_0, \vec{v}_1, \dots, \vec{v}_n) = \Omega(x_0)$.

It is easy to check that if the distributed strategy is winning from a position η in $\mathcal{G}$ then its 0-the component σ_0 considered as a global strategy in $\mathcal{G}(\sigma_1, \dots, \sigma_n)$ is winning from η. As $\mathcal{G}(\sigma_1, \dots, \sigma_n)$ is a parity game there is a positional strategy σ' that is winning from η.

Using σ' we want to define a new strategy σ'_0 for player 0 in $\mathcal{G}$. Given $\vec{v} \in (E_0 \cdot P_0)^+$ we put

$$\sigma'_0(\vec{v} \cdot p_0) = \mathit{proj}_0(\sigma'(p_0, \mathit{view}_1(\vec{u}), \dots, \mathit{view}_n(\vec{u}))) \qquad \text{where } \vec{u} = \mathit{path}(\vec{v})$$

By the definition, the strategy σ'_0 is semi-positional. It can be checked that strategy $\langle \sigma'_0, \sigma_1, \dots, \sigma_n \rangle$ is winning from a position η in $\mathcal{G}$ if $\langle \sigma_0, \sigma_1, \dots, \sigma_n \rangle$ were winning from η. $\square$

Lemma 19 If there is a distributed winning strategy in $\mathcal{G}$ from $(e_0, e_1, \dots, e_n)$ then there is one in $\widetilde{\mathcal{G}}$ from $(\{(p, e_0)\}, e_1, \dots, e_n)$ for any $p \in P_0$.

Proof
By assumption, there are strategies $\sigma_i : (E_i \times P_i)^+ \to E_i$ for all $i = 0, \dots, n$. By lemma 18, we can assume that σ is semi-positional for P0. We define strategies in $\widetilde{G}$. For $i = 1, \dots, n$ the strategies do not change, i.e., $\widetilde{\sigma}_i = \sigma_i$. It remains to define a strategy $\widetilde{\sigma}_0 : (\widetilde{E}_0 \times \widetilde{P}_0)^+ \to \widetilde{E}_0$. We put

$$\widetilde{\sigma}_0(\vec{u} \cdot \widetilde{p}_0) = \{(p_0, e_0) : \exists \vec{v} \in \mathit{threads}(\vec{u} \cdot \widetilde{p}_0).\ \mathrm{last}(\vec{v}) = p_0 \text{ and } \sigma_0(\vec{v} \cdot p_0) = e_0\}$$

Since all σ_i are always defined, all $\widetilde{\sigma}_i$ are always defined too. To shorten the notation we extend the function *threads* to paths in $\widetilde{\mathcal{G}}$. For such a path $\vec{v}$ we put $\mathit{threads}_0(\vec{v}) = \mathit{threads}(\mathit{view}_0(\vec{v}))$.

To show that $\langle\widetilde{\sigma}_0,\ldots,\widetilde{\sigma}_n\rangle$ is winning we will construct for each finite play $\vec{v}$ respecting the strategy and each $\vec{w}\in \mathit{threads}_0(\vec{v})$ a path $\mathit{img}(\vec{v},\vec{w})$ in $\mathcal{G}$ such that:

- it is a play respecting $\langle\sigma_0,\ldots,\sigma_n\rangle$,
- $\mathit{view}_0(\mathit{img}(\vec{v},\vec{w}))=\vec{w}$,
- $\mathit{view}_i(\mathit{img}(\vec{v},\vec{w}))=\mathit{view}_i(\vec{v})$ for $i=1,\ldots,n$.

The base case is when $\vec{v}$ consist of just one element $(\{(p',e_0)\},e_1,\ldots,e_n)$. The unique path $\vec{w}\in \mathit{threads}_0(\vec{v})$ is e_0 and we put $\mathit{img}(\vec{v},e_0)=(e_0,e_1,\ldots,e_n)$. We have two cases for the induction step.

The first case is when $\vec{v}=\vec{v}_1\cdot(\widetilde{e}_0,e_1,\ldots,e_n)\cdot(\widetilde{p}_0,x_1,\ldots,x_n)$. Take $\vec{w}\in \mathit{threads}_0(\vec{v})$. It is of the form $\vec{w}_1\cdot p_0$ with $\vec{w}_1\in \mathit{threads}_0(\vec{v}_1\cdot(\widetilde{e}_0,e_1,\ldots,e_n))$ and $(e_0,p_0)\in\widetilde{p}_0$ where $e_0=\mathrm{last}(\vec{w}_1)$. By the induction hypothesis we have that $\mathit{img}(\vec{v}_1(\widetilde{e}_0,e_1,\ldots,e_n),\vec{w}_1)$ is of the form $\vec{u}\cdot(e_0,e_1,\ldots,e_n)$. By the definition of the game $\widetilde{\mathcal{G}}$ we have a transition $(e_0,e_1,\ldots,e_n)\to(p_0,x_1,\ldots,x_n)$ in $\mathcal{G}$. Hence we can put $\mathit{img}(\vec{v},\vec{w})=\vec{u}\cdot(e_0,e_1,\ldots,e_n)\cdot(p_0,x_1,\ldots,x_n)$.

The second case is when $\vec{v}=\vec{v}_1\cdot(\widetilde{p}_0,x_1,\ldots,x_n)\cdot(\widetilde{e}_0,e_1,\ldots,e_n)$. Let us denote $\widetilde{\pi}=(\widetilde{p}_0,x_1,\ldots,x_n)$ and $\eta=(\widetilde{e}_0,e_1,\ldots,e_n)$. Take $\vec{w}\in \mathit{threads}_0(\vec{v})$. It is of the form $\vec{w}_1\cdot e_0$ with $\vec{w}_1\in \mathit{threads}_0(\vec{v}_1\cdot\widetilde{\pi})$ and $(p_0,e_0)\in\widetilde{e}_0$ where $p_0=\mathrm{last}(\vec{w}_1)$. By the induction hypothesis we have that $\mathit{img}(\vec{v}_1\widetilde{\pi},\vec{w}_1)$ is of the form $\vec{u}\cdot(p_0,x_1,\ldots,x_n)$. Let $(e'_0,e'_1,\ldots,e'_n)$ be the reply of the distributed strategy $\langle\sigma_0,\sigma_1,\ldots,\sigma_n\rangle$ in $\mathcal{G}$ in the position $\vec{u}\cdot(p_0,x_1,\ldots,x_n)$. We put $\mathit{img}(\vec{v},\vec{w})=\vec{u}\cdot(p_0,x_1,\ldots,x_n)\cdot(e'_0,e'_1,\ldots,e'_n)$.

We need to show that $\mathit{img}(\vec{v},\vec{w})$ satisfies the required properties. By definition it is a play respecting the distributed strategy. For the third property we need to show that $e_i=e'_i$ for $i=1,\ldots,n$. If $x_i\in E_i$ then $e'_i=e_i$ by the definition of distributed game. It remains to deal with the case when $x_i\in P_i$ and $i\geq 1$. We have:

$$e_i=\widetilde{\sigma}_i(\mathit{view}_i(\vec{v}_1\cdot\widetilde{\pi}))=\sigma_i(\mathit{view}_i(\vec{v}_1\cdot\widetilde{\pi}))=\sigma_i(\mathit{view}_i(\vec{u}\cdot(p_0,x_1,\ldots,x_n)))=e'_i$$

The third equality in the chain above follows from the induction assumption as $\vec{u}\cdot(p_0,x_1,\ldots,x_n)=\mathit{img}(\vec{v}_1\widetilde{\pi},\vec{w}_1)$.

Finally we need to check that $\mathit{img}(\vec{v},\vec{w})$ satisfies the second condition, i.e., that $e_0=e'_0$. We know that $(p_0,e_0)\in\widetilde{\sigma}_0(\mathit{view}_0(\vec{v}_1)\cdot\widetilde{p}_0)$. By definition of $\widetilde{\sigma}_0$ it means that there is $\vec{w}_2\in \mathit{threads}(\mathit{view}_0(\vec{v}_1)\cdot\widetilde{p}_0)$ such that $\mathrm{last}(\vec{w}_2)=p_0$ and $\sigma_0(\vec{w}_2)=e_0$. We want to show that $\sigma_0(\vec{w}_1)=e_0$. As σ_0 is a semi-positional strategy it is enough to show that $\mathit{view}_i(\mathit{path}(\vec{w}_1))=\mathit{view}_i(\mathit{path}(\vec{w}_2))$ for $i=1,\ldots,n$.

Observe that $\mathit{path}(\vec{w}_1)=\mathit{img}(\vec{v}_1\cdot\widetilde{\pi},\vec{w}_1)$. This is because, by induction hypothesis, $\mathit{view}_0(\mathit{img}(\vec{v}_1\cdot\widetilde{\pi},\vec{w}_1)=\vec{w}_1$ and $\mathit{path}(\vec{w}_1)$ is the unique path with this property. Then we have

$$\begin{aligned}\mathit{view}_i(\mathit{path}(\vec{w}_1))=\mathit{view}_i(\mathit{img}(\vec{v}_1\cdot\widetilde{\pi},\vec{w}_1))=&\\ \mathit{view}_i(\mathit{img}(\vec{v}_1\cdot\widetilde{\pi},\vec{w}_2))=\mathit{view}_i(\mathit{path}(\vec{w}_2)).&\end{aligned}$$

The second equality is by the third property of $img(\vec{v}, \vec{w})$ function. $\square$

Lemma 20 If there is a distributed winning strategy in $\widetilde{\mathcal{G}}$ from a position $(\{(p, e_0)\}, e_1, \ldots, e_n)$ then there is one in $\mathcal{G}$ from $(e_0, e_1, \ldots, e_n)$.

Proof
Suppose that there is a distributed winning strategy $\widetilde{\sigma}$ in $\widetilde{\mathcal{G}}$. We define strategies in G. Once again the strategies for players $1, \ldots, n$ do not change.

The main observation is that $\mathit{threads}_0(\vec{u}_1) \cap \mathit{threads}_0(\vec{u}_2) = \emptyset$ for any two different $\vec{u}_1, \vec{u}_2 \in \widetilde{\mathcal{G}}(\widetilde{\sigma}_0, \widetilde{\sigma}_1, \ldots, \widetilde{\sigma}_n)$. Indeed take any two such paths $\vec{u}_1$ and $\vec{u}_2$. We have two cases:

Case 1. $\vec{u}_1 = \vec{u}' \cdot \eta \cdot \pi_1 \cdot \vec{w}_1$ and $\vec{u}_2 = \vec{u}' \cdot \eta \cdot \pi_2 \cdot \vec{w}_2$ such that $\pi_1 \neq \pi_2$. We show that $\pi_1 \cap \pi_2 = \emptyset$. Indeed, if $(e_0, p_0) \in \pi_1 \cap \pi_2$ then we would have both $(e_0, e_1, \ldots, e_n) \to (p_0, x_1, \ldots, x_n)$ and $(e_0, e_1, \ldots, e_n) \to (p_0, x'_1, \ldots, x'_n)$. By determinacy of G we have $x_i = x'_i$ for $i = 1, \ldots, n$. But then $\pi_1 = \pi_2$, a contradiction.

Case 2. $\vec{u}_1$ is a proper prefix of $\vec{u}_2$. By the assumption that $\mathcal{G}$ has no 0-delays we have that all threads in $\mathit{threads}_0(\vec{u}_1)$ have the same length as $\vec{u}_1$ and similarly for $\vec{u}_2$.

By the above we know that each path $\vec{v} \in (E_0 P_0)^+$ determines a path $\vec{u} \in \widetilde{\mathcal{G}}(\widetilde{\sigma}_0, \widetilde{\sigma}_1, \ldots, \widetilde{\sigma}_n)$ such that $\vec{v} \in \mathit{threads}_0(\vec{u})$. Let us denote this path by $\widetilde{path}(\vec{v})$.

From the strategy $\widetilde{\sigma}_0 : (\widetilde{E}_0 \times \widetilde{P}_0) \to \widetilde{E}_0$ we now define strategy σ_0:

- $\sigma_0(\vec{v} \cdot p_0) = e_0$ where $(p_0, e_0) \in \widetilde{\sigma}_0(\mathit{view}_0(\widetilde{path}(\vec{v} \cdot p_0)))$.

For every finite play $\vec{v}$ respecting $\langle \sigma_0, \ldots, \sigma_n \rangle$ in $\mathcal{G}$ we define the path $\widetilde{img}(\vec{v})$ in $\widetilde{\mathcal{G}}$ such that:

- $\mathit{view}_0(\vec{v}) \in \mathit{threads}_0(\widetilde{img}(\vec{v}))$,
- $\mathit{view}_i(\vec{v}) = \mathit{view}_i(\widetilde{img}(\vec{v}))$ for all $i = 1, \ldots, n$,
- $\widetilde{img}(\vec{v})$ respects $\langle \widetilde{\sigma}_0, \ldots, \widetilde{\sigma}_n \rangle$.

We define $\widetilde{img}(\vec{v})$ by induction on the length of $\vec{v}$.

For the induction step we have two cases.

The first case is when $\vec{v} = \vec{v}_1 \cdot (e_0, e_1, \ldots, e_n) \cdot (p_0, x_1, \ldots, x_n)$. In this case $\widetilde{img}(\vec{v}_1(e_0, e_1, \ldots, e_n))$ is of the form $\vec{u}(\widetilde{e}_0, e_1, \ldots, e_n)$. We have that in $\widetilde{\mathcal{G}}$ a transition $(\widetilde{e}_0, , e_1, \ldots, e_n) \to (\widetilde{p}_0, x_1, \ldots, x_n)$ with $(e_0, p_0) \in \widetilde{p}_0$. We can define $\widetilde{img}(\vec{v}) = \vec{u}(\widetilde{e}_0, e_1, \ldots, e_n)(\widetilde{p}_0, x_1, \ldots, x_n)$.

The second case is when $\vec{v} = \vec{v}_1 \cdot (p_0, x_1, \ldots, x_n) \cdot (e_0, e_1, \ldots, e_n)$. In this case $\widetilde{img}(\vec{v}_1(p_0, x_1, \ldots, x_n))$ is of the form $\vec{u}(\widetilde{p}_0, x_1, \ldots, x_n)$. Consider position $(\widetilde{e}_0, e'_1, \ldots, e'_n)$ which is the response of the strategy $\langle \widetilde{\sigma}_0, \ldots, \widetilde{\sigma}_n \rangle$ in the position $\vec{u}(\widetilde{p}_0, x_1, \ldots, x_n)$. We want to show that $(p_0, e_0) \in \widetilde{e}_0$ and $e_i = e'_i$ for all $i = 1, \ldots, n$ such that $x_i \in P_i$ (if $x_i \in E_i$ then $x_i = e_i = e'_i$).

For $i = 1, \dots, n$, if $x_i \in P_i$, we have

$$e'_i = \widetilde{\sigma}_i(view_i(\vec{u}) \cdot x_i) = \sigma_i(view_i(\vec{u}) \cdot x_i) = \sigma_i(view_i(\vec{v}_1) \cdot x_i) = e_i$$

For the case of $i = 0$ we have by definitions that

$$\widetilde{e}_0 = \widetilde{\sigma}_0(view_0(\widetilde{img}(\vec{v}_1(p_0, x_1, \dots, x_n)))) \text{ and } e_0 = \sigma_0(view_0(\vec{v}_1 \cdot (p_0, x_1, \dots, x_n)))$$

By the definition of σ_0 we have $(p_0, e_0) \in \widetilde{\sigma}_0(view(\widetilde{path}(view_0(\vec{v}_1) \cdot p_0)))$. We are done as $\widetilde{path}(view_0(\vec{v}_1) \cdot p_0)) = \widetilde{img}(\vec{v}_1(p_0, x_1, \dots, x_n))$. This last equality follows because $view_0(\widetilde{img}(\vec{v}_1(p_0, x_1, \dots, x_n))) = view_0(v_1) \cdot p_1$ and $\widetilde{path}(view_0(\vec{v}_1) \cdot p_0))$ is the unique path with this property. □

5.2 II-gluable games

In this section we prove Theorem 13 for II-gluable games. Fix such a game $\mathcal{G} = \langle P, E, T, Acc \rangle$ constructed from $n+1$ local games $G_0, \dots, G_n$ where $G_i = \langle P_i, E_i, T_i \rangle$ for $i = 0, \dots, n$.

Lemma 21 If there is a distributed winning strategy in $\mathcal{G}$ from $(e_0, e_1, \dots, e_n)$ then there is one in $\widetilde{\mathcal{G}}$ from $(\{((p, e_0)\}, e_1, \dots, e_n)$ for any $p \in P_0$.

Proof
By assumption, there are strategies $\sigma_i : (E_i \times P_i)^* \to E_i$ for all $i = 0, \dots, n$. We define strategies in $\widetilde{G}$. For $i = 1, \dots, n$ the strategies do not change, i.e., $\widetilde{\sigma}_i = \sigma_i$. For a moment take arbitrary strategy for $\widetilde{\sigma}_0$. Consider a play $\vec{v}$ in $\widetilde{G}$ respecting the strategy $\langle \widetilde{\sigma}_0, \dots, \widetilde{\sigma}_n \rangle$. Directly from the definition and the fact that P0 does not influence the moves of other players we get that there is a play respecting $\langle \sigma_0, \dots, \sigma_n \rangle$ in $\mathcal{G}$ with the same as $\vec{v}$ projection on components $1, \dots, n$. Hence, for every strategy $\widetilde{\sigma}_0$, the distributed strategy guarantees that for every play respecting the strategy the projection on components $1, \dots, n$ is accepted. In the rest of the proof we are going to define the strategy $\widetilde{\sigma}_0$, so that the 0 component is also accepted, i.e., satisfies the parity condition given by Ω.

We will start by defining an ordering on sequences from $(E_0 \times P_0)^*$. Recall that the winning condition on 0-th component is given by a parity function $\Omega : (E_0 \cup P_0) \to \mathbb{N}$. For a finite or infinite sequence u over $(E_0 \cup P_0)$ let $mp(u)$ be the smallest priority appearing in u, i.e., $mp(u) = \min_{i=1,2,\dots} \Omega(u_i)$. Let $\preccurlyeq$ be the ordering on natural numbers such that all odd numbers are smaller than all even numbers, on odd numbers $\preccurlyeq$ is the standard ordering and on even numbers $\preccurlyeq$ is the reverse of the standard ordering. For example, $3 \preccurlyeq 5$ and $4 \preccurlyeq 2$.

We define a relation $u \preccurlyeq v$ on sequences $u, v \in (E_0 \times P_0)^*$; we use the same symbol for this relation on strings and the above defined relation on natural numbers. We defined the strict version of this ordering. If u and v differ on the first position we put $u \prec v$ if

- $mp(u) \prec mp(v)$; or
- $mp(u) = mp(v) = p$, p is odd (even) the first (last) occurrence of p in u is sooner than its first (last) occurrence in v; or
- not the previous cases and the first position in u is smaller than the first position in v in some arbitrary but fixed ordering on positions.

For u and v having a common prefix we put $u \prec v$ if $u' \prec v'$ where u', v' are the words obtained by erasing the common prefix from u and v respectively. By examining all possible cases we get:

Claim 21.1 $\preccurlyeq$ is a linear ordering.

For a player's position p_0 and a sequence $\vec{v} \in (\widetilde{E}_0 \times \widetilde{P}_0)^*$ we define $hist_{\vec{v}}(p_0)$ to be the smallest in $\preccurlyeq$ ordering thread in $\vec{v}$ finishing with p_0 (such a thread exists if for some e the pair (e, p_0) appears in the last element of $\vec{v}$).

We put

$$\widetilde{\sigma}_0(\vec{v} \cdot \widetilde{p}_0) = \{(p_0, e_0) : \exists (e', p_0) \in \widetilde{p}_0.\ e_0 = \sigma_0(hist_{\vec{v}}(p_0))\}$$

For a thread u of $\vec{v} \in (\widetilde{E}_0 \times \widetilde{P}_0)^\omega$ we define the thread $hist_{\vec{v}}(u)$ to be $hist_{\vec{w}}(p)$ where $\vec{w}$ is the prefix of $\vec{v}$ of the same length as u and p is the last element of u. We call a thread $u \in (E_0 P_0)^\omega$ *historic* iff $hist_{\vec{v}}(u[0,m]) = u[0,m]$ for all m.

Take a play $\vec{v}$ respecting $\langle \widetilde{\sigma}_0, \widetilde{\sigma}_1, \ldots, \widetilde{\sigma}_n \rangle$. We want to show that $view_0(\vec{v})$ is accepted, i.e., satisfies the parity condition given by Ω. The first observation is that all historic threads in $\vec{v}$ are plays according to the strategy σ_0. We get:

Claim 21.2 All historic threads in $view_0(\vec{v})$ are accepted.

In the rest of the proof we show that if all historic threads are accepted than all threads are accepted. For a contradiction suppose that there is a non-accepting thread u in $view_0(\vec{v})$. The following claim shows that we can associate to u a historic thread that is not accepted.

Claim 21.3 For every m there exists $n \geq m$ such that $hist(u[0,n])[0,m] = hist(u[0,n'])[0,m]$ for all $n' > n$.

Proof
Consider the sequence $mp(hist(u[0,n]))$. It is not increasing, hence it must be constant, say equal to q for all n bigger than some n_0. Suppose that q is even; the case when q is odd is similar. Let i_n be the position of the last appearance of q in $u[0,n]$. We have that the sequence i_n is not increasing, hence after some $n_1 > n_0$ it is fixed. Take the smallest thread that leads to a position of priority q in i_{n_1}-th element of $\vec{v}$. This thread is a prefix of $hist(u[0,n'])$ for all $n' > n_1$. We can repeat the reasoning to fix more positions. $\square$

Let hu be the limit of $hist(u[0,n])$, i.e., a thread such that for every m, $hu[0,m] = hist(u[0,n])[0,m]$ for almost all n. It is easy to check that hu is a historic thread. Let $g_n = hist(u[0,n])u[n+1,\infty]$ for every $n = 0,1,\ldots$ Observe that none of g_n is accepted. We will show that hu is not accepted too.

Claim 21.3 says that to calculate a prefix of hu it is enough to take a (maybe longer) prefix of u.

To show that hu is not accepted we will define a sequence of positions $m_0, m_1, \ldots$. Then we will show that the smallest number appearing infinitely often in the sequence $\{mp(hu[m_l+1, m_{l+1}])\}_{l=1,2,\ldots}$ is odd.

We start with $m_0 = 0$, and n_0 being a position such that $g_{n_0}[0,0] = hu[0,0]$. Suppose that we have already defined m_l and n_l having the property that $g_{n_l}[0, m_l] = hu[0, m_l]$. Let $q_l = mp(g_l[m_l+1, \infty])$. If q_l is even then we take m_{l+1} to be the last position when q_l occurs on g_l. If q_l is odd then let m_{l+1} be the first position when q_l occurs on g_l. We take n_{l+1} such that $g_{n_{l+1}}[0, m_{l+1}] = hu[0, m_{l+1}]$. Let $p_l = mp(g_{n_{l+1}}[m_l+1, m_{l+1}])$; we have $p_l = mp(hu[m_l+1, m_{l+1}])$. Let $q_{l+1} = mp(g_{l+1}[m_{l+1}+1, \infty])$

Claim 21.4 If q_l is even then one of the following holds:

E1 p_l, q_{l+1} are even, $q_{l+1} > q_l$ and $p_l \geq q_l$;

E2 p_l is even, q_{l+1} is odd and $q_{l+1} < p$ or $q_{l+1} > q$ and $p_l \geq q_l$;

E3 p_l is odd, q_{l+1} is even and $q_{l+1} > \min(p_l, q_l)$;

E4 p_l, q_{l+1} are odd.

Proof
To see this observe that $g_{n_{l+1}}$ and g_{n_l} are the same up to position m_l, hence $g_{n_{l+1}}[m_l+1, n_{l+1}] \preccurlyeq g_{n_l}[m_l+1, n_{l+1}]$ by definition of $\preccurlyeq$. We know that $mp(g_{n_l}[m_l+1, n_{l+1}]) = q$ and the last appearance of q is in m_{l+1}. Let $r = mp(g_{n_{l+1}}[m_l+1, j])$. If $r > q_l$ then $q_{l+1} > q_l$ and $p_l > q_l$ hence one of E1, E2, E3, E4 holds. If $r = q_l$ then $p_l = q_l$ and $q_{l+1} > q_l$ as all occurrences of q_l in $g_{n_{l+1}}[m_l+1, n_{l+1}]$ are before m_{l+1}; so E1 holds. The last case is when $r < q_l$. We have that r is odd. If p_l is even then $q_{l+1} = r$ and $q_{l+1} < p_l$, so E2 holds. If p_l is odd and q_{l+1} is even then $p_l = r$ and $q_{l+1} > r$; so E3 holds. $\square$

Claim 21.5 If q_l is odd then one of the following holds:

O1 p_l is odd and $p_l \leq q_l$, $p_l \leq q_{l+1}$;

O2 q_{l+1} is odd $p_l > q_l$ and $q_{l+1} < q_l$.

Proof
To see this observe that $r = mp(g_{n_{l+1}}[m_l+1, n_{l+1}])$ is odd and $r \leq q_l$. If r appears in $g_{n_{l+1}}[m_l+1, n_{l+1}]$ then O1 holds. If not then $q_{l+1} = r$. By definition of $\preccurlyeq$ we have $r < q_l$ so O2 holds. $\square$

Claim 21.6 If $q_l > q_{l+1}$ then q_{l+1} is odd or $p_l \leq q_{l+1}$ and p_l is odd.

Proof
We check the possibilities one by one. Property E1 cannot hold. In case E2, E4, O2 we know that q_{l+1} is odd. For the properties E3 and O1 we have that either $q_{l+1} > \min(q_l, p_l)$ hence $q_{l+1} > p_l$. $\square$

Claim 21.7 If $q_l < q_{l+1}$ then either p_l is odd or $p_l \geq q_l$.

Proof
Property O2 is impossible. In case of properties E3, E4 and O1 we have that p_l is odd. Property E1 implies $p_l \geq q_l$. If we have property E2 then either $p_l > q_{l+1} \geq q_l$ or $p_l \geq q_l$. $\square$

Claim 21.8 If q_l is odd then there is $k > l$ such that $p_k = \min(p_l, \ldots, p_k)$, p_k is odd, $p_k \leq q_l$ and $p_k \leq q_{k+1}$.

Proof
If property O1 holds then we know that p_l is odd, $p_l < q_l$ and $p_l \leq q_{l+1}$. If not then 02 holds so $p_l > q_l$ and $q_{l+1} < q_l$. We repeat the argument with q_{l+1}. After finitely many steps the property O1 should hold for some k with $q_k < q_l$. We conclude as we have that $p_l, \ldots, p_{k-1} > q_k$. $\square$

Claim 21.9 If $q_l > q_{l+1}$ then there is $k > l$ such that $p_k = \min(p_l, \ldots, p_k) \leq q_{l+1}$ is odd and $q_{k+1} \geq p_k$.

Proof
Using Claim 21.6 we see that we can have p_l odd and $p \leq q_{l+1}$. In this case we take $l = k$. Otherwise from the same claim we know that q_{l+1} is odd. Using Claim 21.8 we get the desired k. $\square$

Claim 21.10 The smallest priority appearing infinitely often in the sequence $\{p_l\}_{l=0,1,\ldots}$ is odd.

Proof
Consider the sequence $\{q_l\}_{l=0,1,\ldots}$. As we can never have $q_l = q_{l+1}$ then this sequence consists of strictly increasing and strictly decreasing finite intervals. Let r be the minimal priority appearing infinitely often on the sequence $\{q_l\}_{l=0,1,\ldots}$. Whenever $q_i = r$ then $q_{i-1} > q_i$ so Claim 21.6 guarantees that p_i is odd and $p_i < r$. Let n_0 be a position such that $q_l \geq r$ for $l \geq n_0$. For such l we have, by Claim 21.7 that whenever $q_l < q_{l+1}$ then $p_l \geq q_l \geq r$. Whenever $q_l > q_{l+1}$ then by Claim 21.9 we can find k such that $p_k = \min(p_l, \ldots, p_k)$ and p_k is odd. Summarizing, on the sequence $\{p_l\}_{l=n_0,n_0+1,\ldots}$ we have infinitely many times an odd priority smaller than r. Each priority p_l corresponding to increasing position (i.e. $q_l < q_{l+1}$) is not smaller than r. Each priority p_l corresponding to a decreasing position (i.e. $q_l > q_{l+1}$) is bounded from below by some odd priority p_k for $k > l$. Hence the smallest priority appearing infinitely often on $\{p_l\}_{l=0,1,\ldots}$ is odd. $\square$

$\square$

Lemma 22 If there is a distributed winning strategy in $\widetilde{\mathcal{G}}$ from a position $(\{(p, e_0)\}, e_1, \ldots, e_n)$ then there is one in $\mathcal{G}$ from $(e_0, e_1, \ldots, e_n)$.

Proof
Suppose that there is a distributed winning strategy $\widetilde{\sigma}$ in $\widetilde{\mathcal{G}}$. We define strategies in G. Once again the strategies for players $1, \ldots, n$ do not change.

Let us defined a function $pm : \widetilde{E}_0 \times E_0 \times P_0 \to \widetilde{P}_0$ by $(e, p) \in pm(S, e_0, p_0)$ if $\exists p'.\ (p', e) \in S \wedge (e, p) \sim (e_0, p_0)$; recall that $\sim$ is the equivalence relation required to exist by II-gluability conditions. These are the moves that are possible from S knowing that the move from e_0 to p_0 was made by the environment.

Given a finite or infinite sequence $u = e^1 p^1 e^2 p^2 \ldots$ of positions for player 0 we can define a sequence $pump(u) = S_e^1 S_p^1 S_e^2 S_p^2 \ldots$ by

$$S_e^1 = \{(p, e^1)\}, \quad S_p^i = pm(S_e^i, e^i, p^i), \quad S_e^{i+1} = \widetilde{\sigma}_0(S_p^i)$$

We define the strategy σ_0. Let $u \in (E_0 \times P_0)^*$ with $p = \mathrm{last}(u)$, we put

$$\sigma_0(u) = e \quad \text{such that} \quad (p, e) \in \widetilde{\sigma}_0(pump(u))$$

Let $\vec{v}$ be a play respecting $\langle \sigma_0, \sigma_1, \ldots, \sigma_n \rangle$ in $\mathcal{G}$. We construct a play $\vec{w}$ respecting $\langle \widetilde{\sigma}_0, \widetilde{\sigma}_1, \ldots, \widetilde{\sigma}_n \rangle$ in $\widetilde{\mathcal{G}}$ such that:

- $view_0(\vec{v})$ is a thread in $view_0(\vec{w})$,
- $view_i(\vec{v}) = view_i(\vec{w})$ for $i = 1, \ldots, n$.

If we manage to do this then it will show that $\langle \sigma_0, \sigma_1, \ldots, \sigma_n \rangle$ is winning.

The initial position of $\vec{v}$ is some $\langle e_0^0, e_1^0, \ldots, e_n^0 \rangle$. We can fix some arbitrary p_0^{-1} and take $\langle \{(p_0^{-1}, e_0^0)\}, e_1^0, \ldots, e_n^0 \rangle$ as the initial position of $\vec{w}$.

Suppose that we have constructed $\vec{w}$ up to the position i. Suppose that the i-th positions in $\vec{v}$ and $\vec{w}$ are respectively:

$$\langle e_0, e_1, \ldots, e_n \rangle \qquad \langle S_e, e_1, \ldots, e_n \rangle$$

We also assume that

- $view_i(\vec{v}[0, i]) = view_i(\vec{w}[0, i])$;
- $view_0(\vec{w}[0, i]) = pump(view_0(\vec{v}))$ and moreover $view_0(\vec{v}[0, i])$ is a thread in $view_0(\vec{w}[0, i])$.

The prolongations of $\vec{v}[0, i]$ and $\vec{w}[0, i]$ look as follows:

$$\begin{array}{lll}
\langle e_0, e_1, \ldots, e_n \rangle & \langle S_e, e_1, \ldots, e_n \rangle & \\
\langle p_0, x_1, \ldots, x_n \rangle & \langle S_p, x_1, \ldots, x_n \rangle & \text{with } (e_0, p_0) \in S_p \\
\langle e_0', e_1'', \ldots, e_n'' \rangle & \langle S_e', e_1', \ldots, e_n' \rangle & \text{with } (p_0, e_0') \in S_e'
\end{array}$$

Let us explain this sequence. Element $\langle p_0, x_1, \ldots, x_n\rangle$ is just the $(i+1)$-th position of $\vec{v}$. By induction assumption $(p', e_0) \in S_e$ hence the environment can move from $\langle S_e, e_1, \ldots, e_n\rangle$ to $\langle S_p, x_1, \ldots, x_n\rangle$ where $S_p = pm(S_e, e_0, p_0)$. We have $(e_0, p_0) \in S_p$. The element $\langle S'_e, e'_1, \ldots, e'_n\rangle$ is the reply to $\langle S_p, x_1, \ldots, x_n\rangle$ according to the strategy $\langle \widetilde{\sigma}_0, \widetilde{\sigma}_1, \ldots, \widetilde{\sigma}_n\rangle$. Similarly for $\langle e'_0, e''_1, \ldots, e''_n\rangle$ and $\langle \sigma_0, \sigma_1, \ldots, \sigma_n\rangle$. By the definition of σ_0, $(p_0, e'_0) \in \widetilde{\sigma}_0(pump(view_0(\vec{v}[0,i]p_0)))$. We have that $e''_i = e'_i$ and $(p_0, e'_0) \in S'_e$ because S'_e is the last element of $pump(view_0(\vec{v}[0,i]p))$. Hence the invariants hold again and we are done. $\square$

6 Synthesis for pipeline architecture

In this section we show how to encode the synthesis problem for pipeline architectures [27] into the game framework. Then we will use the theorems from preceding sections to solve the resulting games. We will consider branching specifications [13]. For this we need to introduce automata working on trees bounded but varying degrees.

Automata Given a finite set Σ, a deterministic Σ-tree is a prefix-closed subset Υ of Σ^*. The elements of Υ are called *nodes*. The empty word ε is called the *root node*. For each node $x \in \Upsilon$, and letter $b \in \Sigma$, word $x \cdot b$ is a b-successor of x. A *path* (finite or infinite) in a tree Υ is a sequence of nodes $x_0, x_1, \ldots$ such that $x_0 = \varepsilon$ and x_{i+1} is a successor of x_i for every i.

We define an automaton running on deterministic Σ-trees:

$$\mathcal{A} = \langle Q, Q^{\exists}, Q^{\forall}, \Sigma, q^0, \delta : Q \to \mathcal{P}(Moves(\Sigma, Q)), Acc \subseteq (Q^{\forall})^{\omega}\rangle$$

where $(Q^{\exists}, Q^{\forall})$ is a partition of Q, Acc is a regular set of sequences, and

$$Moves(\Sigma, Q) = ((\Sigma \cup \{\varepsilon\}) \times Q) \cup (\Sigma \times \{\to, \not\to\})$$

The intuition is that $(Q^{\exists}, Q^{\forall})$ is a partition of Q into the set of universal and existential states. The transition function assigns to each state a set of possible moves. A move (b, q) means go into direction b (if it exists) and change the state to q. If $b = \varepsilon$ then the automaton stays in the node and just changes the state. A move $(b, \to)$ "tests" for the existence of b transition. The automaton rejects if he decides to execute this move and there is no b transition from the current node, otherwise this part of the run of automaton accepts. Similarly $(b, \not\to)$ test for absence of a b transition.

An automaton as above is *bipartite nondeterministic* if $\delta(q) \subseteq \{\varepsilon\} \times Q^{\forall}$ for every $q \in Q^{\exists}$; and $\delta(q) \subseteq Moves(\Sigma, Q^{\exists}) \setminus (\{\varepsilon\} \times Q)$ for every $q \in Q^{\forall}$. In the case of $q \in Q^{\forall}$ we additionally require that if $(a_1, q_1), (a_2, q_2) \in \delta(q)$, with $q_1, q_2 \in Q$, and $a_1 = a_2$ then $q_1 = q_2$. Moreover, we assume that the initial state q^0 is universal (i.e., in $Q^{\forall}$).

We refer the reader to [4] for the semantics and properties of these automata. Here we will give the semantics only for bipartite nondeterministic automata. We quote the fact from [4] that permits us to do this.

Fact 23 Every automaton is equivalent to a bipartite nondeterministic parity automaton. The automata accept precisely MSOL definable languages.

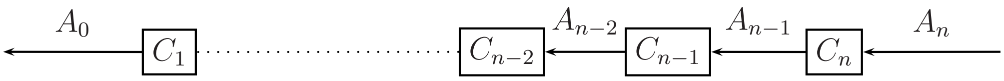
For bipartite nondeterministic automata the notion of run is slightly easier to define than for the general case. Given an nondeterministic bipartite automaton $\mathcal{A}$, a run of $\mathcal{A}$ on a Σ-tree Υ is a function $r : \Upsilon \to Q^{\forall}$ such that $r(\varepsilon) = q^0$ and for every node x of Υ:

- if $(b, \to) \in \delta(r(x))$ then x has a b successor;
- if $(b, \not\to) \in \delta(r(x))$ then x does not have a b successor;
- if $(b, q^{\exists}) \in \delta(r(x))$ then $r(x \cdot b) \in \delta(q^{\exists})$ provided $x \cdot b$ exists.

A run of the automaton $\mathcal{A}$ on Υ is *accepting* if for all the infinite paths $v_0, v_1, \ldots$ in Υ, the sequence $r(v_0), r(v_1), \ldots$ is in Acc. An Σ-tree is *accepted* by the automaton $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on Υ.

Note that only the universal states of the automaton are used for labeling and the consequent definition for acceptance. Since $\mathcal{A}$ is bipartite, this does not result in any loss of generality.

Pipeline 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 $f_i(a_0 a_1 \ldots a_j) = b_j$ for all j.

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

Let $\Sigma = A_n \cdot A_{n-1} \cdots A_0$. Observe that an execution, which is a string from $(A_n A_{n-1} \cdots A_0)^\omega$, can be considered as a string over Σ by identifying a block of letters from $(A_n A_{n-1} \cdots A_0)$ with one letter from Σ. A controller P defines a Σ-tree $\Upsilon(P)$ of Σ-labeled paths which are the executions respecting P.

The *pipeline synthesis problem* is: given a pipeline over alphabets $A_0, \ldots, A_n$ and a nondeterministic parity tree automaton $\mathcal{A}$ over the alphabet $\Sigma = A_n \cdot A_{n-1} \cdots A_0$, find a pipeline controller $P = \langle f_1, \ldots, f_n \rangle$ such that $\Upsilon(P) \in L(\mathcal{A})$.

$\mathsf{Encoding}$ 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 and G_i the role of the i-th component C_i. The game G_0 is:

- $P_0 = Q^\exists \times \Sigma$; $E_0 = Q^\forall$;
- $(q^\exists, w) \to q^\forall \in T_0$ if $(\varepsilon, q^\forall) \in \delta(q^\exists)$, and all possible transitions from environment to player positions.

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 the complete set of transitions between these sets of positions.

To define global transitions from an environment position $((q^\forall, a), f_1, \dots, f_n)$ let us introduce some notation. For a letter $a_n \in A_n$ let $w(a_n) = a_n a_{n-1} \dots a_0$ be the word such that $a_{i-1} = f_i(a_i)$. The word $w(a_n)$ is a letter of Σ. We have the following:

- $(q^\forall, f_1, \dots, f_n) \to (\bot, a_1, \dots, a_n)$ if $(w(a_n), \not\to) \in \delta(q^\forall)$ for some $a_n \in A$ or $(v, \to) \in \delta(q^\forall)$ for some v not of the form $w(a_n)$.
- If not the previous case then $(q^\forall, f_0, \dots, f_n) \to ((q^\exists, w(a_n)), a_1, \dots, a_n)$ where $q^\exists = \delta(q^\forall, w(a_n))$ and $a_n \dots a_0 = w(a_n)$.

The winning condition Acc is defined by the parity condition in the automaton $\mathcal{A}$. The initial position is $\pi_0 = (q^0, a_1, \dots, a_n)$ for some arbitrarily chosen letters $a_1, \dots, a_n$.

Lemma 24 Given a pipeline on alphabets $A_0, \dots, A_n$ and a nondeterministic bipartite parity automaton $\mathcal{A}$ over Σ, the pipeline synthesis problem is solvable iff there is a distributed winning strategy in the pipeline game $\mathcal{G}$ from η_0.

Proof
($\Longrightarrow$) Let $P = (f_1, \dots, f_n)$ be a controller such that $\Upsilon(P)$ is accepted by $\mathcal{A}$. Let $r : \Upsilon(P) \to Q^\forall$ be an accepting run of $\mathcal{A}$ on $\Upsilon(P)$.

We define distributed strategy $\sigma = \langle \sigma_0, \sigma_1, \dots, \sigma_n\rangle$. For $i = 1, \dots, n$ strategy σ_i is of type $A_i^* \to (A_i \to A_{i-1})$. We put $\sigma_i(\vec{v})(a_n) = f(\vec{v} \cdot a_n)$.

Strategy σ_0 is of type $P_0^* \to Q^\forall$. We put $\sigma_0(\vec{v}) = r(path(\vec{v}))$, where $path((q_0^\exists, w_0) \dots (q_n^\exists, w_n)) = w_1 \dots w_n$.

It is easy to see that for every play $\vec{v}$ respecting σ, then there is a path $\vec{u}$ of $\Upsilon(P)$ such that the sequence of universal states in both paths is the same. Since r is accepting P, $\vec{u}$ satisfies the parity condition. Therefore, the play $\vec{v}$ also satisfies the parity condition and is winning.
($\Longleftarrow$) Let $(\sigma_0, \sigma_1, \dots, \sigma_n)$ be a distributed winning strategy in the pipeline game $\mathcal{G}$. We put $f_i(v_1 \dots v_n v_{n+1}) = \sigma_i(v_1 \dots v_n)(v_{n+1})$. We define the run on $\Upsilon(P)$ by $r(w_1 \dots w_k) = \sigma_0((q_0^\exists, w_0) \dots (q_n^\exists, w_n))$ where $(q_i^\exists, w_i)$ are uniquely defined by: $(q_i^\exists, w_i) \in \delta(r(w_1 \dots w_{i-1}))$. $\square$

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 25 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 26 We call a game $\langle 0,n\rangle$*-proper* if it satisfies the following:

P1a. $\mathcal{G}$ is 0-deterministic,

P1b. P0 has no 0-delays,

P1c. the winning condition is a parity condition on P0;

P2. $\mathcal{G}$ is n-deterministic;

P3. $\mathcal{G}$ is i-sequential for all $i \in \{1, \dots, n\}$.

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 the sequentiality does not say anything about player 0.

Directly from the definitions we obtain.

Fact 27 The pipeline game $\mathcal{G}$ is $\langle 0,n\rangle$-proper.

The following lemmas are easy to check consequences of the constructions for DIVIDE and GLUE operations. They will allow us to solve proper games.

Lemma 28 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.

Proof
If $\mathcal{G}$ is $\langle 0,n\rangle$-proper then it is 0-deterministic and n-deterministic. Hence DIVIDE($\mathcal{G}$) exists. By the definition DIVIDE($\mathcal{G}$) satisfies P1c. The condition P1b also follows directly from the definition and assumption that P1b is true for $\mathcal{G}$.

To check P1a suppose that in DIVIDE($\mathcal{G}$) we have moves from $((e_0, e_n), \overline{e})$ to two different positions $((x_0, x_n), \overline{x})$ and $((x'_0, x'_n), \overline{x'})$. By the definition of division, in $\mathcal{G}$ we have transitions form $(e_0, \overline{e}, e_n)$ to $(x_0, \overline{x}, x_n)$ and to $(x'_0, \overline{x'}, x'_n)$. Since $\mathcal{G}$ is 0-deterministic (by P1a), $x_0 \neq x'_0$ and $x_0, x'_0 \in P_i$. This implies $(x_0, x_n) \neq (x'_0, x'_n)$ and $(x_0, x_n), (x'_0, x'_n) \in \widetilde{P}_i$.

Along the same lines one can verify the condition P3. □

Lemma 29 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.

Proof
Observe that $\mathcal{G}$ is I-gluable because of conditions P1a, P1b and P1c. By definition, GLUE($\mathcal{G}$) satisfies conditions P1b and P1c (by Remark 5).

In order to prove that GLUE($\mathcal{G}$) is 0-deterministic, it suffices to consider the transitions of the form $(\widetilde{e_0}, \overline{e}) \to (\widetilde{p_0}, \overline{x})$ and $(\widetilde{e_0}, \overline{e}) \to (\widetilde{p'_0}, \overline{x'})$ such that $\overline{x} \neq \overline{x'}$. We want to show a stronger property that $\widetilde{p_0} \cap \widetilde{p'_0} = \emptyset$. Assuming conversely, we have some (e_0, p_0) in the intersection. By the definition of $\widetilde{\mathcal{G}}$ we have $(e_0, \overline{e}) \to (p_0, \overline{x})$ and $(e_0, \overline{e}) \to (p_0, \overline{x'})$ in G. But this is impossible since $\mathcal{G}$ is 0-deterministic by P1a.

To show condition P2 we consider a position $(\widetilde{e_0}, \overline{e})$ with transitions to $(\widetilde{p_0}, \overline{x})$ and to $(\widetilde{p'_0}, \overline{x'})$. We need to show that the last elements of $\overline{x}$ and $\overline{x'}$ are different. Denote them by x_n and x'_n respectively. From Corollary 14 we know that $\overline{x} \neq \overline{x'}$. By the n-sequentiality of $\mathcal{G}$ we have that $x_n \neq x'_n$.

For condition P3 consider two transitions:

$$(\widetilde{e_0}, \overline{e}) \to (\widetilde{p_0}, \overline{x}) \quad \text{and} \quad (\widetilde{e'_0}, \overline{e'}) \to (\widetilde{p'_0}, \overline{x'})$$

We want to establish that $x_i \neq x'_i$ whenever $\overline{e}[1,i] = \overline{e'}[1,i]$ but $\overline{x}[1, i-1] \neq \overline{x'}[1, i-1]$. This follows directly from the i-sequentiality of $\mathcal{G}$. $\square$

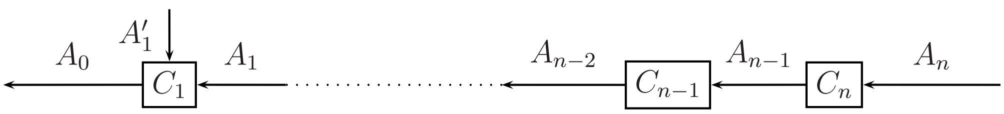
Theorem 30
The synthesis problem in the pipeline is decidable.

Proof
Given a pipeline game of $n+1$ players. By the fact above, it is $\langle 0, n\rangle$-proper. Now, by the preceding lemmas successive application of DIVIDE and GLUE result in a $\langle 0, n-1\rangle$-proper game $\mathcal{G}'$. By theorems 8 and 13, $\mathcal{G}$ has a distributed winning strategy iff $\mathcal{G}'$ has a distributed winning strategy.

By $n-1$ repeated application of DIVIDE and GLUE followed at the end by a DIVIDE operation, we eventually have the game consisting of a single player and the environment with a parity winning condition. A distributed strategy in this game is just the winning strategy for the player. By Theorem1, existence of winning strategy in this game is decidable. $\square$

7 Local specifications

In [17, 16] Madhusudan and Thiagarajan have considered the synthesis problem for distributed architectures and local specifications, i.e., specifications that are conjunctions of requirements on the behaviour of each of the components separately. They have shown that the synthesis problem is decidable only for double flanked pipeline architecture.



The peculiarity of the double flanked pipeline is that the component closest to the end has an additional input from the environment. Otherwise the behaviour of such a pipeline is the same as of the normal one: when the input comes on the channel A_n then the component C_n sends a letter on the channel A_{n-1} and so on up to the component C_1 which not only receives a letter on the channel A_1 but also an additional letter from the environment on the channel A'_1.

The global behaviour of a pipeline is a sequence from $(A_n \cdot A_{n-1} \cdots A_1 \cdot A'_1 \cdot A_0)^\omega$. Let Σ_i denote the alphabet of the component i, i.e., $\Sigma_i = A_i \cup A_{i-1}$ for $i > 1$ and $\Sigma_1 = A_1 \cup A'_1 \cup A_0$. The behaviour of the component i is the projection of the global behaviour on the alphabet Σ_i.

A *local specification* is given by a tuple of regular languages $(L_1, \dots, L_n)$ over respective alphabets. A global behaviour u *satisfies the specification* if for each i the projection of u on Σ_i is in L_i.

Encoding into a game The coding of this synthesis problem into a distributed game is very similar to the coding of a simple pipeline. The game $\mathcal{G}$ will consist of local games G_i: one for each component of the pipeline. Each game G_i, for $i = 2, \dots, n$, is as in the case of simple pipeline:

- $P_i = A_i$, $E_i = (A_i \to A_{i-1})$, and there is a complete set of transitions between P_i and E_i.

For game G_1 the definition is different as the first component takes additional argument and as we want to put a parity acceptance condition on this component. Take a deterministic parity automaton $\mathcal{A}_1 = \langle Q_1, \Sigma_1, q_1^0, \delta_1, \Omega_1 \rangle$ recognizing L_1. We have:

- $P_1 = Q \times A_1 \times A'_1$, $E_1 = Q \times ((A_1 \times A'_1) \to A_0)$ and payer's transitions are restricted to $(q, a_1, a'_1) \to (q, f)$, i.e., player is not allowed to change state.

The global moves of the environment are:

$$((q_1, f_1), f_2 \dots, f_n) \to ((q'_1, a_1, a'_1), a_2, \dots, a_n)$$

where $a_{i-1} = f_i(a_i)$ for $i > 1$, and $q'_1 = \delta_1(q_1, a_1 a'_1 f_1(a_1, a'_1))$. This means that the whole global position is determined by letters a_n and a'_1. The winning condition is that on each component $i = 2, \dots, n$ the sequence $f_i^1, a_i^1, f_i^2, a_i^2, \dots$ is such that the word $a_i^1, f_i^1(a_i^1), a_i^2, f^2(a_i^2), \dots$ is in L_i. For the first component the condition is that the sequence of states satisfies the parity condition of the automaton $\mathcal{A}_1$. Actually we see only every third state of the run of $\mathcal{A}_1$ so we need to assume some special form of $\mathcal{A}_1$ like that only each third state has a significant priority and the rest have some big irrelevant priority.

The argument that the game indeed codes the behaviour of the double flanked pipeline is the same as in the case of simple pipelines. To solve the game $\mathcal{G}$, we first show that we can apply to it the gluing operation over the first component. For this we check II-gluability conditions (cf. 12).

- The moves of other players are not influenced by P1 moves, i.e., having

$$((q, f_1), f_2 \dots, f_n) \to ((q', a_1, a'_1), a_2, \dots, a_n)$$

we also have

$$((q^1, f_1^1), f_2 \dots, f_n) \to ((q^2, a_1, a'_1), a_2, \dots, a_n)$$

for any other (q_1^1, f_1^1) and some appropriate q^2.

- The moves of player 1 are almost context independent. Required equivalence relation on pairs of positions of player 0 is defined by:

$$((q_1^1, f_1^1), (q_1^2, a_1 a'_1)) \sim ((q_1^3, f_1^3), (q_1^4, a_1 b'_1))$$

i.e., the pairs having the same a_1 letter.

- Directly from the definition it follows that player 1 moves at each step and that the winning condition is a conjunction of the parity condition for player 0 and some conditions on other components.

By Theorem 12, there is a distributed strategy in $\mathcal{G}$ iff there is one in $\text{GLUE}_2(\mathcal{G})$. The game $\text{GLUE}_2(\mathcal{G})$ is essentially a pipeline game we have seen before but with the difference that the acceptance condition is not a parity condition on the 0-th component. To have this we just add one component to the game which "observes" all the others. We quickly describe the construction below.

Consider a deterministic parity automaton $\mathcal{A}_0 = \langle Q_0, \Sigma_0, q_0^0, \delta_0, \Omega_0 \rangle$ recognizing the set of winning plays in $\mathcal{G}$. In particular the alphabet Σ_0 of $\mathcal{A}_0$ consist of the global positions in the game $\text{GLUE}_2(\mathcal{G})$. We define G_0 to be the game with: $P_0 = Q_0$, $E_0 = Q_0 \times \{e\}$, where e is just a marker to distinguish positions of player and environment. The player is restricted to move from q to (q, e), i.e., he has no choice.

We now define the global game $\widehat{\mathcal{G}}$ with the components $G_0, \widetilde{G}_1, \dots, \widetilde{G}_n$, where each $\widetilde{G}_i$ is i-th component in th game $\text{GLUE}_2(\mathcal{G})$. The global moves of the environment are: $((q, e), \overline{\eta}) \to (\delta(q, \overline{\eta}\,\overline{\pi}), \overline{\pi})$ if $\overline{\eta} \to \overline{\pi}$ is a global move in the game $\text{GLUE}_2(\mathcal{G})$. By $((q, e), \overline{\eta})$ we mean a $(n+1)$-tuple with the first component (q, e) and the other components as in $\overline{\eta}$. The winning condition is the parity condition on the 0-th component.

We check that $\widehat{\mathcal{G}}$ is a $\langle 0, n \rangle$-proper game (cf. Definition 26). All the conditions but P3 follow directly from the definitions. For P3 we observe that whenever we have a move from $((q, e), S, f_2, \dots, f_n)$ to $(q', S', a_2, \dots, a_n)$ then a_i determines S' and a_j for all $j < i$. By definition $a_{i-1} = f_i(a_i)$, so it remains to consider S'. We have by the definition of the glue operation:

$$\begin{aligned} S' = \{((q, f), (q', a_1, a'_1)) : \exists p \in P_0.\ & (p, (q, f)) \in S \\ & ((q, f), f_2, \dots, f_n) \to ((q', a_1, a'_1), a_2, \dots, a_n)\} \end{aligned}$$

Recall that by definition $q' = \delta_1(q, a_1 a'_1 f(a_1, a'_1))$. As a_1 is determined by a_2 and a'_1 is not influenced by $f_2, \dots, f_n$ we have that S' is determined by S and a_2.

This shows that $\widehat{\mathcal{G}}$ is a $\langle 0, n\rangle$-proper game and we can solve it in the way described in Section 6.

8 Synthesis for communicating machines

In this section we show how to formulate and solve the synthesis problem considered by Madhusudan and Thiagarajan in [18]. In their setting a system consists of several components that can synchronize with each other on common actions; additionally there are environment actions that are local for each component. The problem is to synthesize a controller for each of the components so that the global behaviour satisfies a given specification. In general the problem is undecidable. It becomes decidable if restricted to "trace closed" specifications and a particular class of controllers.

Let us fix a set of processes $\{1, \dots, n\}$. Each process has its alphabet Σ_i of actions which is partitioned into Σ_i^c and Σ_i^e of actions controlled by the process and by the environment respectively. We assume that $\Sigma_i^e \cap \Sigma_j^e = \emptyset$ for $i \neq j$; but we do not assume this for controllable actions. Alphabets Σ_i define the distribution function $\theta : \bigcup \Sigma_i \to \{1, \dots, n\}$ by $\theta(a) = \{i : a \in \Sigma_i\}$. We will use the notation $\Sigma^c = \bigcup_i \Sigma_i^c$ and $\Sigma^e = \bigcup_i \Sigma_i^e$

Consider a tuple $\{\langle S_i^c, S_i^e, \{\to_i^a\}_{a \in \Sigma_i}, s_i^{in}\rangle\}_{i=1,\dots,n}$ of deterministic transition systems, where the transitions on Σ_i^c always go from S_i^c to S_i^e and the transitions on Σ_i^e go from S_i^e to S_i^c. Such a tuple defines a product transition system:

$$\langle \Pi_{i=1,\dots,n}(S_i^c \cup S_i^e), \{\to^a\}_{a \in \Sigma}, s_{in}\rangle$$

with $s_{in} = (s_1^{in}, \dots, s_n^{in})$ and $(s_1, \dots, s_n) \to^a (t_1, \dots, t_n)$ if $s_i \to_i^a t_i$ for $i \in \theta(a)$ and $s_i = t_i$ for $i \notin \theta(a)$.

We will be interested in *trace closed specifications*. Such a specification is a regular language $L \subseteq \bigcup_{i=1,\dots,n} \Sigma_i$ closed under trace equivalence determined by the alphabets $(\Sigma_i)_{i=1,\dots,n}$. Although we will use the trace terminology, we will not recall it here (we refer the reader to [18, 8]). We can assume that L is given as a deterministic parity automaton $\mathcal{A} = \langle Q, \Sigma, q^0, \delta, \Omega\rangle$ with a diamond property. This means that for every state q and two letters a, b such that $\theta(a) \cap \theta(b) = \emptyset$, we have that $\delta(\delta(q, a), b) = \delta(\delta(q, b), a)$ and the visited priorities are the same: $\{\Omega(\delta(q, a)), \Omega(\delta(\delta(q, a), b))\} = \{\Omega(\delta(q, b)), \Omega(\delta(\delta(q, b), a))\}$. These properties allow us to define $\delta(q, \{a_1, \dots, a_k\})$ for a set of pairwise independent letters from Σ and $\min(q, \{a_1, \dots, a_k\})$ which is the minimal priority among $\Omega(q_1), \dots, \Omega(q_k)$ where $q_i = \delta(q, \{a_1, \dots, a_i\})$.

The goal is to find a controller for the product system such that all the possible paths satisfy the given specification. As the system is distributed, of course we are interested in local controllers that cannot use the global view of the system.

A *local controller* for process i is a function $\rho_i : \Sigma_i^* \times \Sigma_i^e \to \mathcal{P}(\Sigma_i^c)$. This function suggests a set of possible moves for the system depending on the part of the execution up to that moment. *Distributed (global) controller* is the set of local controllers, one for each process. An execution respecting distributed controller $(\rho_1, \ldots, \rho_n)$ is an infinite path $\vec{s}_1 \to^{a_1} \vec{s}_2 \to^{a_2} \cdots$ such that for each $a_k \in \Sigma^c$ we have $a_k \in \rho_i((a_1 \ldots a_{k-1}) \restriction \Sigma_i)$ for all $i \in \theta(a_k)$; in particular it means that $(a_1 \ldots a_{k-1}) \restriction \Sigma_i$ should end on a letter from Σ_i^e. A partial executions a finite sequence with the same property. Intuitively the property says that an action can be taken if it is allowed by controllers of all the processes involved in it.

A distributed controller is *non-blocking* if every partial execution can be prolonged.

As it was shown in [18] it is undecidable to determine if for a given product system and an asynchronous automaton there is a non-blocking distributed strategy such that all the plays respecting this strategy are accepted by the automaton.

Madhusudan and Thiagarajan give two restrictions on the set of controllers which make the problem decidable:

1. Controller $(\rho_i)_{i=1,\ldots,n}$ is *com-rigid* if every possible value R of every ρ_i is com-rigid which means that $\theta(a) = \theta(b)$ for all $a, b \in R$.

2. Controller $(\rho_i)_{i=1,\ldots,n}$ is *clocked* if for every $i = 1, \ldots, n$ and $u, v \in \Sigma_i^*$, if $|u| = |v|$ then $\rho_i(u) = \rho_i(v)$.

The *MT-control problem* is: given $S = \langle S_i^c, S_i^e, \{\to_i^a\}_{a \in \Sigma_i}, s_i^{in} \rangle_{i=1,\ldots,n}$ and an automaton $\mathcal{A}$ decide if there exists a distributed non-blocking, com-rigid and clocked controller such that all the paths of S respecting this controller are accepted by $\mathcal{A}$.

Encoding into a game Take the tuple of local systems and an automaton defining a trace closed language as above:

$$\{\langle S_i^c, S_i^e, \{\to_i^a\}_{a \in \Sigma_i}, s_i^{in} \rangle\}_{i=1,\ldots,n} \qquad \mathcal{A} = \langle Q, \Sigma, q^0, \delta, \Omega \rangle$$

We define a game $\mathcal{G}$ consisting of components $G_0, G_1, \ldots, G_n$. Then we will show that the distributed strategy in $\mathcal{G}$ gives a distributed controller for the system as described above.

The games $G_1 \ldots, G_n$ are defined to reflect the behaviour of the respective controllers. We put $G_i = \langle P_i, E_i, \theta_i \rangle$ where:

1. $P_i = \{?\}$, $E_i = \{R \mid R \subseteq \Sigma_i^c$ and R is com-rigid $\}$.

2. $? \to R$, $R \to ?$ are in T_i for all com-rigid R,

Intuitively in the game G_i player is repeatedly asked to provide a set of actions he agrees to execute.

Game G_0 keeps track of the moves of the automaton $\mathcal{A}$ and the states of the local transition systems. Let $S^c = S_1^c \times \cdots \times S_n^c$, $AP = \mathcal{P}(\{1, \ldots, n\})$.

1. $E_0 = S^c \times AP \times Q \times range(\Omega)$.

2. $P_0 = (E_0 \times \mathcal{P}(\Sigma)) \cup \{\text{LOSE}\}$.

3. $(\vec{s}, ap, q, min, \varsigma) \rightarrow (\vec{t}, ap, \delta(q, \varsigma), min(q, \varsigma))$ where $\vec{s} \rightarrow^{\varsigma} \vec{t}$, i.e. a path labelled by ς in the product system. Moreover LOSE does not have any successors (if this position is reached then the players lose).

4. $\Omega(\text{LOSE}) = 1$ (fixed arbitrarily), for other positions, the priority is the value of the min component.

Intuitively, the S^c and Q components keep track of the path in the system and in the automaton respectively. The min component is need as in one step the automaton reads many letters and we need to memorize the minimal state visited while it did so. The AP component specifies which of the processes are still active. It is needed because the environment may decide to block some of the components forever.
Consider an environment position

$$\eta = ((\vec{s}, ap, q, min), R_1, \ldots, R_n).$$

Define $\mathit{enabled}(\eta, ap') = \{i : \exists_{a \in R_i}\ \theta(a) \subseteq ap' \wedge \forall_{j \in \theta(a)}\ a \in R_j\}$. This is the set of processes than can move in the current position. We define when we have a global transition $\eta \rightarrow \pi$:

1. if $\mathit{enabled}(\eta, ap) = \emptyset$ and $ap = \{1, \ldots, n\}$ then $\pi = \text{LOSE}$,

2. if $\mathit{enabled}(\eta, ap) \neq \emptyset$ then for all $ap' \subseteq ap$ and all $\varsigma, x_1, \ldots, x_n$ we have a transition to $\pi = ((\vec{s}, ap', q, min, \varsigma), x_1, \ldots, x_n,)$ whenever:

   T1 $\mathit{enabled}(\eta, ap') \neq \emptyset$.

   T2 For each $a \in \varsigma$, if $a \in \Sigma^c$ then $\theta(a) \subseteq \mathit{enabled}(\eta, ap')$ and if $a \in \Sigma^e$ then $\theta(a) \subseteq \mathit{enabled}(\eta, ap)$.

   T2' For each $i \in \mathit{enabled}(\eta, ap')$ there is precisely one letter in $\varsigma \cap \Sigma_i^c$ and precisely one letter in $\varsigma \cap \Sigma_i^e$.

   T3 $x_i =?$ for all $i \in \mathit{enabled}(\eta, \{1, \ldots, n\})$ and $x_i = R_i$ otherwise; observe that here we use $\{1, \ldots, n\}$ and not ap or ap'.

The intuition behind this transition is relatively simple. The environment decides which actions ς to execute. These should be executable actions, i.e., the respective processes should be in $\mathit{enabled}(\eta, ap)$. There should be at most one controllable and one environment action for each process. Additionally the environment may decide to block some processes forever by removing them from the set ap' of active processes. The processes that remain active should execute an action. The condition $T3$ says that all the components, even those that are not in ap, will move if they can. This means that the choice of ap influences only the 0-th component and not the other components of the game.

Lastly, the acceptance condition in $\mathcal{G}$ is given by the parity condition on the positions of P0.

Lemma 31 $\mathcal{G}$ is I-gluable and $\text{GLUE}(\mathcal{G})$ is an environment deterministic game.

Proof
Consider an environment position $\eta = ((\vec{s}, ap, q, min), R_1, \ldots, R_n)$. By the condition T3 for every player position such that $\eta \to \pi$, all the components of π but the 0-th component are fixed by $R_1, \ldots, R_n$.

We want to apply glue operation to 0-th component. For this we need to check that $\mathcal{G}$ is gluable. By the above paragraph $\mathcal{G}$ is 0-deterministic. P0 is never idle by the definition of transitions, hence $\mathcal{G}$ is 0-independent. Lastly, the acceptance condition is the parity condition on 0. Hence, by Theorem 13, there is a distributed strategy in $\text{GLUE}(\mathcal{G})$ iff there is one in $\mathcal{G}$. Additionally $\text{GLUE}(\mathcal{G})$ is $[1, n]$-deterministic.

By the observation from the first paragraph of this proof it follows that every environment position $\widetilde{\eta}$ in $\text{GLUE}(\mathcal{G})$ has at most one successor. Hence, $\text{GLUE}(\mathcal{G})$ is environment deterministic. □

Using Fact 5 we obtain:

Corollary 32 It is decidable if there is a distributed strategy in the game $\mathcal{G}$ constructed from a given system and a given local specification as described above.

Equivalence It remains to show that there is a distributed strategy in the game $\mathcal{G}$ iff there is a non-blocking, com-rigid and clocked controller in the original system. In this subsection we assume that both the system and the game are fixed as above.

The first observation is that a clocked controller $\rho : \Sigma^* \times \Sigma^c \to \mathcal{P}(\Sigma^e)$ can be as well presented as a function $\rho : \mathbb{N} \to \mathcal{P}(\Sigma^e)$ because such a controller depends only on the length of the argument. Similarly if we have a distributed strategy $(\sigma_1, \ldots, \sigma_n)$ in $\mathcal{G}$ then each of the components is a function $\sigma_i : (E_i P_i)^+ \to E_i$. As P_i is a singleton we can identify such a function with a function $\sigma_i : \mathbb{N} \to E_i$.

Lemma 33 If there is a non-blocking, com-rigid and clocked controller for the system then there is a distributed winning strategy in $\mathcal{G}$.

Proof
Given a controller $(\rho_1, \ldots, \rho_n)$ for the system we define $\sigma_i(k) = \rho_i(k)$ for all i and k. Consider a play:

$$((\vec{s}_1, ap_1, q_1, min_1), \vec{R}_1) \to ((\vec{s}_1, ap_2, q_1, min_1, \varsigma_1), \vec{R}'_1) \to \\ ((\vec{s}_2, ap_2, q_2, min_2), \vec{R}_2) \to ((\vec{s}_2, ap_3, q_2, min_2, \varsigma_2), \vec{R}'_2) \cdots$$

in $\mathcal{G}$ respecting the strategy $(\sigma_1, \ldots, \sigma_n)$ and such that $\vec{s}_1 = s^{in}$, $ap_1 = \{1, \ldots, n\}$, $min_1 = 0$ and $\vec{R}_1 = (\rho_1(0), \ldots, \rho_n(0))$.

We would like to show that it is winning. For this it is enough to observe that
$$\vec{s}_1 \to^{\varsigma_1} \vec{s}_2 \to^{\varsigma_2} \cdots$$
is a path in the global system respecting the controller $(\rho_1, \ldots, \rho_n)$. Moreover
$$q_1 \to^{\varsigma_1} q_2 \to^{\varsigma_2} \cdots$$
is the run of $\mathcal{A}$ on $\varsigma_1 \cdot \varsigma_2 \cdots$ and $\min_i = min(q_i, \varsigma_i)$. As the run is accepting the minimal priority appearing infinitely often in the sequence $min_1, min_2, \ldots$ is even. Hence the play in $\mathcal{G}$ is indeed winning for the players. $\square$

Lemma 34 If there is a distributed winning strategy in $\mathcal{G}$ then there is a distributed controller for the system.

Proof
Let $(\sigma_1, \ldots, \sigma_n)$ be a distributed winning strategy in $\mathcal{G}$. We define the controller for S by $(\rho_1, \ldots, \rho_n)$ with $\rho_i(k) = \sigma_i(k)$ for all i and k.

Recall that on the words $w \in \Sigma^*$ we have the trace equivalence relation induced by the the dependence relation: $(a, b) \in D$ iff $\theta(a) \cap \theta(b) \neq \emptyset$. We denote by $w \sim w'$ the fact that the two words are trace equivalent. Let $M(w)$ denote the set of minimal letters in w, i.e., the letters a such that $aw' \sim w$ for some w'. Let u_1 be a linearization of $M(w)$ and let w_1 be such that $u_1 \cdot w_1 \sim w$. We then define u_2 in a similar way but starting from w_1. Repeating this procedure ad infinitum we obtain the word $nice(w) = u_1 \cdot u_2 \cdots$. It follows that $nice(w) \sim w$.

Let w be a path in S respecting $(\rho_1, \ldots, \rho_n)$ and starting in s^{in}. Directly from the definitions we have that $nice(w)$ is also a path in S. It is of the form:
$$\vec{s}_1 \to^{u_1} \vec{s}_2 \to^{u_2} \vec{s}_3 \to^{u_3} \cdots$$
where $u_1 \cdot u_2 \cdots$ is a decomposition of $nice(w)$ as described above. From the definition of $u_1 \cdot u_2 \cdots$ we have that $u_i \in (\Sigma^c)^*$ for even i and $u_i \in (\Sigma^e)^*$ for odd i. Let
$$\vec{q}_1 \to^{u_1} \vec{q}_2 \to^{u_2} \vec{q}_3 \to^{u_3} \cdots$$
be the run of $\mathcal{A}$ on $nice(w)$ and let $\min_i = \min(q_i, u_i)$. Finally let $ap_k = \{i : \exists_{j \geq k} \exists_{a \in \Sigma_i}\ a \in u_j\}$. It the follows that $ap_k \supseteq ap_{k+1}$ and $ap_k = ap_{k+1}$ if k is odd. We obtain that
$$\begin{aligned}((\vec{s}_1, ap_1, q_1, min_1, \vec{R}_1) \to ((\vec{s}_1, ap_2, q_1, min_1, \varsigma_1), \vec{R}'_1) \to \\ ((\vec{s}_2, ap_2, q_2, min_2, \vec{R}_2) \to ((\vec{s}_2, ap_3, q_2, min_2, \varsigma_2), \vec{R}'_2) \cdots\end{aligned}$$
is a path of $\mathcal{G}$ respecting the strategy $(\sigma_1, \ldots, \sigma_n)$. $\square$

Corollary 35 The MT-control problem is decidable.

Proof
Given an instance of the control problem we translate it into a game $\mathcal{G}$. By Corollary 32 we can decide if there is a distributed winning strategy in $\mathcal{G}$. By the two lemmas above we know that there is such a strategy iff the instance of the control problem has a solution. $\square$

9 Discrete-event systems

In this section we describe how to formalize and solve the decentralized controller synthesis problem of Rudi and Wonham. This is the only problem where we do not have a simpler solution via distributed games. Instead of artificially forcing the solution to use the game terminology we prefer to show how the problem itself can be presented as a game. For completeness we present also a very simple argument why the problem is decidable [6]. The argument does not use any of our theorems about games neither we could find any interesting generalization of it in the language of games.

A *plant* P over a set of actions Σ is a deterministic finite state automaton i.e. $P = \langle S, \Sigma, s_0, \delta\rangle$ where S is a finite set of states; $\delta : S \times \Sigma \dot{\to} S$ is the partial transition function and s_0 is the initial state. We do not specify the set of final states because we assume that all the states are final. The language of P is the set of all the sequences $w = a_1a_2 \dots a_k$ that are accepted by P. Hence $L(P)$ is closed by prefixes.

A *controller* with the set $\Sigma^c \subseteq \Sigma$ of controllable actions and set $\Sigma^o \subseteq \Sigma$ of observable actions is a deterministic finite state automaton $C = \langle S^c, \Sigma, s^c_0, \delta^c\rangle$ such that $\delta^c(s, a)$ is defined for every $a \in \Sigma \setminus \Sigma^c$ and $\delta^c(s, a) = s$ for every $a \in \Sigma \setminus \Sigma^o$ for which $\delta^c(s, a)$ is defined.

By $P \times C$ we denote the synchronized product of the two automata, i.e., $P \times C = \langle S \times S^c, \Sigma, (s_0, s^c_0), \delta^\times\rangle$ where $\delta^\times((s, s^c), a) = (\delta(s, a), \delta^c(s^c, a))$. Of course $\delta^\times((s, s^c), a)$ is only defined if both components are defined.

Fix a number n of processes and for each process $i = 1, \dots, n$ fix two alphabets $\Sigma^c_i \subseteq \Sigma$ and $\Sigma^o_i \subseteq \Sigma$ of controllable and observable actions, respectively. For an action a, let $obs(a) = \{i \mid a \in \Sigma^o_i\}$ be the set of controllers that can observe it and let $con(a) = \{i \mid a \in \Sigma^c_i\}$ be the set of controllers that can control it.

The *Rudie and Wonham control problem* is: for given P, two languages $M, N \subseteq \Sigma^*$ and $(\Sigma^c_i, \Sigma^o_i)_{i=1,\dots,n}$ decide if there exist controllers $C_1, \dots, C_n$ such that

$$M \subseteq L(P \times C_1 \times \cdots \times C_n) \subseteq N$$

Observe that we can "hide" the plant into the specification because the above is equivalent to

$$M \subseteq L(C_1 \times \cdots \times C_n) \subseteq N \cup (\Sigma^* \setminus L(P)) \tag{1}$$

if we assume that $L(M) \subseteq L(P)$. We can assume this and the fact that M is prefix closed because otherwise the problem does not have a solution.

Encoding into a game The distributed game encoding the problem has $n+1$ players. The Player 0 codes the the automaton $\mathcal{A}_M = \langle S^M, \Sigma, s_0^M, \delta_M \rangle$ for the language M. Players $1, \ldots, n$ code controllers.

The 0 component coding the automaton $\mathcal{A}_M$ is $G_0 = \langle P_0, E_0, T_0 \rangle$ where:

- $P_0 = \Sigma \times S^M \cup \{\perp\}$, $E_0 = S^M$; and the players' moves are restricted to $(a, s) \to \delta^M(s, a)$.

For each i, the component $G_i = \langle P_i, E_i, T_i \rangle$ is defined by:

- $P_i = \Sigma_i^o$; $E_i = \mathcal{P}(\Sigma_i^c)$ and no restriction on the moves of the player or environment

We have a global transition $(s, R_1, \ldots, R_n) \to ((a, s), x_1, \ldots, x_n)$ from an environment position if

- $\delta^M(s, a)$ is defined and $a \in \bigcap_{i \in con(a)} R_i$;
- $x_i = a$ for all $i \in obs(a)$ and $x_i = R_i$ for all $i \notin obs(a)$.

We also have a transition $(s, R_1, \ldots, R_n) \to (\perp, R_1, \ldots, R_n)$ if $\delta^M(s, a)$ is defined and there is no move to $((a, s), x_1, \ldots, x_n)$ as described above. The winning condition is to avoid a state with $\perp$.

In the 0-th component the player has no choice, so there is unique strategy for this component, denote it by σ_0. A strategy for every other component is a function $\sigma_i : P_i(E_i \cdot P_i)^* \to E_i$. We can as well consider it to be a function $\sigma_i : P_i^* \to E_i$ by putting

$$\sigma_i(p_1 p_2 \ldots p_n) = \sigma_i\big(p_1 \cdot \sigma_i(p_1) \cdot p_2 \cdot \sigma_i(p_1 \sigma_i(p_1) p_2) \cdots p_n\big)$$

This is because if we play according to the strategy, then the environment positions we see are determined by the strategy.

Definition 36 The *language of a strategy* $\sigma_i : P_i^* \to E_i$, denoted $L(\sigma_i)$ is the set of words $w \in \Sigma^*$ such that $v = w|_{\Sigma_i^o}$ respects σ_i which means that for each prefix $u a_i a_{i+1}$ of v, if $a_{i+1} \in \Sigma_i^c$ then $a_{i+1} \in \sigma(u a_i)$.

Lemma 37 A distributed strategy $\langle \sigma_1, \ldots, \sigma_n \rangle$ is winning in $\mathcal{G}$ from the position $(s_0^M, \sigma_1(\varepsilon), \ldots, \sigma_n(\varepsilon))$ iff $M \subseteq \bigcap_{i=1,\ldots,n} L(\sigma_i)$.

Proof

This lemma follows directly from the definition of $\mathcal{G}$ by observing that each word $w \in M$ determines at most one path respecting the distributed strategy in $\mathcal{G}$. The distributed strategy is winning if for every word $w \in M$ there is a path respecting the strategy labelled by w. On the other hand there is such a path iff $w \in \bigcap_{i=1,\ldots,n} L(\sigma_i)$. $\square$

Now we define the strategy

$$\widehat{\sigma}_i(v) = \{a \in \Sigma_i^c : \exists w.\ w|_{\Sigma_i^o} = v \text{ and } \delta(s_0^M, wa) \text{ is defined}\}$$

By definition $M \subseteq L(\widehat{\sigma}_i)$. The next lemma says that $\widehat{\sigma}_i$ is the smallest strategy with this property.

Lemma 38 For every i, if $M \subseteq L(\sigma_i)$ then $L(\widehat{\sigma}_i) \subseteq L(\sigma_i)$.

Proof
Suppose conversely that $w \in L(\widehat{\sigma}_i) \setminus L(\sigma_i)$. By the definition of $L(\sigma_i)$ it means that w ends on the letter in Σ_i^c and by the definition of $\widehat{\sigma}_i$ we have that $\delta^M(s_0^M, w)$ is defined. But then $w \in L(M)$ which is a contradiction. $\square$

Corollary 39 A Rudie and Wonham synthesis problem as in (1) has a solution iff $\bigcap_{i=1,\dots,n} L(\widehat{\sigma}_i) \subseteq N \cup (\Sigma^* \setminus P)$.

Proof
Given a strategy $\langle \sigma_1, \dots, \sigma_n \rangle$ we can define the controllers $C_1, \dots, C_n$ where C_i is the minimal deterministic automaton recognizing $L(\sigma_i)$. It is easy to see that it is a controller with set Σ_i^o of observable and set Σ_i^c of controllable actions. From the definition $L(C_1 \times \cdots \times C_n) = \bigcap_{i=1,\dots,n} L(\sigma_i)$.

For the other direction, suppose that we have controllers $C_1, \dots, C_n$ solving the problem. For each $i = 1, \dots, n$ we define a strategy $\sigma_i : P_i^* \to E_i$ by $\sigma_i(v) = \{a \in \Sigma_i^c : va \text{ is permitted by } C_i\}$. Once again $L(C_1 \times \cdots \times C_n) = \bigcap_{i=1,\dots,n} L(\sigma_i)$. $\square$

10 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 [22, 2, 7].

Having decided on a model we have looked for a set of tools that can be applied in different instances of DSP. We have given two theorems allowing to simplify distributed games. We have shown how they can be used in two instances of DSP from the literature. In the appendix we consider two more cases. 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, for each particular architecture, a class of specifications for which the problem is decidable. We want to do this by analyzing the encoding of this DSP 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 a 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*, 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] P. Morris. *Introduction to game theory.* Springer-Verlag, 1994.

[21] A. W. Mostowski. Games with forbidden positions. Technical Report 78, University of Gdansk, 1991.

[22] M. J. Osborne and A. Rubinstein. *A Course in Game Theory.* The MIT Press, 1994.

[23] R. Parikh. Social software. Synthese, 2002 (To appear).

[24] M. Pauly. *Logic for social software.* PhD thesis, Institute for Logic, Language and computation, Universiteit van Amsterdam, 2001. ILLC Dissertation Series 2001-10.

[25] G. L. Peterson and J. H. Reif. Multi-person alternation. In *Proc. IEEE FOCS*, pages 348–363, 1979.

[26] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. ACM POPL*, pages 179–190, 1989.

[27] 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.

[28] P. J. G. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(2):81–98, 1989.

[29] S. Ricker and K. Rudie. Incorporating knowledge into discrete-event control systems. *IEEE Trans. on Automatic Control*, 45(9):1656–1668, 2000.

[30] K. Rudie and W. Wonham. Think globally, act locally: Decentralized supervisory control. *IEEE Trans. on Automat. Control*, 37(11):1692–1708, 1992.

[31] 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.

[32] S. Tripakis. Undecidable problems of decentralized observation and control. In *IEEE Conference on Decision and Control*, 2001.