Petri Games: Synthesis of Distributed Systems with Causal Memory

Bernd Finkbeiner
Universität des Saarlandes
finkbeiner@cs.uni-saarland.de

Ernst-Rüdiger Olderog
Carl von Ossietzky Universität Oldenburg
olderog@informatik.uni-oldenburg.de

We present a new multiplayer game model for the interaction and the flow of information in a distributed system. The players are tokens on a Petri net. As long as the players move in independent parts of the net, they do not know of each other; when they synchronize at a joint transition, each player gets informed of the causal history of the other player. We show that for Petri games with a single environment player and an arbitrary bounded number of system players, deciding the existence of a safety strategy for the system players is EXPTIME-complete.

1 Introduction

Games are a natural model of the interaction between a computer system and its environment. Specifications are interpreted as winning conditions, implementations as strategies. An implementation is correct if the strategy is *winning*, i.e., it ensures that the specification is met for all possible behaviors of the environment. Algorithms that determine the winner in the game between the system and its environment can be used to determine whether it is possible to implement a specification (the *realizability* question) and, if the answer is yes, to automatically construct a correct implementation (the *synthesis* problem).

We present a new game model for the interaction and the flow of information in a distributed system. The players are tokens on a Petri net. In Petri nets, causality is represented by the flow of tokens through the net. It is therefore natural to designate tokens also as the carriers of information. As long as different players move in concurrent places of the net, they do not know of each other. Only when they synchronize at a joint transition, each player gets informed of the history of the other player, represented by all places and transitions on which the joint transition causally depends. The idea is that after such a joint transition, a strategy for a player can take the history of all other players participating in the joint transition into account. Think of a workflow where a document circulates in a large organization with many clerks and has to be signed by everyone, endorsing it or not. Suppose a clerk wants to make the decision whether or not to endorse it depending on who has endorsed it already. As long as the clerk does not see the document, he is undecided. Only when he receives the document, he sees all previous signatures and then makes his decision.

We call our extension of Petri nets *Petri games*. The players are organized into two teams, the system players and the environment players, where the system players wish to avoid a certain "bad" place (i.e., they follow a safety objective), while the environment players wish to reach just such a place. To partition the tokens into the teams, we label each place as belonging to either the system or the environment. A token belongs to a team whenever it is on a place that belongs to the team.

In the tradition of Zielonka's automata [28], Petri games model distributed systems with *causal memory*, i.e., distributed systems where the processes memorize their causal history and communicate it to each other during each synchronization [10, 11, 16]. Petri games thus abstract from the concrete content of a communication in that we assume that the processes always exchange the *maximal* possible

Adriano Peron and Carla Piazza (Eds.):
Proceedings of the Fifth International Symposium on
Games, Automata, Logics and Formal Verification (GandALF 2014)
EPTCS 161, 2014, pp. 217–230, doi:10.4204/EPTCS.161.19

information, i.e., their entire causal history. This is useful at a design stage before the details of the interface have been decided and one is more interested in restricting *when* a communication can occur (e.g., when a device is connected to its base station, while a network connection is active, etc.) than *what* may be communicated. The final interface is then determined by the information actually used by the winning strategies, which is typically only a small fraction of the causal history. Note that even though we assume the players to communicate everything they know, the flow of information in a Petri game is far from trivial. At any point, the players of the Petri game may have a different level of knowledge about the global state of the game, and the level of informedness changes dynamically as a result of the synchronizations chosen by the players.

Consider the development of a distributed security alarm system. If a burglar triggers the alarm at one location, the alarm should go off everywhere, and all locations should report the location where the original alarm occurred. This situation is depicted as a Petri net in Fig. 1. The token that initially resides on place *Env* represents the environment, which is, in our example, the burglar, who can decide to break into our building either at location A or B. The tokens that initially reside on places *A* and *B* represent the distributed controller consisting of two processes, the one on the left for location A and the one on the right for location B. In the following, we will refer to the Petri net of Fig. 1 as a *Petri game*, to emphasize that the tokens in fact represent players and that the nondeterminism present in the net is to be restricted by the (yet to be determined) strategy of the system players.

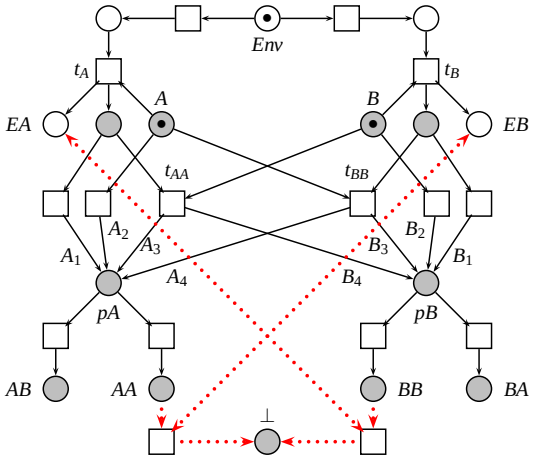

Figure 1: Introductory example of a Petri game modeling a distributed security alarm. Places belonging to the system players *A* and *B* are shown in gray. In the Petri game, the transitions to the bad place ⊥ are shown with dotted lines.

The system players and the environment players move on separate places in the net, the places belonging to the system players are shown in gray. In the example, our goal is to find a strategy for the system players that avoids a *false alarm*, i.e., a marking where the environment token is still on *Env* and at least one system token is on one of the places at the bottom, i.e., *AA*, *AB*, etc., and a *false report*, i.e., a marking where the environment token is on place *EA* and some system token is on *AB* or *BB* or a marking where the environment token is on *EB* and some system token is on *AA* or *BA*. To identify such undesirable markings we introduce a distinguished place ⊥. Fig. 1 shows (dashed) transitions towards ⊥ firing at two instances of false reports, when tokens are on both *EA* and *BB* or on both *EB* and *AA*. Similar transitions for other erroneous situations are omitted here to aid visibility.

Suppose that, in our Petri game, the burglar breaks into location *A* by taking the left transition. Once the system token in *A* has recorded this via transition t_A, it has two possibilities: either synchronize with the system token in *B* by taking transition t_{AA}, or skip the communication and go straight to *pA* via transition A_1. Intuitively, only the choice to synchronize is a good move, because the system token in *B* has no other way of hearing about the alarm. The only remaining move for the system token in *B* would be to move "spontaneously" via transition B_2 to *pB*, at which point it would need to move to *BA*, because

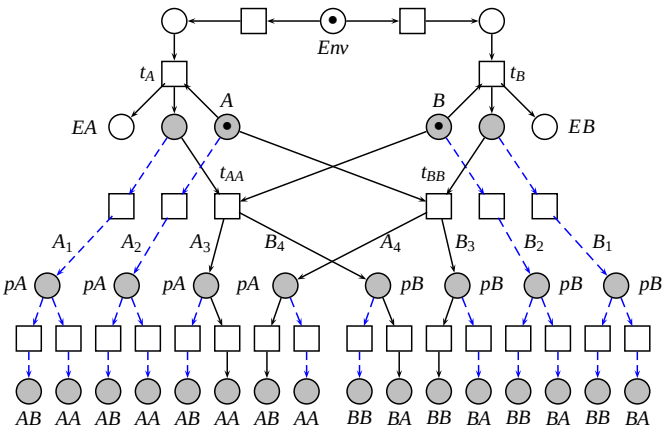

Figure 2: Unfolding of the Petri game in Fig. 1. To aid visibility, the transitions leading to $\perp$ are omitted from the unfolding. If the transitions shown with dashed lines are removed from the unfolding, the resulting net is a winning strategy for the system players.

the combination of *BB* and *EA* would constitute a false alarm. However, the token in *pB* has no way of distinguishing this situation from one where the environment token is still on *Env*; in this situation, the move to *EA* would also reach a false alarm.

Our definition of strategies is based on the *unfolding* of the net, which is shown for our example in Fig. 2. By eliminating all joins in the net, *net unfoldings* [6, 8, 19] separate places that are reached via multiple causal histories into separate copies. In the example, place *pB* has been unfolded into four separate copies, corresponding to the four different ways to reach *pB*, via the transition arcs B_1 through B_4. Each copy represents different knowledge: in B_1, only *B* knows that there has been a burglary at location *B*; in B_2, *B* knows nothing; in B_3, *B* knows that *A* knows that there has been a burglary at position *B*; in B_4, *B* knows that there has been a burglary at location *A*. (Symmetric statements hold for *pA* and the transition arcs $A_1 - A_4$.) In the unfolding, it becomes clear that taking transition B_2 is a bad move, because reaching the bad marking containing *Env* and either *BA* or *BB* has now become unavoidable. A *strategy* is a subprocess of the unfolding that preserves the local nondeterminism of the environment token. Fig. 2 shows a winning strategy for the system players: by omitting the dashed arrows, they can make the bad place $\perp$ unreachable and therefore win the game.

We show that for a single environment token and an arbitrary (but bounded) number of system tokens, deciding the existence of a safety strategy for the system players is EXPTIME-complete. This means that as long as there is a single source of information, such as the *input* of an algorithm or the *sender* in a communication protocol, solving Petri games is no more difficult than solving standard combinatorial games under complete information [25]. The case of Petri games with two or more environment tokens, i.e., situations with two or more *independent* information sources, remains open.

The remainder of the paper is structured as follows. In Section 3 we introduce the notion of Petri games and define strategies based on net unfoldings. In Section 4 we show that for concurrency preserving games every strategy can be distributed over local controllers. In Section 5 we introduce the new notion of mcuts on net unfoldings. In Section 6 we show that the problem of deciding the winner of a Petri game is EXPTIME-complete. Related work and conclusions are presented in Sections 7 and 8. Due to space limitations, proofs have been moved into the full version of this paper.

2 Petri nets

We recall concepts from Petri net theory [4, 6–8, 14, 18, 19, 23]. A *place/transition* (*P/T*) *Petri net* or simply *net* $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, In)$ consists of possibly infinite, disjoint sets $\mathcal{P}$ of *places* and $\mathcal{T}$ of transitions, a *flow relation* $\mathcal{F}$, which is a multiset over $(\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$, and an *initial marking* In. In general, a *marking* of $\mathcal{N}$ is a finite multiset over $\mathcal{P}$. It represents a global state of $\mathcal{N}$. By convention, a net named $\mathcal{N}$ has the components $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, In)$, and analogously for nets with decorated names like $\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}^U$.

The elements of $\mathcal{P} \cup \mathcal{T}$ are called *nodes* of $\mathcal{N}$, thereby referring to the bipartite graphic representation of nets, where places are drawn as circles and transitions as boxes. The flow relation $\mathcal{F}$ is represented by directed arrows between places and transitions. An arrow from a place p to a transition t is decorated by a *multiplicity* k if $\mathcal{F}(p,t) = k$, and analogously, an arrow from a transition t to a place p is decorated by a *multiplicity* k if $\mathcal{F}(t,p) = k$. We use a double arrow arc between a place and a transition if there are arcs in both directions. A marking M is represented by placing $M(p)$ *tokens* in every place p.

$\mathcal{N}$ is *finite* if it has only finitely many nodes, and *infinite* otherwise. For nodes x, y we write $x \mathcal{F} y$ if $\mathcal{F}(x,y) > 0$. The *precondition* of y is the multiset $pre(y)$ over nodes defined by $pre(y)(x) = \mathcal{F}(x,y)$. The *postcondition* of x is the multiset $post(x)$ over nodes defined by $post(x)(y) = \mathcal{F}(x,y)$. When stressing the dependency on the net $\mathcal{N}$, we write $pre^{\mathcal{N}}(y)$ and $post^{\mathcal{N}}(x)$ instead of $pre(y)$ and $post(x)$. As in [6] we require *finite synchronization* [4] and non-empty pre- and postconditions: $pre(t)$ and $post(t)$ are finite, non-empty multisets for all transitions $t \in \mathcal{T}$.

A transition t is *enabled* at a marking M if the multiset inclusion $pre(t) \subseteq M$ holds. *Executing* or *firing* such a transition t at M yields the successor marking M' defined by $M' = M - pre(t) + post(t)$. We denote this by $M[t\rangle M'$. The set of *reachable markings* of a net $\mathcal{N}$ is denoted by $\mathcal{R}(\mathcal{N})$ and defined by $\mathcal{R}(\mathcal{N}) = \{M \mid \exists\, t_1, \ldots, t_n \in \mathcal{T} : In[t_1\rangle M_1[t_2\rangle \ldots [t_n\rangle M_n = M\}$. A net $\mathcal{N}$ is *k-bounded* for a given $k \in \mathbb{N}$ if $M(p) \leq k$ holds for all $M \in \mathcal{R}(\mathcal{N})$ and all $p \in \mathcal{P}$. It is *bounded* if it is k-bounded for some given k and *safe* if it is 1-bounded.

$\mathcal{F}^+$ denotes the transitive closure and $\mathcal{F}^*$ the reflexive, transitive closure of $\mathcal{F}$. Nodes x and y are *in conflict*, abbreviated by $x \sharp y$, if there exists a place $p \in \mathcal{P}$, different from x and y, from which one can reach x and y via $\mathcal{F}^+$, exiting p by different arcs. A node x is in *self-conflict* if $x \sharp x$.

We use the notations ${}^\circ\mathcal{N} = \{p \in \mathcal{P} \mid pre(p) = \emptyset\}$ and $\mathcal{N}^\circ = \{p \in \mathcal{P} \mid post(p) = \emptyset\}$ for the sets of places without incoming or outgoing transitions, respectively. For a multiset M over $\mathcal{P}$ let $\mathcal{N}[M]$ result from $\mathcal{N}$ by changing its initial marking In to M. For a set X of nodes we define the *restriction* of $\mathcal{N}$ to X as the net $\mathcal{N} \restriction X = (\mathcal{P} \cap X, \mathcal{T} \cap X, \mathcal{F} \restriction (X \times X), In \restriction X)$.

Consider two nets $\mathcal{N}_1$ and $\mathcal{N}_2$. Then $\mathcal{N}_1$ is an *initial subnet* or simply *subnet* of $\mathcal{N}_2$, denoted by $\mathcal{N}_1 \sqsubseteq \mathcal{N}_2$, if $\mathcal{P}_1 \subseteq \mathcal{P}_2$, $\mathcal{T}_1 \subseteq \mathcal{T}_2$, $\mathcal{F}_1 \subseteq \mathcal{F}_2$, and $In_1 = In_2$. A *homomorphism* from $\mathcal{N}_1$ to $\mathcal{N}_2$ is a mapping $h : \mathcal{P}_1 \cup \mathcal{T}_1 \to \mathcal{P}_2 \cup \mathcal{T}_2$ with $h(\mathcal{P}_1) \subseteq \mathcal{P}_2$ and $h(\mathcal{T}_1) \subseteq \mathcal{T}_2$, and with $\forall t \in \mathcal{T}_1 : h[pre(t)] = pre(h(t))$ and $h[post(t)] = post(h(t))$. If additionally $h[In_1] = In_2$, then h is called an *initial* homomorphism. An (*initial*) *isomorphism* is a bijective (initial) homomorphism.

Occurrence nets and unfoldings. To represent the occurrences of transitions with both their causal dependency and conflicts (nondeterministic choices), we consider occurrence nets, branching processes, and unfoldings of Petri nets as in [6, 8, 14, 19]. We follow the axiomatic presentation in [6], taking [18] into account for dealing with P/T Petri nets.

An *occurrence net* is a Petri net $\mathcal{N}$, where $\forall t \in \mathcal{T} : pre(t)$ and $post(t)$ are sets, $\forall p \in \mathcal{P} : |pre(p)| \leq 1$, the inverse flow relation $\mathcal{F}^{-1}$ is well-founded, no transition $t \in \mathcal{T}$ is in self-conflict, and $In = {}^\circ\mathcal{N}$. Note that an occurrence net is a safe net. Two nodes x, y of an occurrence net are *causally related* if $x \mathcal{F}^* y$ or $y \mathcal{F}^* x$. They are *concurrent* if they are neither causally related nor in conflict. If $x \mathcal{F}^+ y$ then x is called

a *causal predecessor* of y, abbreviated $x < y$. We write $x \leq y$ if $x < y$ or $x = y$. The *causal past* of a node y is the set $past(y) = \{x \mid x \leq y\}$.

A *branching process* of a net $\mathcal{N}$ is a pair $\beta = (\mathcal{N}^U, \lambda)$, where $\mathcal{N}^U$ is an occurrence net and λ is a "labeling", i.e., a homomorphism from $\mathcal{N}^U$ to $\mathcal{N}$ that is injective on transitions with the same precondition: $\forall t_1, t_2 \in \mathcal{T}^U : pre(t_1) = pre(t_2) \wedge \lambda(t_1) = \lambda(t_2)$ implies $t_1 = t_2$. If λ is initial, β is called an *initial branching process*. The *unfolding* of a net $\mathcal{N}$ is an initial branching process $\beta_U = (\mathcal{N}^U, \lambda)$ that is *complete* in the sense that every transition of the net is recorded in the unfolding: $\forall t \in \mathcal{T}, \forall C \subseteq \mathcal{P}^U$: if C is a set of concurrent places and $\lambda[C] = pre(t)$, then there exists a transition $t^U \in \mathcal{T}^U$ such that $pre(t^U) = C$ and $\lambda(t^U) = t$.

Let $\beta_1 = (\mathcal{N}_1, \lambda_1)$ and $\beta_2 = (\mathcal{N}_2, \lambda_2)$ be two branching processes of $\mathcal{N}$. A homomorphism from β_1 to β_2 is a homomorphism h from $\mathcal{N}_1$ to $\mathcal{N}_2$ with $\lambda_1 = \lambda_2 \circ h$. It is called *initial* if h is initial; it is an *isomorphism* if h is an isomorphism. β_1 and β_2 are *isomorphic* if there exists an initial isomorphism from β_1 to β_2. β_1 *approximates* β_2 if there exists an initial injective homomorphism from β_1 to β_2. β_1 is a *subprocess* of β_2 if β_1 *approximates* β_2 with the identity on $\mathcal{P}_1 \cup \mathcal{T}_1$ as the homomorphism. Thus $\mathcal{N}_1 \sqsubseteq \mathcal{N}_2$ and $\lambda_1 = \lambda_2 \restriction (\mathcal{P}_1 \cup \mathcal{T}_1)$. If β_1 *approximates* β_2 then β_1 is isomorphic to a subprocess of β_2.

In [6] is shown that the unfolding $\beta_U = (\mathcal{N}^U, \lambda)$ of a net $\mathcal{N}$ is unique up to isomorphism and that every initial branching process β_1 of $\mathcal{N}$ approximates β_U. Thus up to isomorphism we can assume that β_1 is a subprocess of β_U.

Cuts and sequential composition. A *cut* of an occurrence net $\mathcal{N}$ is a maximal subset of the places that are pairwise concurrent. For a cut C let $C^- = \{x \in \mathcal{P} \cup \mathcal{T} \mid \exists s \in C : \ x \leq s\}$ and $C^+ = \{x \in \mathcal{P} \cup \mathcal{T} \mid \exists s \in C : \ s \leq x\}$. A cut C splits $\mathcal{N}$ into the two nets $\mathcal{N} \restriction C^-$ and $(\mathcal{N} \restriction C^+)[C]$; it also splits a branching process $(\mathcal{N}, \lambda)$ into two branching processes $(\mathcal{N}_1, \lambda_1)$ and $(\mathcal{N}_2, \lambda_2)$, where $\mathcal{N}_1 = \mathcal{N} \restriction C^-$ and $\mathcal{N}_2 = (\mathcal{N} \restriction C^+)[C]$ and $\lambda_1 = \lambda \restriction C^-$ and $\lambda_2 = \lambda \restriction C^+$.

Two branching processes $(\mathcal{N}_1, \lambda_1)$ and $(\mathcal{N}_2, \lambda_2)$ of a given P/T Petri net are *compatible* if $\lambda_1[\mathcal{N}_1^\circ] = \lambda_2[{}^\circ\mathcal{N}_2]$. Given two compatible branching processes $(\mathcal{N}_1, \lambda_1)$ and $(\mathcal{N}_2, \lambda_2)$, we can up to isomorphisms of $\mathcal{N}_1$ and of $\mathcal{N}_2$ assume that $\mathcal{N}_1^\circ = {}^\circ\mathcal{N}_2$ and construct a unique branching process $(\mathcal{N}, \lambda)$ with $\mathcal{N} \restriction C^- = \mathcal{N}_1$ and $(\mathcal{N} \restriction C^+)[C] = \mathcal{N}_2$, and $\lambda \restriction C^- = \lambda_1$ and $\lambda \restriction C^+ = \lambda_2$, for the cut $C = \mathcal{N}_1^\circ = {}^\circ\mathcal{N}_2$. This branching process is the *sequential composition* of $(\mathcal{N}_1, \lambda_1)$ and $(\mathcal{N}_2, \lambda_2)$, denoted by $(\mathcal{N}, \lambda) = (\mathcal{N}_1, \lambda_1); (\mathcal{N}_2, \lambda_2)$. If $(\mathcal{N}_1, \lambda_1)$ is an initial branching process, then so is $(\mathcal{N}, \lambda)$.

Causal nets and concurrent runs. Executions of Petri nets are represented by causal nets and concurrent runs as in [4, 19]. A *causal net* is an occurrence net $\mathcal{N}$, where $\forall p \in \mathcal{P} : |post(p)| \leq 1$. Thus in a causal net there are no (self-) conflicts. A (*concurrent*) *run* or *process* of $\mathcal{N}$ is a special case of a branching process $\beta_R = (\mathcal{N}^R, \rho)$, where $\mathcal{N}^R$ is a causal net. If ρ is initial, β_R is called an *initial run*. Note that every initial run of $\mathcal{N}$ approximates the unfolding $\beta_U = (\mathcal{N}^U, \lambda)$ of $\mathcal{N}$. Thus up to isomorphism we can assume the an initial run of $\mathcal{N}$ is a subprocess of β_U.

The marking *reached by* a finite initial run $\beta_R = (\mathcal{N}^R, \rho)$ of $\mathcal{N}$ is denoted by $[\beta_R\rangle$ and defined as the multiset $[\beta_R\rangle = \rho[(\mathcal{N}^R)^\circ]$. We remark that the set $\mathcal{R}(\mathcal{N})$ of reachable markings of $\mathcal{N}$ can be obtained via the runs as follows: $\mathcal{R}(\mathcal{N}) = \{\, [\beta_R\rangle \mid \beta_R \text{ is a finite initial run of } \mathcal{N} \,\}$.

3 Petri Games

We wish to model games where the players proceed independently of each other, without information of each others state, unless they explicitly communicate. To this end, we introduce Petri games, defined as place/transition (P/T) Petri nets, where the set of places is partitioned into a subset $\mathcal{P}_S$ belonging to the *system players* and a subset $\mathcal{P}_E$ belonging to the *environment*. Additionally, the Petri game identifies a

set $\mathscr{B}$ of *bad* places (from the point of view of the system), which indicate a victory for the environment. Formally, a Petri game is a structure $\mathscr{G} = (\mathscr{P}_S, \mathscr{P}_E, \mathscr{T}, \mathscr{F}, In, \mathscr{B})$, where the (*underlying*) *Petri net of the game* $\mathscr{G}$ is $\mathscr{N} = (\mathscr{P}, \mathscr{T}, \mathscr{F}, In)$ with places $\mathscr{P} = \mathscr{P}_S \cup \mathscr{P}_E$. Players are modeled by the tokens of $\mathscr{N}$. Throughout this paper we stipulate that there is only one environment player.

Example 3.1 *Fig. 3 shows the underlying P/T net $\mathscr{N}$ of a small Petri game for two system players in place Sys and one environment player in place Env. Environment places are white and system places are gray. The environment chooses A or B by executing one of the transitions t_1 or t_2. The goal of the system players is to achieve the same decisions as Env, i.e., both system players should choose A' if Env chooses A, and B' if Env chooses B. Without communication, the system players do not know which decision the environment has taken. However, when both system players and the environment communicate by synchronizing via the transitions $test_1$ or $test_2$, the system players learn about the decision taken by the environment and can mimic it. If $test_1$ was successful, they choose A' via transition t'_1, and if $test_2$ was successful, they choose B' via transition t'_2.* □

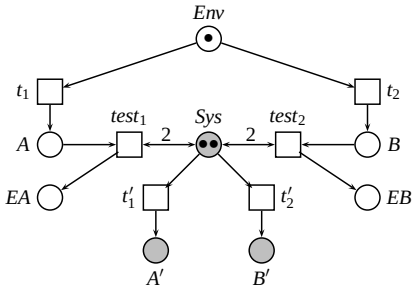

Figure 3: Petri game for achieving same decisions, where *Env* and *Sys* can synchronize via two transitions $test_1$ and $test_2$. Transitions from *EA* and B' and from *EB* and A' to a bad place have been omitted to aid visibility.

We wish to model that players learn about previous decisions of other players by communication. To this end, we use the *unfolding* of the net, where each place that is reachable via several transition paths is duplicated into several copies of the place, each one representing its causal past. The *unfolding* of a game $\mathscr{G}$ is the unfolding of the underlying net $\mathscr{N}$, denoted by the branching process $\beta_U = (\mathscr{N}^U, \lambda)$, where $\mathscr{N}^U$ is an occurrence net and λ is an initial homomorphism from $\mathscr{N}^U$ to $\mathscr{N}$, which "labels" the places and transitions of $\mathscr{N}^U$ with the places and transitions of $\mathscr{N}$. In the graphic representation of games and unfoldings gray places denote elements of $\mathscr{P}_S$ and white places elements of $\mathscr{P}_E$.

Example 3.2 *Fig. 4 shows the unfolding of the Petri game in Fig. 3.* □

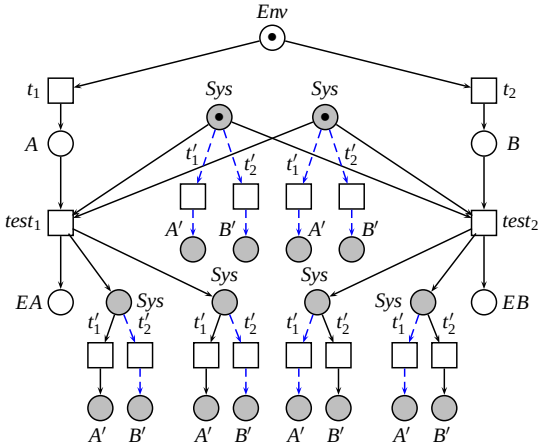

Figure 4: Unfolding of the Petri game in Fig. 3. If the transitions shown with dashed lines are removed from the unfolding, the resulting net represents a winning strategy for the system players, i.e., on the left-hand side, the system players choose A', and on the right-hand side, the system players choose B'.

A global strategy is obtained from the unfolding by deleting some of the branches that are under control of the system players. We call this a "global" strategy because it looks at all players simultaneously. Note that nevertheless a strategy describes for each place which transitions the player in that place can take. Formally, this is expressed by the net-theoretic notion of subprocess.

An *unfolded* (*global*) *strategy* for the system players in $\mathscr{G}$ is a subprocess $\sigma = (\mathscr{N}^\sigma, \lambda^\sigma)$ of the unfolding $\beta_U = (\mathscr{N}^U, \lambda)$ of $\mathscr{N}$ subject to the following conditions for all $p \in \mathscr{P}^\sigma$:

(S1) if $p \in \mathscr{P}^\sigma_S$ then σ is deterministic at p,

(S2) if $p \in \mathscr{P}^\sigma_E$ then $\forall t \in \mathscr{T}^U : (p,t) \in \mathscr{F}^U \wedge |pre^U(t)| = 1 \Rightarrow (p,t) \in \mathscr{F}^\sigma$, i.e., at an environment place the strategy does not restrict any local transitions.

Here $\mathscr{P}^\sigma_S = \mathscr{P}^\sigma \cap \lambda^{-1}(\mathscr{P}_S)$ denotes the system places and $\mathscr{P}^\sigma_E = \mathscr{P}^\sigma \cap \lambda^{-1}(\mathscr{P}_E)$ the environment places in $\mathscr{P}^\sigma$. A strategy σ is *deterministic at a place* p if for all $M \in \mathscr{R}(\mathscr{N}^\sigma)$, the set of reachable markings in $\mathscr{N}^\sigma$:

$$p \in M \Rightarrow \exists^{\leq 1} t \in \mathscr{T}^\sigma : p \in pre(t) \subseteq M.$$

Due to the unfolding, a decision taken by σ in a place p depends on the causal past of p, which may be arbitrarily large. The adjective "global" indicates that σ looks at all players simultaneously. Local controllers are discussed in Section 4.

Example 3.3 *Fig. 4 shows also a global strategy for the system players of the Petri game in Fig. 3.* □

A (*concurrent*) *play* of a Petri game $\mathscr{G}$ is an initial concurrent run π of the underlying net $\mathscr{N}$. If π contains a place of $\mathscr{B}$, the *environment wins* π. Otherwise, the *system players win* π. Note that up to isomorphism we can assume that π is a subprocess of the unfolding β^U. A play π *conforms to* a strategy σ if π is a subprocess of σ. A strategy σ for the system players is *winning* if the system players win every play that conforms to σ.

Since the winning condition of a game is a *safety objective*, the system players can satisfy it by doing nothing. To avoid such trivial solutions, we look for strategies σ that are *deadlock avoiding* in the sense that $\forall M \in \mathscr{R}(\mathscr{N}^\sigma) : \exists t \in \mathscr{T}^U : pre(t) \subseteq M \Rightarrow \exists t \in \mathscr{T}^\sigma : pre(t) \subseteq M$, i.e., if the unfolding can execute a transition the strategy σ can as well, thus avoiding unnecessary deadlocks. A marking where there is no enabled transition in the unfolding either is not a deadlock. Then we say that the game has *terminated*.

A (*global*) *strategy* for the system players in $\mathscr{G}$ is a pair $\sigma = (\mathscr{N}^\sigma, h^\sigma)$ consisting of a safe net $\mathscr{N}^\sigma$ and an initial homomorphism h^σ from $\mathscr{N}^\sigma$ to $\mathscr{N}$ that is injective on transitions with the same preset, i.e., $\forall t_1, t_2 \in \mathscr{T}^U : pre(t_1) = pre(t_2) \wedge \lambda(t_1) = \lambda(t_2)$ implies $t_1 = t_2$, subject to the conditions (S1) and (S2) above. A global strategy σ may have cycles and thus be finite, i.e., have a finite set $\mathscr{P}^\sigma \cup \mathscr{T}^\sigma$.

4 Distribution

We show that for Petri games with a concurrency preserving underlying net, every global strategy σ is distributable over local controllers. A net $\mathscr{N}$ is *concurrency preserving* if every transition $t \in \mathscr{T}$ satisfies $|pre(t)| = |post(t)|$. The *parallel composition* $\mathscr{N}_1 || \mathscr{N}_2$ of two nets $\mathscr{N}_i = (\mathscr{P}_i, \mathscr{T}_i, \mathscr{F}_i, In_i)$, $i = 1,2$, with $\mathscr{P}_1 \cap \mathscr{P}_2 = \emptyset$ is defined as the Petri net $\mathscr{N}_1 || \mathscr{N}_2 = (\mathscr{P}_1 \cup \mathscr{P}_2, \mathscr{T}_1 \cup \mathscr{T}_2, \mathscr{F}_1 \cup \mathscr{F}_2, In_1 \cup In_2)$ obtained by taking the componentwise union. The two nets synchronize on each common transition $t \in \mathscr{T}_1 \cap \mathscr{T}_2$ as in the process algebra CSP [13, 20].

Let $\mathscr{N} = (\mathscr{P}, \mathscr{T}, \mathscr{F}, In)$ be a concurrency preserving, safe net with the places partitioned into system and environment places $\mathscr{P} = \mathscr{P}_S \cup \mathscr{P}_E$. A *slice* of $\mathscr{N}$ describes the course of one token in $\mathscr{N}$. Formally, it is a net $S = (\mathscr{P}^S, \mathscr{T}^S, \mathscr{F}^S, In^S)$, where $\mathscr{P}^S \subseteq \mathscr{P}_S$ or $\mathscr{P}^S \subseteq \mathscr{P}_E$, $\mathscr{T}^S \subseteq \mathscr{T}$, $\mathscr{F}^S \subseteq \mathscr{F}$, $In^S \subseteq In$ are minimal subsets satisfying

- $|In^S| = 1$ and $\forall p \in \mathscr{P}^S : post^N(p) \subseteq \mathscr{T}^S$ and $\forall t \in \mathscr{T}^S : |pre^S(t)| = |post^S(t)| = 1$,
- $\mathscr{F}^S = \mathscr{F} \restriction (\mathscr{P}^S \times \mathscr{T}^S) \cup (\mathscr{T}^S \times \mathscr{P}^S)$.

The net $\mathcal{N}$ is called *reachable* if every place and transition of $\mathcal{N}$ is reachable from its initial marking.

Lemma 4.1 (Parallel Composition of Slices) *Every safe reachable net $\mathcal{N}$ which is concurrency preserving is the parallel composition of slices: $\mathcal{N} = \|_{S\in\mathcal{S}}\, S$, where $\mathcal{S}$ is a family of slices of $\mathcal{N}$ such that $\{\mathcal{P}^S \mid S \in \mathcal{S}\}$ is a partition of $\mathcal{P}$.*

A *local controller* specifies the moves of a single player in a Petri game. It is a pair $\mathcal{C} = (\mathcal{N}^C, h^C)$ consisting of a safe net $\mathcal{N}^C$ with one token, i.e., $|In^C| = 1$ and $\forall t \in \mathcal{T}^C : |pre^C(t)| = |post^C(t)| = 1$, and a *weak homomorphism* h^C from $\mathcal{N}^C$ to $\mathcal{N}$, the underlying net of the Petri game. A local controller $\mathcal{C}$ is *finite* if $\mathcal{P}^C \cup \mathcal{T}^C$ is a finite set. It may have nondeterministic choices of transitions that are resolved (later) by synchronization with other controllers working in parallel. Unfolding $\mathcal{N}^C$ yields a branching process $\beta^C = (\mathcal{N}^{CU}, \lambda^C)$, where λ^C is an initial homomorphism from $\mathcal{N}^{CU}$ to $\mathcal{N}^C$. Then $\mathcal{C}^U = (\mathcal{N}^{CU}, h^C \circ \lambda^C)$ is an *unfolded local controller*.

A (n unfolded) strategy σ is *distributable* if σ can be represented as the parallel composition of (unfolded) local controllers for the environment and the system players in the sense that the reachable part of the parallel composition is isomorphic to σ. Using Lemma 4.1 we show:

Lemma 4.2 (Distribution) *Every unfolded global strategy for a concurrency-preserving Petri game is distributable.*

Example 4.3 *The global strategy of Fig. 4 can be distributed into the local controllers of Fig. 5.* □

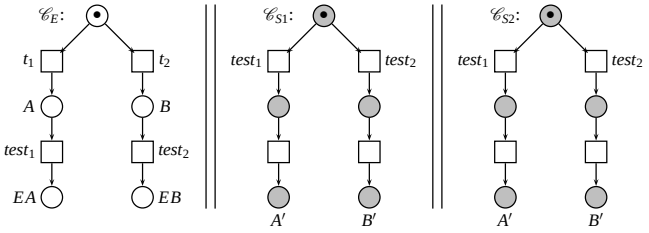

Figure 5: The local controllers $\mathcal{C}_E$ for the environment and $\mathcal{C}_{S1}$, $\mathcal{C}_{S2}$ for the system players work in parallel and synchronize on the transitions $test_1$ and $test_2$. Applying the parallel composition $\|$ to the three controller nets yields the winning strategy of Fig. 4.

Theorem 4.4 *If the system players in a bounded and concurrency preserving Petri game have a winning strategy, then they have a finite* distributable *winning strategy.*

5 Cuts

In an unfolded strategy σ, a decision taken by σ in a place p depends on the causal past of p, which may be arbitrarily large. Similar to model checking approaches based on net unfoldings [7], we use *cuts* (maximal subset of pairwise concurrent places) as small summaries of the causal past. The standard notion of cuts is, however, problematic for games with multiple players, because it collects places without regard for the (possibly different) knowledge of the individual players about the causal past. To solve this problem, we introduce a new kind of cut, called *mcut*, which guarantees that the system players can be considered to be perfectly informed about the environment decisions.

Throughout this section, we consider a Petri game $\mathcal{G}$ with underlying net $\mathcal{N}$, unfolding $\beta_U = (\mathcal{N}^U, \lambda)$, and an unfolded strategy $\sigma = (\mathcal{N}^\sigma, \lambda^\sigma)$, so $\mathcal{N}^\sigma \sqsubseteq \mathcal{N}^U$ and $\lambda^\sigma = \lambda \restriction (\mathcal{P}^\sigma \cup \mathcal{T}^\sigma)$. Since

in $\mathcal{N}^\sigma$ the nondeterminism of $\mathcal{N}^U$ has been restricted, we distinguish for a node $x \in \mathscr{P}^\sigma \cup \mathscr{T}^\sigma$ the postconditions $post^\sigma(x)$ and $post^U(x)$ taken in the nets $\mathcal{N}^\sigma$ and $\mathcal{N}^U$, respectively. Note that $post^\sigma(x) \subseteq post^U(x)$. For preconditions we have $pre^\sigma(x) = pre^U(x)$. Thus, while the postconditions of nodes may be different in $\mathcal{N}^\sigma$ and $\mathcal{N}^U$, their preconditions are identical.

Futures, mcuts and ecuts.

For a cut C of an occurrence net let $C^+ = \{x \in \mathscr{P} \cup \mathscr{T} \mid \exists s \in C : \ s \leq x\}$, where $\leq$ denotes the reflexive *causal predecessor* relation given by $\mathscr{F}^*$. For a subnet $\mathcal{N}' \sqsubseteq \mathcal{N}^U$ and a cut C of $\mathcal{N}'$ we write $\mathcal{N}'_{C^+} = (\mathcal{N}' \restriction C^+)[C]$.

Note that $(\mathcal{N}^U_{C^+}, \ \lambda \restriction C^+)$ is an initial branching process of the net $\mathcal{N}[\lambda[C]]$, which is like $\mathcal{N}$ but starts at the initial marking $\lambda[C]$. For cuts C and C' we write $C \leq C'$ if $\forall x \in C \ \exists y \in C' : x \leq y$, and $C < C'$ if $C \leq C'$ and $C \neq C'$.

The *future in* $\mathcal{N}^\sigma$ of a node x in $\mathcal{N}^\sigma$ is the set $fut^\sigma(x) = \{y \in \mathscr{P}^\sigma \cup \mathscr{T}^\sigma \mid x \leq y\}$. A *p-cut* is a cut containing the place p.

For an environment place $p \in \mathscr{P}^\sigma$ we introduce now $mcut(p)$ as the w.r.t. $\leq$ minimal p-cut C such that for all places $q \in C$, either the system players have *maximally progressed* at q, in the sense that any further system transition would require an additional environment transition starting from place p, or the future starting at q does not depend on the environment.

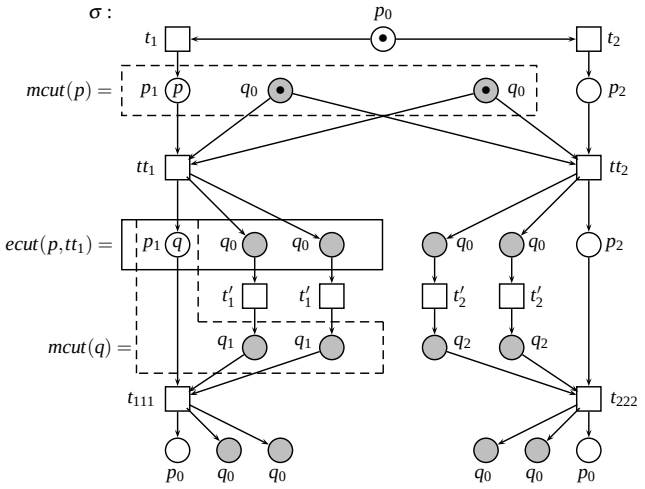

Figure 6: Shown is an initial part of an unfolding. Consider the places p and q both labeled with p_1. Then $mcut(p)$ contains the upper places labeled p_1, q_0, q_0 and $ecut(p, tt_1)$ contains the places labeled p_1, q_0, q_0 in the middle, whereas $mcut(q)$ contains the places labeled p_1, q_1, q_1, with the system players maximally progressed. Both mcuts have only places of type 1.

The formal definition is as follows: For a p-cut C and a place $q \in C$ we define $type(q) = 1$ if $\forall t \in post^\sigma(q) : (t$ reachable in $\mathcal{N}^\sigma_{C^+} \Rightarrow p \leq t)$ and $type(q) = 2$ if $\forall t \in fut^\sigma(q) : (t$ reachable in $\mathcal{N}^\sigma_{C^+} \Rightarrow p \not\leq t)$. Note that $type(p) = 1$. By *type-1(C)* we denote the set of all places in C that have type 1, and analogously for *type-2(C)*. Then we define: $mcut(p) = \min_{\leq}\{C \mid C$ is a p-cut of $\mathcal{N}^\sigma \ \wedge \forall q \in C : type(q) = 1 \vee type(q) = 2\}$. For an example, see Fig. 6.

Lemma 5.1 (Existence of mcuts) *For every environment place $p \in \mathscr{P}^\sigma$, $mcut(p)$ is well-defined.*

An *ecut* results from an mcut by firing a single *environment* transition. Formally, given an environment place $p \in \mathscr{P}^\sigma$ and a transition $t \in post^\sigma(p)$ with environment participation let $ecut(p,t)$ be the cut C obtained by firing t at $mcut(p)$, formally $mcut(p)[t\rangle C$. For an example, see Fig. 6.

6 Deciding Petri Games

We now reduce Petri games to games over finite graphs, which can subsequently be solved by a standard fixed point construction. Unlike the Petri game, the finite-graph game has only two players, Player 0 and

Player 1, which both act on complete information. We construct a finite-graph game that is equivalent to the Petri game in the sense that the system players have a deadlock-avoiding and winning strategy in the Petri game iff Player 0 has a winning strategy in the finite-graph game. The key idea is that Player 1, representing the environment, is only allowed to make a decision at mcuts, which guarantees that the system players learn about the decision before they have to make their next choice. In this way, the system players can be considered to be perfectly informed.

A *finite-graph game* $(V, V_0, V_1, I, E, W_0, W_1)$ consists of a finite set $V = V_0 \cup V_1$ of states, partitioned into Player 0's states V_0 and Player 1's states V_1, a set of initial states $I \subseteq V$, an edge relation $E \subseteq V \times V$, and disjoint sets of winning states $W_0, W_1 \subseteq V$ for Player 0 and Player 1, respectively. A play is a possibly infinite sequence of states, constructed by letting Player 0 choose the next state from the E-successors whenever the play is in V_0 and letting Player 1 choose otherwise. Player 0 wins if the play reaches W_0 or forever avoids visiting W_1.

A *strategy* for Player 0 is a function $f : V^* \cdot V_0 \to V$ that maps a prefix of a play ending in a state owned by Player 0, i.e., a sequence of states that ends in a V_0 state, to some successor state according to E. A play *conforms to* a strategy f, if all successors of V_0 states in the play are chosen according to f. A strategy is *winning* for Player 0 if there is an initial state $v_0 \in I$ such that all plays that start in v_0 and conform to f are won by Player 0.

To simulate a Petri game $\mathscr{G} = (\mathscr{P}_S, \mathscr{P}_E, \mathscr{T}, \mathscr{F}, In, \mathscr{B})$, we build a finite-graph game where the states are multisets consisting of triples $(p, type, T)$, where $p \in \mathscr{P}$ is a place, *type* is a type, i.e., 1 or 2, and $T \in 2^{\mathscr{T}} \cup \{\top\}$ is either a set of transitions representing the transitions chosen by a token in p or a special symbol $\top$, indicating that a new choice needs to be made. We call these multisets *decision sets*. For k-bounded Petri games, we limit the cardinality of the decision sets to $|\mathscr{P}| \cdot k$. A state belongs to Player 1 if the decision set corresponds to an mcut, and to Player 0 otherwise; i.e., the states of Player 1 consist of all decision sets where there is no $\top$ symbol and the outgoing transitions from type-1 places are either disabled or have an environment place in their precondition, the states of Player 0 consist of all other decision sets. The game starts with some initial marking, which fixes an arbitrary classification of types, all outgoing transitions for the environment places and an arbitrary selection of transitions for the system places. When there is a $\top$ symbol, Player 0 makes a choice for the transition set. In other situations, the game continues by Player 0 choosing transitions from system places and Player 1 choosing transitions that involve an environment place. The choices of both players are restricted to the transitions allowed in the decision set. There is an additional restriction based on the type of the places, which we will discuss below. Whenever a transition has fired, Player 0 chooses a new set of transitions for the newly reached system places. (In environment places, all outgoing transitions are always allowed.)

If no more transitions from type-1 places are enabled, the game ends. If this is due to termination, or if the decision set includes type-2 places, Player 0 wins. Player 1 wins if the game ends due to deadlock, if nondeterminism is encountered (i.e., two separate transitions, or two separate instances of the same transition, are enabled that share some system place in their precondition and have no environment places in their precondition), or if a bad place is visited.

Example 6.1 *Figure 7 shows (a part of) the finite-graph game corresponding to the Petri game from Fig. 3. In Fig. 3, the transitions leading to bad places have been omitted. For the purposes of this example, we assume that there is one additional transition $t_\perp$, which takes one token each from places EA and B' and puts one token on the bad place $\perp$ and one token back on EA. States of Player 0 are shown as rectangles, states of Player 1 as diamonds. Winning states for Player 0 are shown with double lines, winning states for Player 1 with bold lines. In addition to the initial state v_0 shown in Fig. 7, the game has further initial states that are omitted here. Player 0 has a winning strategy from v_0 (following the*

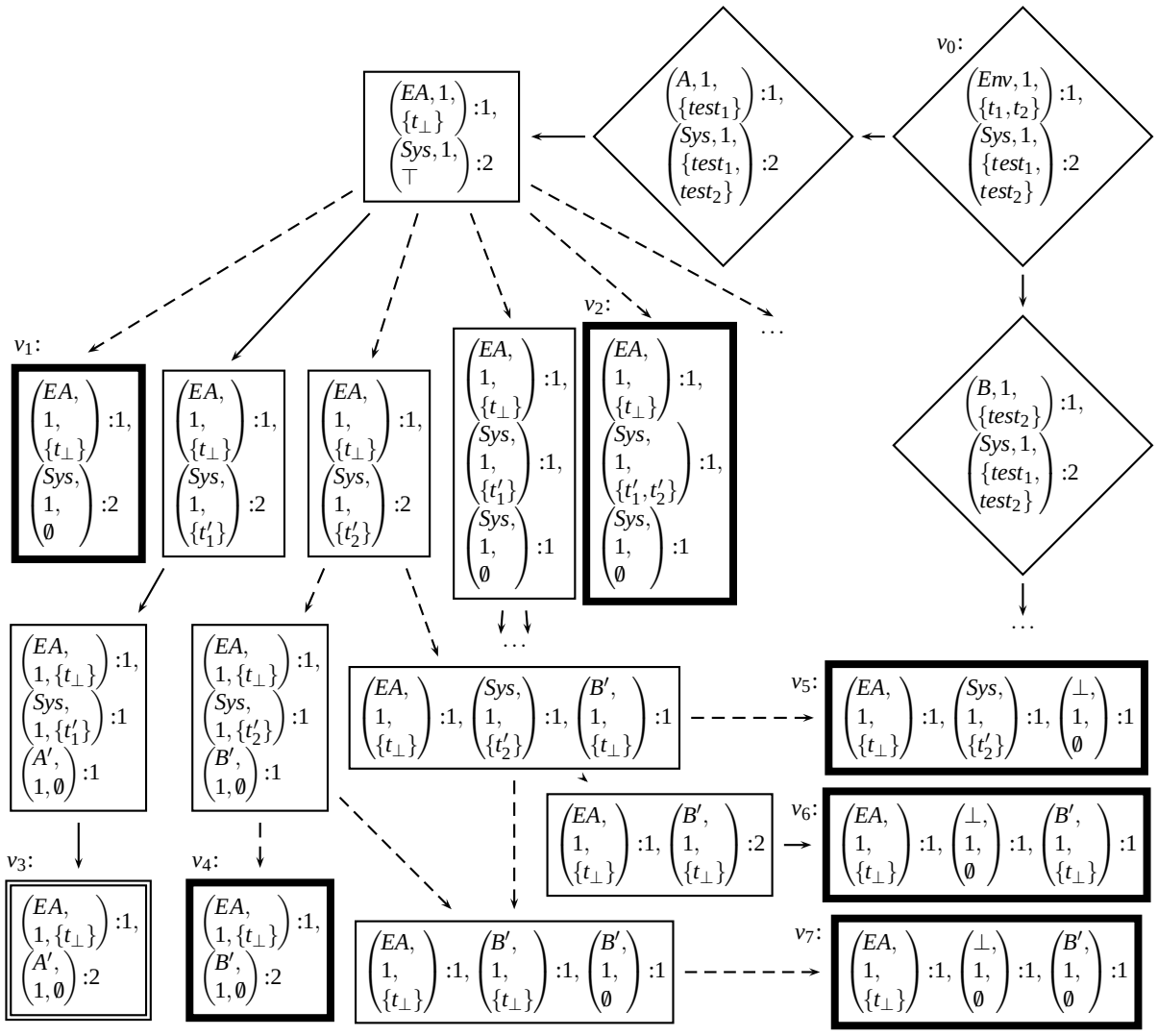

Figure 7: Part of the finite-graph game corresponding to the Petri game from Fig. 3.

edges shown with solid lines). In state v_3, Player 0 wins, because the game terminates. In states v_1 and v_4, Player 1 wins, because a deadlock is reached. In state v_2, Player 1 wins, because nondeterminism is encountered. In states v_5, v_6, and v_7, Player 1 wins, because the bad place $\perp$ is reached.

The finite-graph game as described so far does not yet ensure the correctness of the classification of the places into types 1 and 2. We need to make sure that the Petri game can indeed continue from type-2 places without dependencies on the environment and without visits to bad places. For this purpose, we identify, in a preprocessing step, the largest subset $\mathscr{D}$ of the set of decision sets that consists of only those decision sets that are either terminating or have at least one transition from type-2 places to another decision set in $\mathscr{D}$ that does not contain a bad place. We restrict the game to $\mathscr{D}$ and only allow (for both players) transitions that originate from type-1 places.

Lemma 6.2 (Reduction to Finite-Graph Games) *The system players have a deadlock-avoiding winning strategy in the Petri game iff Player 0 has a winning strategy in the finite-graph game.*

To prove Lemma 6.2, we translate a winning strategy for Player 0 in the finite-graph game into a deadlock-avoiding winning strategy for the system players in the Petri game and vice versa.

Given a winning strategy f of the finite-graph game, we inductively build a strategy σ for the Petri game following the tree structure given by the possible choices of the environment token. In this way, we construct for each environment place in σ a unique mcut, and for each subsequent ecut a causal net connecting the type-1 places of the ecut to the next mcut. The strategy is deadlock-avoiding because the decision sets of mcuts with deadlocks are winning for Player 1. The strategy is winning, because the plays that conform to f avoid bad places. If the play in the finite-graph game is infinite, then the play in the Petri game is also infinite, traversing an infinite sequence of mcuts. If the play in the finite-graph game is finite, then this may be due to termination, in which case the play in the Petri game terminates as well; otherwise, the play must have reached a decision set with type-2 places, from which the play in the Petri game continues infinitely.

Given a deadlock-avoiding winning strategy σ of the Petri game and a prefix $w \in V^* \cdot V_0$ of a play of the finite-graph game, we compute the choice $f(w)$ of the strategy for the finite-graph game by simulating w in σ: starting with the initial marking and firing the transitions of w in σ, we arrive at a cut of σ which is not an mcut; we choose an arbitrary enabled system transition and choose the decision set of the resulting cut as $f(w)$. For a cut C in σ, the decision set is the multiset $dec[C] = \{(\lambda(p), type(p), \lambda(post^\sigma(p))) \mid p \in C\}$. The resulting strategy f is winning from the decision set of the initial cut of σ.

The size of the finite-graph game is exponential in the size of the Petri game; the Petri game can therefore be solved in single-exponential time. A matching lower bound follows from the EXPTIME-hardness of combinatorial games [25].

Theorem 6.3 (Game Solving) *For bounded Petri games with one environment player and a bounded number of system players, the question whether the system players have a winning strategy is EXPTIME-complete. If a winning strategy for the system players exists, it can be constructed in exponential time.*

Although the reachability problem is decidable also for unbounded Petri nets [17], we cannot decide unbounded Petri games. This is an immediate consequence of the undecidability of VASS (Vector Addition Systems with States) games [2].

Theorem 6.4 *For unbounded Petri games, the question whether the system players have a winning strategy is undecidable.*

7 Related Work

There is a significant body of work on synthesis and control based on Petri nets (cf. [5,12,22,27]). These approaches differ from ours in that they solve supervisory control problems or two-player games on the state space created by the Petri net. Hence, these approaches solve the single-process synthesis problem, as opposed to the multi-process synthesis problem for concurrent systems considered in this paper.

For distributed systems, much work has focused on finding architectures for which the realizability question is decidable. Most research on this problem is in the setting of synchronous processes with *shared-variable* communication, introduced by Pnueli and Rosner. A general game model for these types of realizability problems are Walukiewicz and Mohalik's *distributed games* [26]. While undecidable in general [21], the distributed synthesis problem can be solved in the Pnueli/Rosner setting for a number of interesting architectures, including pipelines [24], rings [15], and generally all architectures where the processes can be *ordered* according to their informedness [9]. Unfortunately, all these decision procedures have nonelementary complexity. For the asynchronous games based on Zielonka's automata,

decidability has been also been established for specific classes of architectures such as trees [11]. Another important line of work concerns the alternating-time temporal logics, which are interpreted over concurrent game structures [3]. The difference between Petri games and these approaches is that Petri games link informedness to causality instead of referring to a separate, static, specification of the relative informedness in an architecture.

In the literature on Petri nets, unfoldings have been used conceptually to connect Petri net theory with event structures [4, 6, 18, 19] and practically to obtain algorithms for deciding reachability. These algorithms are based on constructing a finite canonical prefix of the in general infinite net unfolding that contain all reachable markings [7, 8, 14]. We use net unfoldings as a *uniform conceptual basis* to define strategies and plays as well as suitable cuts for analyzing the strategies. Net unfoldings enable us to formalize the intended degree of informedness of each player at a given place: it is the causal past of that place, concurrent activities beyond that past are not visible. Such a causal view is also chosen in [10], for the setting of Zielonka's automata [28].

8 Conclusions

We have introduced Petri games, an extension of Petri nets where the tokens represent players who make individual, independent decisions. Using tokens as the carriers of information, Petri games link information flow to causality: decisions may only use information resulting from decisions that they also depend on causally. This makes Petri games a convenient formalism to reason about asynchronous concurrent programs as well as manufacturing cells [27], business work flows [1], and other distributed applications. Our synthesis algorithm is applicable to Petri games where the number of system tokens is bounded by some arbitrary number, and the number of environment tokens is bounded by 1. This leaves two important open problems. The first open problem is whether Petri games with more than one environment token are decidable; if so, what is the precise complexity? The decidability result for tree architectures [11] is both encouraging and discouraging; encouraging, because at least some architectures that are undecidable in the Pnueli/Rosner setting are decidable for distributed systems with causal memory. Discouraging, because the complexity of the synthesis algorithm is nonelementary. The second open problem is to find synthesis methods for unbounded Petri games. While we have shown that the problem is in general undecidable, it is an interesting challenge for future research to develop semi-algorithms for unbounded Petri games and to find other restrictions besides boundedness that make the synthesis problem decidable.

References

[1] W.M.P.v. Aalst (1998): *The application of Petri nets to workflow management*. *J. of Circuits, Systems and Computers* 8, pp. 21–66, doi:10.1142/S0218126698000043.

[2] P.A. Abdulla, A. Bouajjani & J. d'Orso (2003): *Deciding Monotonic Games*. In: *Proc. CSL, LNCS* 2803, Springer-Verlag, pp. 1–14, doi:10.1007/978-3-540-45220-1_1.

[3] R. Alur, T.A. Henzinger & O. Kupferman (2002): *Alternating-time temporal logic*. *Journal of the ACM* 49(5), pp. 672–713, doi:10.1145/585265.585270.

[4] E. Best & C. Fernández (1988): *Nonsequential Processes*. Springer, doi:10.1007/978-3-642-73483-0.

[5] U. Buy, H. Darabi, M. Lehene & V. Venepally (2005): *Supervisory Control of Time Petri Nets Using Net Unfolding*. *Annual International Computer Software and Applications Conference* 2, pp. 97–100, doi:10.1109/COMPSAC.2005.148.

[6] J. Engelfriet (1991): *Branching processes of Petri nets*. *Acta Informatica* 28(6), pp. 575–591, doi:10.1007/BF01463946.

[7] J. Esparza (1994): *Model checking using net unfoldings*. *Science of Computer Programming* 23, pp. 151–195, doi:10.1016/0167-6423(94)00019-0.

[8] J. Esparza & K. Heljanko (2008): *Unfoldings – A Partial-Order Approach to Model Checking*. Springer, doi:10.1007/978-3-540-77426-6.

[9] B. Finkbeiner & S. Schewe (2005): *Uniform Distributed Synthesis*. In: *Proc. LICS*, IEEE Computer Society Press, pp. 321–330, doi:10.1109/LICS.2005.53.

[10] P. Gastin, B. Lerman & M. Zeitoun (2004): *Distributed Games with Causal Memory Are Decidable for Series-Parallel Systems*. In: *Proc. FSTTCS*, pp. 275–286, doi:10.1007/978-3-540-30538-5_23.

[11] B. Genest, H. Gimbert, A. Muscholl & I. Walukiewicz (2013): *Asynchronous Games over Tree Architectures*. In: *Proc. ICALP'13, Part II, LNCS* 7966, Springer, pp. 275–286, doi:10.1007/978-3-642-39212-2_26.

[12] A. Giua (1992): *Petri Nets as Discrete Event Models for Supervisory Control*. Ph.D. thesis, Rensselaer Polytechnic Institute.

[13] C.A.R. Hoare (1985): *Communicating Sequential Processes*. Prentice Hall, doi:10.1145/359576.359585.

[14] V. Khomenko, M. Koutny & W. Vogler (2003): *Canonical prefixes of Petri net unfoldings*. *Acta Informatica* 40, pp. 95–118, doi:10.1007/3-540-45657-0_49.

[15] O. Kupferman & M.Y. Vardi (2001): *Synthesizing Distributed Systems*. In: *Proc. LICS*, IEEE Computer Society Press, pp. 389–398, doi:10.1109/LICS.2001.932514.

[16] P. Madhusudan, P.S. Thiagarajan & S. Yang (2005): *The MSO Theory of Connectedly Communicating Processes*. In: *Proc. FSTTCS'05, LNCS* 3821, Springer, pp. 201–212, doi:10.1007/11590156_16.

[17] E.W. Mayr (1981): *An algorithm for the general Petri net reachability problem*. In: *Proc. 13th ACM STOC*, ACM, pp. 238–246, doi:10.1145/800076.802477.

[18] J. Meseguer, U. Montanari & V. Sassone (1996): *Process versus unfolding semantics for Place/Transition Petri nets*. *TCS* 153, pp. 171–210, doi:10.1016/0304-3975(95)00121-2.

[19] M. Nielsen, G.D. Plotkin & G. Winskel (1981): *Petri Nets, Event Structures and Domains, Part I*. *Theor. Comput. Sci.* 13, pp. 85–108, doi:10.1016/0304-3975(81)90112-2.

[20] E.R. Olderog (1991): *Nets, Terms and Formulas: Three Views of Concurrent Processes and Their Relationship*. Cambridge University Press, doi:10.1017/CBO9780511526589.

[21] A. Pnueli & R. Rosner (1990): *Distributed Reactive Systems are Hard to Synthesize*. In: *Proc. FOCS*, IEEE Computer Society Press, pp. 746–757, doi:10.1109/FSCS.1990.89597.

[22] J.F. Raskin, M. Samuelides & L.V. Begin (2003): *Petri Games are Monotone but Difficult to Decide*. Technical Report, Université Libre De Bruxelles.

[23] W. Reisig (1985): *Petri Nets – An Introduction*. Springer, doi:10.1007/978-3-642-69968-9.

[24] R. Rosner (1992): *Modular Synthesis of Reactive Systems*. Ph.D. thesis, Weizmann Institute of Science, Rehovot, Israel.

[25] L.J. Stockmeyer & A.K. Chandra (1979): *Provably Difficult Combinatorial Games*. *SIAM J. Comput.* 8(2), pp. 151–174, doi:10.1137/0208013.

[26] I. Walukiewicz & S. Mohalik (2003): *Distributed Games*. In: *Proc. FSTTCS'03, LNCS* 2914, pp. 338–351, doi:10.1007/978-3-540-24597-1_29.

[27] Q. Zhou, M. Wang & S.P. Dutta (1995): *Generation of optimal control policy for flexible manufacturing cells: A Petri net approach*. *Intern. Journal of Advanced Manufacturing Technology* 10, pp. 59–65, doi:10.1007/BF01184279.

[28] W. Zielonka (1995): *Asynchronous Automata*. In G. Rozenberg & V. Diekert, editors: *Book of Traces*, World Scientific, pp. 205–248, doi:10.1142/9789814261456_0007.