Available online at www.sciencedirect.com
ScienceDirect
Information and Computation 00 (2016) 1–27
Procedia Computer Science

Petri Games: Synthesis of Distributed Systems with Causal Memory[1]

Bernd Finkbeiner[a], Ernst-Rüdiger Olderog[b]

[a]*Department of Computer Science, Universität des Saarlandes, Saarbrücken, Germany*
[b]*Department of Computing, Universität Oldenburg, Oldenburg, Germany*

Abstract

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.

Keywords: Petri nets, causality, unfolding, cuts, strategies, graph games, synthesis

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.

[1]This research was partially supported by the German Research Foundation (DFG) in the Transregional Collaborative Research Center SFB/TR 14 AVACS. The paper is a revised and extended version of [1].

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 asynchronous automata [2], 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 [3, 4, 5]. Petri games thus abstract from the concrete content of a communication in that we assume that the processes always exchange the *maximal* possible 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.

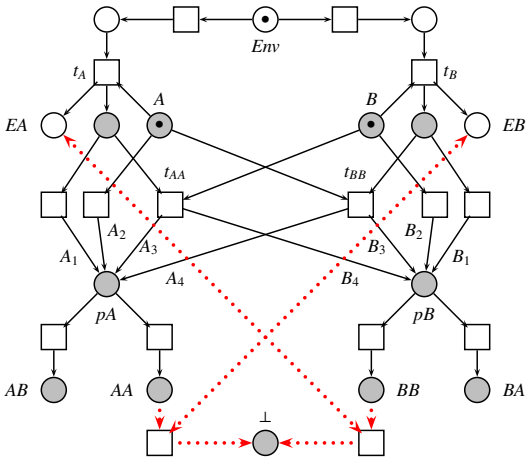

Figure 1. Introductory example of a Petri game modeling a distributed security alarm. Places belonging to the system players *A* and *B* are shown in gray. In the Petri game, the transitions to the bad place ⊥ are shown with dotted lines.

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

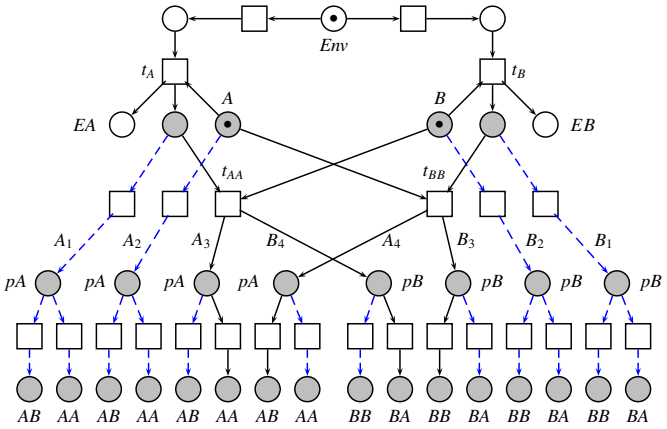

Figure 2. Unfolding of the Petri game in Fig. 1. To aid visibility, the transitions leading to $\perp$ are omitted from the unfolding. If the transitions shown with dashed lines are removed from the unfolding, the resulting net is a winning strategy for the system players.

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 the combination of BB and EA would constitute a false report. 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 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, 7, 8] 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 [9]. The case of Petri games with two or more environment tokens, i.e., situations with two or more *independent* information sources, remains open.

The role of the knowledge-based subset construction in the standard synthesis algorithms is to keep track of the relative informedness between the players. In order to avoid the nonelementary complexity that results from tracking separate (nested) knowledge sets for each player, it would be helpful to identify situations where all players have the *same* knowledge. Since the players in a Petri game exchange their causal histories during each synchronization, one might intuitively hope that such situations must occur repeatedly. Unfortunately, this intuition is incorrect, as the example in Fig. 3 demonstrates: after every firing of transition t_1 or t_3, the system player S_1 is better informed of the environment's decisions than S_2; after every firing of transition t_2 or t_4, the informedness of S_1 and S_2 is exactly reversed, and it is never the case that the two players have the same information.

It turns out that there are nevertheless situations where we can "pretend," without falsifying the result of the analysis, that the players have the same information. We call such situations *maximal cuts*, or short *mcuts*. They occur

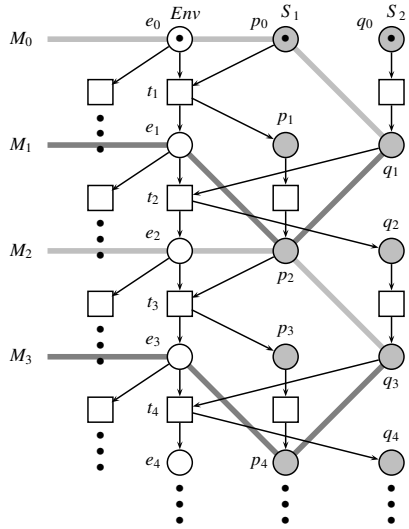

Figure 3. Part of an unfolding illustrating mcuts: the sets of places $M_0 - M_3$ are mcuts induced by the environment places $e_0 - e_3$, respectively.

when all system players that need to synchronize with the environment at some point in the future have run sufficiently far to be *blocked* waiting for the environment token to move. Since the next transition taken by each such player is a synchronization that causally depends on this environment transition, all players will, after their next move is played, have (at least) the information held by the environment token after the environment transition has been executed. And since they are blocked until this happens, giving them the information right away does not do any harm. In Fig. 3, the sets of places $M_0 = \{e_0, p_0, q_1\}$, $M_1 = \{e_1, p_2, q_1\}$, $M_2 = \{e_2, p_2, q_3\}$, and $M_3 = \{e_3, p_4, q_3\}$ are mcuts induced by the environment places $e_0 - e_3$, respectively. At M_2, system player S_2 is better informed of the environment's decisions than system player S_1, but we can safely assume that they are equally informed, because S_1 will obtain the missing information as soon as t_3 fires. Likewise, at M_1 and at M_3, S_1 is better informed than S_2, but we can safely assume that they are equally informed, because S_2 will obtain the missing information as soon as transition t_2 or transition t_4, respectively, fires. To solve the game, we partition each play into segments that lead from mcut to mcut; since each such segment plays out without participation of the environment, we can choose a shared strategy for the system players, which they execute deterministically until the next mcut is reached.

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 with safety objectives is EXPTIME-complete. In Section 7 we consider an application motivated by a traffic scenario. Related work and conclusions are presented in Sections 8 and 9.

2. Petri nets

We recall concepts from Petri net theory [10, 11, 12, 6, 7, 13, 14, 15, 8]. They are illustrated in Fig. 4 at the end of this section. 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 $\mathcal{N}$. (For notations on multisets see Appendix A.) 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* or *weight* k

if $F(p,t) = k$, and analogously, an arrow from a transition t to a place p is decorated by a *multiplicity* or *weight* k if $F(t,p) = k$. 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 putting $M(p)$ *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 $x\,F\,y$ if $F(x,y) > 0$. For a place p its *precondition* is the set $pre(p) = \{t \in T \mid t\,F\,p\}$, and its *postcondition* is the set $post(p) = \{t \in T \mid p\,F\,t\}$. For a transition t its *precondition* is the multiset $pre(t)$ over P defined by $pre(t)(p) = F(p,t)$ for $p \in P$, and its *postcondition* is the multiset $post(t)$ over P defined by $post(t)(p) = F(t,p)$ for $p \in P$. The multisets are needed when the transition t requests or puts back multiple tokens on some places. When stressing the dependency on the net $\mathcal{N}$, we write $pre^{\mathcal{N}}(x)$ and $post^{\mathcal{N}}(x)$ for nodes x instead of $pre(x)$ and $post(x)$. As in [7] we require *finite synchronization* [12] and non-empty pre- and postconditions: $pre(t)$ and $post(t)$ are finite, non-empty multisets for all transitions $t \in T$.

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 $R(\mathcal{N})$ and defined by $R(\mathcal{N}) = \{M \mid \exists\, t_1, \ldots, t_n \in T : In[t_1\rangle M_1[t_2\rangle \ldots [t_n\rangle M_n = M\}$. For a set $P \subseteq P$ of places and some $k \in \mathbb{N}$, a net $\mathcal{N}$ is *k-bounded on* P if $M(p) \leq k$ holds for all $M \in R(\mathcal{N})$ and all $p \in P$. It is *bounded on* P if it is k-bounded on P for some given k; otherwise it is *unbounded on* P. The net is *safe on* P if it is 1-bounded on P. If $P = P$, the set of all places, we drop the qualification '*on* P'.

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

We use the notations ${}^\circ\mathcal{N} = \{p \in P \mid pre(p) = \emptyset\}$ and $\mathcal{N}^\circ = \{p \in P \mid post(p) = \emptyset\}$ for the sets of places without incoming or outgoing transitions, respectively. For a multiset M over 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 $\mathcal{N} \restriction X = (P \cap X, T \cap X, F \restriction (X \times X), In \restriction X)$.

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 $P_1 \subseteq P_2$, $T_1 \subseteq T_2$, $F_1 \subseteq F_2$, and $In_1 = In_2$. A *homomorphism* from $\mathcal{N}_1$ to $\mathcal{N}_2$ is a mapping $h : P_1 \cup T_1 \to P_2 \cup T_2$ that preserves

- the type of the elements: $h(P_1) \subseteq P_2$ and $h(T_1) \subseteq T_2$;
- the pre- and postconditions of the transitions: $\forall t \in T_1 : h[pre(t)] = pre(h(t))$ 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, 7, 15, 8]. We follow the axiomatic presentation in [7], taking [14] into account for dealing with P/T Petri nets. An *occurrence net* is a Petri net $\mathcal{N}$, where

- $\forall\, t \in T$: $pre(t)$ and $post(t)$ are sets (rather than multisets);
- $\forall\, p \in P : |pre(p)| \leq 1$, i.e., each place has at most one incoming transition;
- the inverse flow relation F^{-1} is well-founded, i.e., starting from any node of $\mathcal{N}$ there does not exist an infinite path following the flow relation backwards;
- no transition $t \in T$ is in self-conflict;
- $In = {}^\circ\mathcal{N}$.

Note that an occurrence net is a safe net. If nodes $x \neq y$ of an occurrence nets are in conflict, they are mutually exclusive, i.e., there is a nondeterministic choice between x and y. Two nodes x, y of an occurrence net are *causally related* if $x\,F^*\,y$ or $y\,F^*\,x$. They are *concurrent* if they are neither causally related nor in conflict. If $x\,F^+\,y$ 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 $\lambda(t^U) = t$.

Note that even finite Petri nets may have infinite unfoldings due to cycles or unbounded many tokens generated.

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 [7] it 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. 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 $\lambda_2 = \lambda \restriction C^+$.

Causal nets and concurrent runs. Executions of Petri nets are represented by causal nets and concurrent runs as in [6, 12]. 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 that 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: $\mathcal{R}(\mathcal{N}) = \{\, [\beta_R\rangle \mid \beta_R \text{ is a finite initial run of } \mathcal{N} \,\}$.

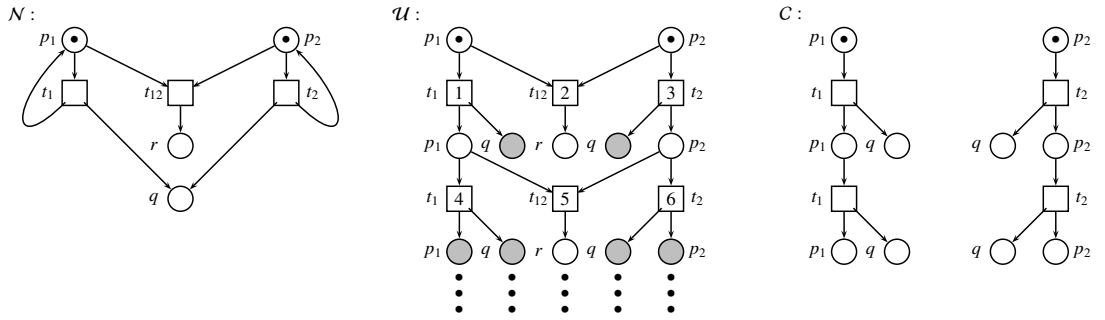

Figure 4. Illustration of Petri net concepts. *Left*: Example of a net $\mathcal{N}$ with places p_1, p_2, q, r and transitions t_1, t_2, t_{12}. The initial marking is $In = \{p_1, p_2\}$. The net is unbounded because the reachable markings can put unboundedly many tokens in place q. *Middle*: An initial part of the infinite unfolding $\mathcal{U}$ of $\mathcal{N}$ is shown. The names of the places and transitions of $\mathcal{N}$ appear as labels of the places and transitions of the unfolding. The two cycles and the common place q in the postconditions of the two transitions t_1 and t_2 in $\mathcal{N}$ are unfolded. Each place in $\mathcal{U}$ has at most one incoming transition, but it may have several outgoing transitions like p_1 and p_2, representing conflicts (nondeterminism). Transitions 1 and 2 are in conflict, transitions 4 and 5 are causally dependent on transition 1, transitions 1 and 3 are concurrent. The six gray places constitute a cut, i.e., a maximal set of pairwise concurrent places. *Right*: C is a causal net representing a finite concurrent run of $\mathcal{N}$. All nondeterminism present in $\mathcal{U}$ has been resolved: in C, each place has at most one outgoing transition. The marking of $\mathcal{N}$ reached by this run is M with $M(p_1) = M(p_2) = 1$, $M(q) = 4$, and $M(r) = 0$. It corresponds to the gray cut of $\mathcal{U}$.

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 finite set of places is partitioned into a subset $\mathcal{P}_S$ of *system places* and a subset $\mathcal{P}_E$ of *environment places*. Additionally, the Petri game identifies a set $\mathcal{B} \subseteq \mathcal{P}_S \cup \mathcal{P}_E$ 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 $\mathcal{G} = (\mathcal{P}_S, \mathcal{P}_E, \mathcal{T}, \mathcal{F}, In, \mathcal{B})$, where the (*underlying*) *Petri net of the game* $\mathcal{G}$ is $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, In)$, with a finite set of places $\mathcal{P} = \mathcal{P}_S \cup \mathcal{P}_E$. In the graphical representation, environment places are white and system places are gray. Players are modeled by the tokens of $\mathcal{N}$. A token on a system place represents a *system player*, and a token on an environment place an *environment player*. A Petri game $\mathcal{G}$ is (k-) *bounded* if $\mathcal{N}$ is; it has *at most k environment players* if $\mathcal{N}$ is k-bounded on $\mathcal{P}_E$.

Three examples illustrating the modeling opportunities with Petri games are shown in Example 5. The *processor allocation game* on the left is about *load balancing* of processors. In this game, four processes should be allocated on two processors so that no processor gets more than two processes allocated. Two of the allocations are by the two environment players and hence cannot be influenced, but the other two can be controlled by the two system players, which before their decision are informed by the environment's choices. The game in the middle models a *workflow*. It admits infinite plays and has *fork and join structure* for temporarily creating more players. The environment generates two types of tasks, which need to be processed correctly by the system player. In this game, for one (difficult) type of task the system player temporarily creates a second system player (fork) so that two system players concurrently work on that task. When this task has been accomplished successfully, the game continues with only one player (join). The game on the right is about *job processing*. In this game, the environment can generate *unboundedly many* orders that need to be processed by one out of two types of machines. The right choice of machine needs to be done by the system player.

Example 3.1. *Fig. 6 shows the underlying P/T net $\mathcal{N}$ of a small Petri game for two system players in place Sys and one environment player in place Env. 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.* □

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 $\mathcal{G}$ is the unfolding of the underlying net $\mathcal{N}$, denoted by the branching process $\beta_U = (\mathcal{N}^U, \lambda)$, where $\mathcal{N}^U$ is an occurrence net and λ is an initial homomorphism from $\mathcal{N}^U$ to $\mathcal{N}$, which "labels" the places and transitions of $\mathcal{N}^U$ with the places and transitions of $\mathcal{N}$. In the graphic representation of games and unfoldings gray places denote elements of $\mathcal{P}_S$ and white places elements of $\mathcal{P}_E$. For an example of an unfolding, see Fig. 7. Note that the place *Sys* with two tokens has been unfolded into several places labeled with *Sys*. This allows us to distinguish the causal past of the different occurrences of *Sys* in a play, in particular *Sys* before and after the transitions $test_1$ and $test_2$.

A strategy is obtained from the unfolding by deleting some of the branches that are under control of the system players. It describes for each place which transitions the player in that place can take. Formally, this is expressed by the net-theoretic notion of subprocess.

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

(S1) If $p \in \mathcal{P}^\sigma_S$ then σ is deterministic at p.
(S2) If $p \in \mathcal{P}^\sigma_E$ then $\forall\, t \in \mathcal{T}^U : (p,t) \in \mathcal{F}^U \wedge\ pre(t) \subseteq \mathcal{P}^\sigma_E \Rightarrow (p,t) \in \mathcal{F}^\sigma$, i.e., at an environment place the strategy does not restrict any purely environmental transition.
(S3) $\forall\, t \in \mathcal{T}^U : t \notin \mathcal{T}^\sigma \Rightarrow \exists\, p \in pre(t) \cap \mathcal{P}^U_S : (\forall t' \in post^U(p) : \lambda(t') = \lambda(t) \Rightarrow t' \notin \mathcal{T}^\sigma)$, i.e., if an instance t of a transition is forbidden by σ, then the reason is that from a place p in the precondition of t, σ uniformly forbids all instances t' of this transition.

Here $\mathcal{P}^\sigma_S = \mathcal{P}^\sigma \cap \lambda^{-1}(\mathcal{P}_S)$ denotes the system places and $\mathcal{P}^\sigma_E = \mathcal{P}^\sigma \cap \lambda^{-1}(\mathcal{P}_E)$ the environment places in $\mathcal{P}^\sigma$. A transition $t \in \mathcal{T}^\sigma$ with $pre(t) \subseteq \mathcal{P}^\sigma_S$ is called a *system transition*, and a transition $t \in \mathcal{T}^\sigma$ with $pre(t) \cap \mathcal{P}^\sigma_E \neq \emptyset$ is called

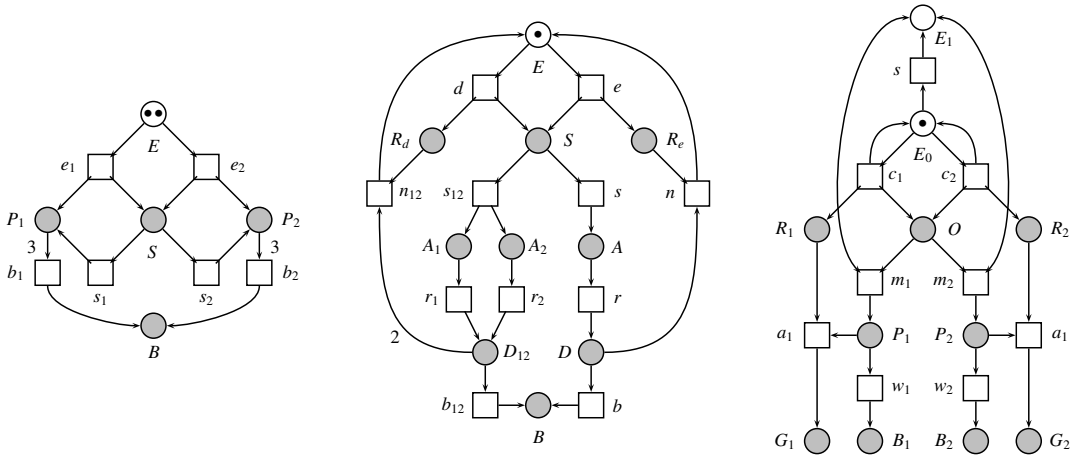

Figure 5. *Left*: *Processor allocation game.* First, two environment players in E decide via uncontrollable transitions e_1 or e_2 whether to place an environment process on processor P_1 or P_2. After each decision, the token from E is on the system place S, representing a system player that can now choose via controllable transitions s_1 or s_2 whether to place a system process on processor P_1 or P_2. Both processors should not handle more than two processes at a time. Therefore transitions b_1 and b_2 can put a token on the bad place B whenever three processes are placed on the same processor. To win the game, the system players should make the choices so that B is avoided. *Middle*: *Petri game with fork and join structure.* The environment player in E puts in place S one paper to review: a difficult one via transition d and an easy one via e. It keeps a record of this classification in place R_d or R_e, respectively. The system player, when activated by a paper in S, can choose to ask a single reviewer (via transition s to place A) or two independent reviewers (via transition s_{12} 'forking' to the places A_1 and A_2) to judge the paper. The work of the reviewers is represented by the transitions r and r_1, r_2, respectively. The two reviewers output their reviews in place D_{12}, which then has two tokens, and the single reviewer outputs its review in place D, which then has one token. Only if the system has made the right decision, i.e., asking two reviewers for the difficult paper and one reviewer for the easy paper, the transitions n_{12} and n, respectively, can fire to start a next round for the environment generating a new paper to review. Note that n_{12} 'joins' both tokens from D_{12} to deliver one token at E. This way the play may continue forever. If neither n_{12} nor n can fire, the only way for the system player(s) to avoid a deadlock is to fire transitions b_{12} or b, thereby reaching the bad place B. Then the system lost the game. *Right*: *Unbounded Petri game.* The environment player in place E_0 can generate unboundedly many orders by putting tokens on place O, either via choice c_1 or via choice c_2, and 'receipts' of these choices on place R_1 or R_2, respectively. When the environment player has generated enough orders, it moves via transition s to place E_1 and stops. Each token in O is now treated as a system player that has to process the order, either by 'machine' m_1, which puts the token on place P_1, or by 'machine' m_2, which puts the token on place P_2, for collecting the processed orders. Note that m_1 and m_2 can only fire when the environment player has stopped in E_1 due to the double arrows connecting E_1 with m_1 and with m_2. The processing is correct if for $i = 1, 2$ the number of activations of machine m_i agrees with the number of choices c_i by the environment. This is checked via the 'acknowledgments' a_1 and a_2, which put tokens on the 'good' places G_1 and G_2, respectively. If a_1 and a_2 cannot fire and a token in left in P_1 or P_2, only transitions w_1 or w_2 (for 'wrong' processing) will avoid a deadlock, at the price of putting a token on the bad places B_1 or B_2. Then the system players lost the game.

an *environment transition*. Thus a system transition depends only on the system players, whereas an environment transition depends (also) on the environment player. For a *purely environmental transition* t we require $pre(t) \subseteq \mathcal{P}_E^\sigma$. A special case is a *local environment transition* t, for which we require $|pre(t) \cap \mathcal{P}_E^\sigma| = 1$.

A strategy σ is *deterministic at a place* p if for all $M \in \mathcal{R}(\mathcal{N}^\sigma)$, the set of reachable markings in $\mathcal{N}^\sigma$, the following holds:

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

An example of strategy is shown on the right in Fig. 7. Local controllers are discussed in Section 4. Note that the determinism requirement does not mean that every place can only have one outgoing transition. For example, the upper *Sys* places in the strategy in Fig. 7 are both in the preset of transition $test_1$ and in the preset of transition $test_2$. The strategy is nevertheless deterministic in these places, because at every reachable marking only one of the transitions is enabled.

A (*concurrent*) *play* of a Petri game $\mathcal{G}$ is an initial concurrent run π of the underlying net $\mathcal{N}$. If π contains a place of $\mathcal{B}$, the *environment wins* π. Otherwise, the *system players win* π. For an example, see Fig. 8. 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. For example, the strategy shown in Fig. 7 is a winning strategy for the system players of the Petri game on the right-hand

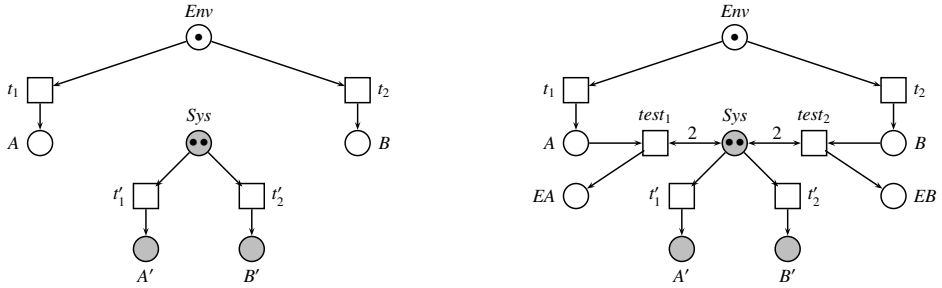

Figure 6. Two Petri games, where two system players, initially in place *Sys*, should achieve the same decisions as the environment, initially in place *Env*, i.e., when the environment chooses A, the system players should choose A', and when the environment chooses B, the system players should choose B'. *Left*: The system players cannot observe the decisions taken by the environment. Transitions from A and B' and from B and A' to a bad place have been omitted to aid visibility. *Right*: By communicating via transitions $test_1$ and $test_2$, the system players can learn about the decisions of the environment. Transitions from EA and B' and from EB and A' to a bad place have been omitted to aid visibility.

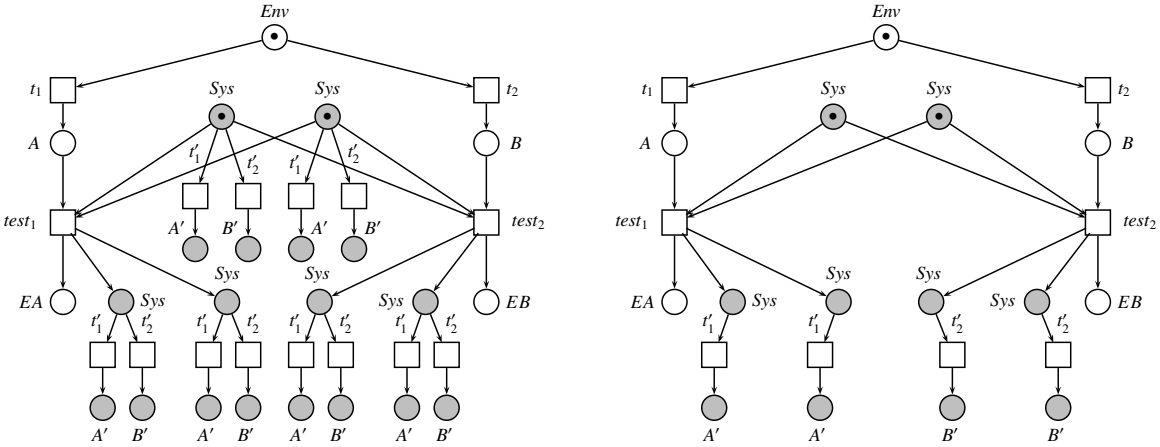

Figure 7. *Left*: Unfolding of the Petri game on the right-hand side of Fig. 6. *Right*: A strategy for the system players of this Petri game. When the environment chooses A, the system players after synchronizing with $test_1$ choose A', and when the environment chooses B, the system players after synchronizing with $test_1$ choose B'.

side of Fig. 6.

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 for all $M \in \mathcal{R}(\mathcal{N}^\sigma)$:

$$\exists\, t \in \mathcal{T}^U : pre(t) \subseteq M \Rightarrow \exists\, t \in \mathcal{T}^\sigma : pre(t) \subseteq M,$$

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 strategy σ is *non-terminating* if for all $M \in \mathcal{R}(\mathcal{N}^\sigma)$:

$$\exists\, t \in \mathcal{T}^\sigma : pre(t) \subseteq M.$$

Note that a non-terminating strategy is deadlock avoiding.

A *representation of a strategy* for the system players in $\mathcal{G}$ is a pair $\sigma = (\mathcal{N}^\sigma, h^\sigma)$ consisting of a safe net $\mathcal{N}^\sigma$ and an initial homomorphism h^σ from $\mathcal{N}^\sigma$ to $\mathcal{N}$ that is injective on transitions with the same preset, i.e., $\forall\, t_1, t_2 \in \mathcal{T}^U$: $pre(t_1) = pre(t_2) \wedge \lambda(t_1) = \lambda(t_2)$ implies $t_1 = t_2$, subject to the conditions (S1) – (S3) above. A representation of a strategy σ is *finite* if the set $\mathcal{P}^\sigma \cup \mathcal{T}^\sigma$ is finite. To achieve finiteness, a representation may have cycles.

In the remainder of this paper we focus on the special case, where Petri games have at most *one* environment player.

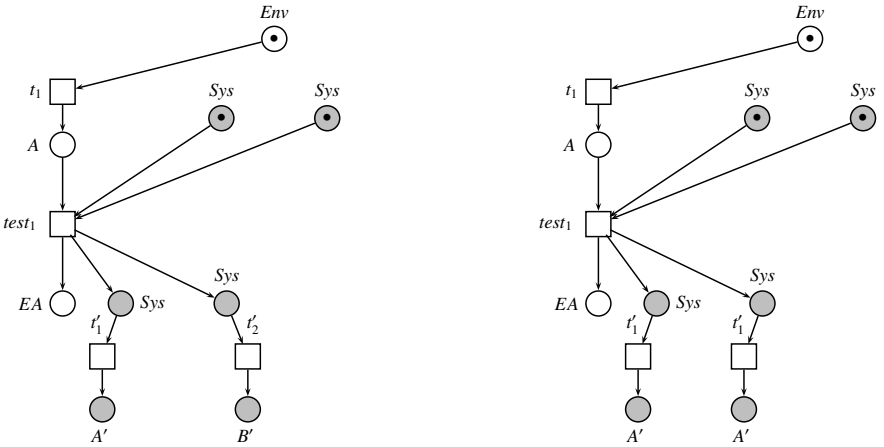

Figure 8. Two plays of the Petri game in Fig. 6, right, are shown. *Left*: The environment wins this play, which does not conform to the strategy shown in Fig. 7. *Right*: The system players win this play, which conforms to the strategy shown in Fig. 7.

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 $\mathcal{N}$ is *concurrency-preserving* if every transition $t \in \mathcal{T}$ satisfies $|pre(t)| = |post(t)|$. The *parallel composition* $\mathcal{N}_1 \,||\, \mathcal{N}_2$ of two nets $\mathcal{N}_i = (\mathcal{P}_i, \mathcal{T}_i, \mathcal{F}_i, In_i)$, $i = 1, 2$, with $\mathcal{P}_1 \cap \mathcal{P}_2 = \emptyset$ is defined as the Petri net $\mathcal{N}_1 \,||\, \mathcal{N}_2 = (\mathcal{P}_1 \cup \mathcal{P}_2, \mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{F}_1 \cup \mathcal{F}_2, In_1 \cup In_2)$ obtained by taking the componentwise union. The two nets synchronize on each common transition $t \in \mathcal{T}_1 \cap \mathcal{T}_2$ as in the process algebra CSP [16, 17].

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

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

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 $\mathcal{P}$.*

Proof: Consider $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, In)$ with $|In| = k$. Since $|pre^N(t)| = |post^N(t)|$ for all $t \in \mathcal{T}$ and $\mathcal{N}$ is reachable, the family $\mathcal{S}$ consists of k slices, say $S_1, \ldots, S_k$. We have to show $\mathcal{N} = S_1 \,||\ldots||\, S_k$. By the partition property of $\mathcal{S}$, we have $\mathcal{P} = \mathcal{P}_1 \cup \ldots \cup \mathcal{P}_k$ and $In = In_1 \cup \ldots \cup In_k$. By the definition of slices, $\mathcal{T}_1 \cup \ldots \cup \mathcal{T}_k \subseteq \mathcal{T}$. To show the converse inclusion, consider some $t \in \mathcal{T}$. Since $\emptyset \neq pre^N(t) \subseteq \mathcal{P}$, there exist $i \in \{1, \ldots, k\}$ and $p \in \mathcal{P}_i \cap pre^N(t)$ with $t \in post^N(p) \subseteq T_i$. Also, by the definition of slices, $\mathcal{F}_1 \cup \ldots \cup \mathcal{F}_k \subseteq \mathcal{F}$. To show the converse inclusion, consider some $(p, t) \in \mathcal{F}$. As before, there exists some $i \in \{1, \ldots, k\}$ with $p \in \mathcal{P}_i$ and thus $t \in T_i$. Hence $(p, t) \in \mathcal{F}_i$. Now consider some $(t, q) \in \mathcal{F}$. Then there exists some $i \in \{1, \ldots, k\}$ with $q \in \mathcal{P}_i$ and thus $t \in T_i$ (backward reasoning). □

A *local controller* specifies the moves of a single player in a Petri game. It is a pair $C = (\mathcal{N}^C, h^C)$ 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, i.e., with

- $h^C(\mathcal{P}^C) \subseteq \mathcal{P}$ and $h^C(\mathcal{T}^C) \subseteq \mathcal{T}$ and $h^C(In^C) \subseteq In$,
- $\forall p, q \in \mathcal{P}^C, t \in \mathcal{T}^C : (p, t), (t, q) \in \mathcal{F}^C \Rightarrow (h^C(p), h^C(t)), (h^C(t), h^C(q)) \in \mathcal{F}$,

satisfying the following conditions:

- if C is for a system player then $h^C(\mathcal{P}^C) \subseteq \mathcal{P}_S$,
- if C is for the environment player then $h^C(\mathcal{P}^C) \subseteq \mathcal{P}_E$ and $\forall p \in \mathcal{P}^C : h^C[post^C(p)] = post(h^C(p))$, i.e., all outgoing transitions (choices) of $\mathcal{N}$ are preserved in C.

A local controller 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 $C^U = (\mathcal{N}^{CU}, h^C \circ \lambda^C)$ 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:

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

Proof: By Lemma 4.1, $\mathcal{N}^\sigma$ is the parallel composition of slices: $\mathcal{N}^\sigma = \|_{S \in \mathcal{S}} S$, where $\mathcal{S}$ is a family of slices of $\mathcal{N}^\sigma$ such that $\{\mathcal{P}^S \mid S \in \mathcal{S}\}$ is a partition of $\mathcal{P}^\sigma$. For each system player its slice S is also an unfolded local controller for this player. For the environment player, the unfolded local controller C_E is defined as follows: the transitions of C_E are all transitions that are possible according to the unfolding. The slice S_E for the environment approximates C_E. Replacing S_E by C_E in the parallel composition yields a net with possibly more sync transitions syntactically present, but none of these transitions can actually fire due to the failing synchronization with the slices of the system players. So the reachable part of the new parallel composition is isomorphic to $\mathcal{N}^\sigma$. □

Note that the slice S_E for the environment itself need not be an unfolded local controller for the environment because S_E may get stuck at a sync transition $t \in \mathcal{T}^U$, where one of the involved system players does not take part in it in σ. Formally, this is the case when $\exists\, p, q \in pre^U(t) : p \in \mathcal{P}^{S_E} \subseteq \mathcal{P}_E \wedge q \in \mathcal{P}_S \wedge (q \in \mathcal{P}^\sigma \Rightarrow (q, t) \notin \mathcal{F}^\sigma)$.

Example 4.3. *The strategy of Fig. 7 can be distributed into the local controllers of Fig. 9.* □

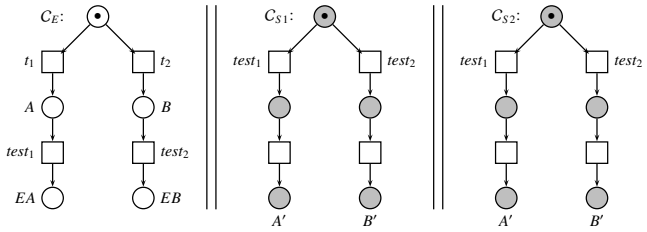

Figure 9. The local controllers C_E for the environment and C_{S1}, C_{S2} for the system players work in parallel and synchronize on the transitions $test_1$ and $test_2$. Applying the parallel composition $\|$ to the three controller nets yields the winning strategy of Fig. 7.

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 [13], 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 $x \in \mathcal{P}^\sigma \cup \mathcal{T}^\sigma$ 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.

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

$$fut^\sigma(x) = \{y \in \mathcal{P}^\sigma \cup \mathcal{T}^\sigma \mid x \leq y\}.$$

A *p-cut* is a cut containing the place p. For an environment place $p \in \mathcal{P}^\sigma$ we introduce now special p-cuts which we call *maximal cuts*, abbreviated *mcuts*. For all places q in such a p-cut, either (1) 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 (2) the future starting at q does not depend on the environment. The formal definition considers a p-cut C of an environment place $p \in \mathcal{P}^\sigma$ and a place $q \in C$. We say that q is a *type-1* place, for short $type(q) = 1$, if

$$\forall t \in post^\sigma(q) : (t \text{ reachable in } \mathcal{N}^\sigma_{C^+} \Rightarrow p \leq t).$$

Thus from a type-1 place q every reachable outgoing transition t is causally dependent on p, so the player in q can progress only when the environment player in p has progressed as well. Note that $type(p) = 1$.

We say that q is a *type-2* place, for short $type(q) = 2$, if

$$\forall p \in \mathcal{P}^\sigma_E : p \not\leq q \Rightarrow \forall x \in fut^\sigma(q) : p \not\leq x.$$

Thus for a type-2 place q we require that whenever it does not causally depend on an environment place p, none of the nodes in its future depends on p.

By *type-1(C)*, we denote the set of all places in C that have type 1, and analogously for *type-2(C)*. Then a p-cut of $\mathcal{N}^\sigma$ is called an *mcut*, more precisely a *p-mcut*, if

$$(\forall q \in C : type(q) = 1 \vee type(q) = 2)\ \wedge\ \mathcal{N}^\sigma_{type-2(C)+} \text{ is infinite.}$$

For an example, see Fig. 10. Type-2 places are shown in Fig. 15 in Section 7. For a p-cut C, we call $q \in C$ a *non-type-2* place if $type(q) \neq 2$. In general, the set of non-type-2 places of C is a superset of the set of type-1 places of C. If C is a p-mcut then the non-type-2 places of C coincide with the type-1 places.

An *ecut* results from an mcut by firing a single *environment* transition. Formally, given a p-mcut C for an environment place $p \in \mathcal{P}^\sigma$ and a transition $t \in post^\sigma(p)$ let $ecut(C, t)$ be the cut obtained by firing t at C, formally $C\ [t\rangle\ ecut(C, t)\ C'$. For an example, see Fig. 10.

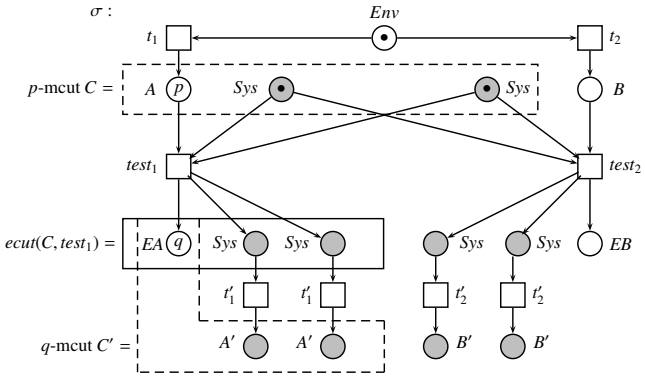

Figure 10. Shown is the strategy σ of Fig. 7. Consider the places p labeled with A and q labeled with EA. The p-mcut C contains the upper places labeled A, Sys, Sys, and $ecut(C, test_1)$ contains the places labeled EA, Sys, Sys in the middle, whereas q-mcut C' contains the places labeled EA, A', A', with the system players maximally progressed. Both mcuts have only type-1 places.

6. Deciding Petri Games

Petri games are based on net-theoretic concepts. The unfolding of the underlying net provides the information that the players have of each other. Strategies are obtained by restricting the nondeterminism present in the unfolding. Since for games with cycles the unfoldings are usually infinite, so are the strategies. Winning strategies are those that admit only plays (concurrent runs) without any bad places.

We now address the question whether there exists a winning strategy for a given Petri game. We wish to show that for bounded Petri games with one environment players this question is decidable and determine the complexity of the decision process. To solve this question directly on Petri games we would have to construct finite representations of the strategies. These could be obtained by identifying repetitions in the infinite strategies. We have been pursuing this approach by using a suitable notion of equivalence on mcuts, along which we would fold the strategies. Since the details were getting rather intricate, we looked for an alternative approach, which we pursue in this paper.

It is based on a translation of Petri games into Büchi games over finite graphs, for which it is known that they can be solved in polynomial time in the number of edges in the graph [18]. The crux is how to translate the information of the players in the Petri game in a suitable way to the states of the graph game and to limit the size of the graph.

6.1. Büchi Games

Unlike the Petri game, the Büchi game has only two players, Player 0 and Player 1, which both act on complete information. We construct a Büchi 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 if and only if Player 0 has a winning strategy in the Büchi 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.

Formally, a *Büchi game* $\mathbb{G} = (V, V_0, V_1, I, E, F)$ 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 a set $F \subseteq V$ of *accepting states*. 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 visits some state in F infinitely often. Otherwise, Player 1 wins. We assume that every state has an outgoing edge, i.e., for all $v \in V$ there is a $v' \in V$ such that $(v, v') \in E$.

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.

Example 6.1. *How can a Büchi game over a finite graph, where the players act on complete information simulate a Petri game, where the players see only the causal past of their decisions? Let us look at a version of the Petri game on the left-hand side of Fig. 6, where for simplicity we assume that there is only one system player, i.e., one token in the place Sys. The goal is that the system player achieves the same choice as the environment. The Büchi game explores a state set that corresponds to a part of the set of all markings of the Petri game if we forget for a moment about the annotations needed for decision sets. Fig. 11 shows the graph of all reachable markings. The initial marking is* $\{Env, Sys\}$. *Transitions due to the system player are drawn with solid arrows, and transitions due to the environment are drawn with dashed arrows. There are two bad states (markings), namely* $\{B, A'\}$ *and* $\{A, B'\}$. *The set of accepting states is given by* $F = \{\{A, A'\}, \{B, B'\}\}$. *The states in F and the bad states have self-loops.*

Suppose for a moment, the system player can see all markings in the graph. Then it has the following winning strategy. The system player waits until the environment has taken its decision to move from Env to A or B. Afterwards the system player copies this choice by moving from Sys to the corresponding A' *or* B', *respectively.*

However, in our Büchi game the environment can only move at markings corresponding to mcuts. These markings are decorated by a superscript m in Fig. 11. Informally, this means that the Büchi game consists only of the initial marking $\{Env, Sys\}$ *and the six (markings corresponding to) mcuts. The system player is allowed to wait only at mcuts. So it has to move out of the initial marking by one of the transitions* t'_1 *or* t'_2 *to* A' *or* B'. *No matter where the system player moves to, the environment can react with a transition entering a bad marking.* □

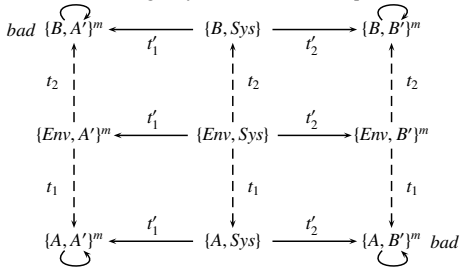

Figure 11. Graph showing all reachable markings, where those corresponding to mcuts are decorated by a superscript m.

6.2. Constructing a Büchi Game

We give an informal preview of the construction by looking at an example.

Example 6.2. *Figure 12 shows (a part of) the Büchi game corresponding to the Petri game on the right-hand side of Fig. 6, where the transitions leading to bad places were omitted. For the purposes of this example, we assume that there is one additional transition $t_\perp$ in the Petri game, 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. Accepting states for Player 0 are shown with double lines. In addition to the initial state v_0 shown in Fig. 12, the game has further initial states that are omitted here. Each state shows a so-called decision set*, which is an enriched marking of the Petri game. State v_0 represents the initial marking $\{Env, Sys, Sys\}$, enriched by the information that the environment player in place Env is committed to choosing t_1 or t_2 as its next transition, and the two system players in place Sys are committed to choosing $test_1$ or $test_2$ next. The flags 'false' signal that none of the places is a type-2 place, which would require a special treatment. Consider the edge from v_0 to v_1 performed by Player 1: it corresponds to the transition t_1 taken by the environment player in the Petri game. State v_1 shows the decision set after firing that transition: the commitments of the system players are the same as in v_0, but the environment player in place A is now committed to perform $test_1$ as its next transition. According to their decision sets neither in v_0 nor in v_1 a system player could fire a transitions without participation of the environment. Hence these states correspond to mcuts in the unfolding of the Petri game and belong to Player 1.*

All other states shown in Fig. 12 belong to Player 0 because the system players can perform a transition on their own. For example, in state v_2 the symbol $\top$ indicates that the system players have to select new commitments, some of which are shown in the states $v_3, \ldots, v_7$. In state v_4 each of the two system players is committed to the transition t'_1. Firing these two transitions is represented by the edges to state v_8 and from there to state v_9.

Player 0 has a winning strategy from v_0 (following the edges shown with solid lines, thus visiting states v_0, v_1, v_2, v_4, v_8, and v_9). In state v_9, Player 0 wins, because the Petri game terminates. In states v_3 and v_{10}, Player 1 wins, because a deadlock is reached. In state v_7, Player 1 wins, because nondeterminism is encountered. In states v_{11}, v_{12}, and v_{13}, Player 1 wins, because the bad place $\perp$ is reached. From each of these states a self-loop starts to permit infinite plays. □

Now we make the above discussion formal. Consider a k-bounded Petri game $\mathcal{G} = (\mathcal{P}_S, \mathcal{P}_E, \mathcal{T}, \mathcal{F}, In, \mathcal{B})$ with underlying net $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, In)$ and places $\mathcal{P} = \mathcal{P}_S \cup \mathcal{P}_E$. By k-boundedness, the multiplicity of each place $p \in \mathcal{P}$ in each reachable marking of $\mathcal{N}$ is at most k.

We construct a Büchi game $\mathbb{G} = (V, V_0, V_1, I, E, F)$ that should simulate $\mathcal{G}$. The set V of states is given by a set of enriched markings of $\mathcal{N}$, where "enriched" means that each occurrence of a place $p \in \mathcal{P}$ carries some extra information. Formally, instead of markings of $\mathcal{N}$, i.e., multisets $M : \mathcal{P} \to \{0, \ldots, k\}$, we consider now so-called *decision sets* of $\mathcal{G}$, i.e., multisets

$$D : \mathcal{P} \times \{true, false\} \times (2^{\mathcal{T}} \cup \{\top\}) \to \{0, \ldots, k\}$$

of triples $(p, type\text{-}2, T)$, where *type*-2 is a Boolean flag indicating whether p is a type-2 place and $T \in 2^{\mathcal{T}} \cup \{\top\}$ represents a *commitment* for the next moves that can be made by a token in p. A commitment is either a set of transitions chosen by a token in p or the special symbol $\top$ (top), indicating that a new choice needs to be made. We

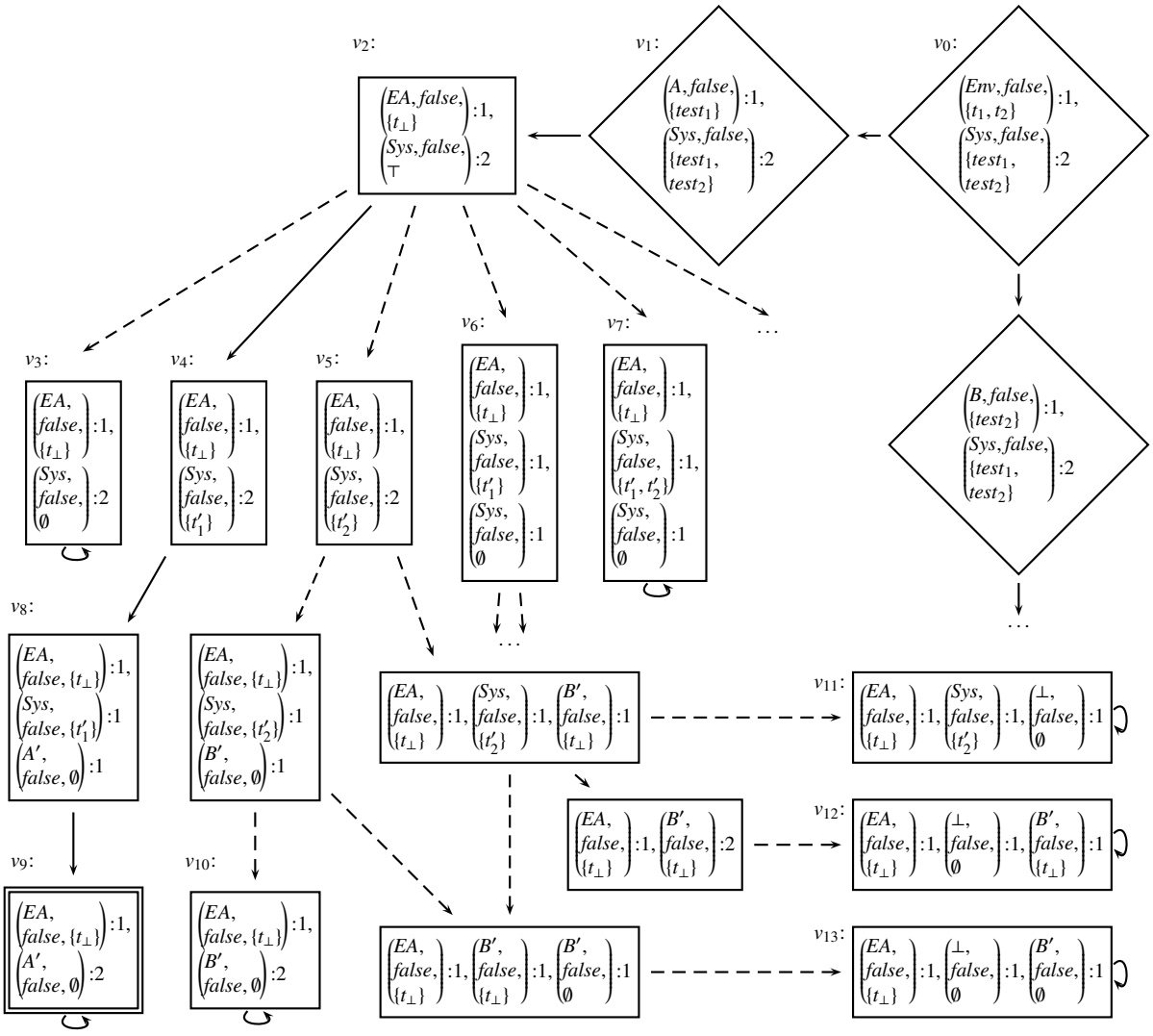

Figure 12. Part of the Büchi game corresponding to the Petri game from Fig. 6, right.

require that only commitments $T \in 2^{post^{\mathcal{N}}(p)} \cup \{\top\}$, i.e., subsets of the *outgoing* transitions from p in $\mathcal{N}$ or $\top$, occur in a triple $(p, type\text{-}2, T) \in D$. Altogether

- the set V of states on $\mathbb{G}$ is defined as the set of all such decision sets of $\mathcal{G}$.

Before we can continue with the definition of $\mathbb{G}$, we need some auxiliary concepts on decision sets. From a decision set D of $\mathcal{G}$ we can go back to a marking of $\mathcal{N}$ by projecting to the places occurring in the triples of D. The result we call the *underlying marking* denoted by $M(D)$. Formally, $M(D) : \mathcal{P} \to \{0, \ldots, k\}$ is defined by adding the multiplicities of triples in which p occurs:

$$M(D)(p) = \sum_{type\text{-}2 \in \{true,false\},\ T \in 2^T \cup \{\top\}} D(p, type\text{-}2, T).$$

Since we consider only Petri games with one environment player, there is at most one environment place in each marking $M(D)$. A decision set D with environment place $p \in M(D)$ *corresponds to an mcut* if there is no $\top$ symbol in

D and for all triples $(q, \mathit{false}, T) \in D$ and transitions $t \in T$ the following holds: t is not enabled at $M(D)$ or it contains p in its precondition, i.e., $p \in \mathit{pre}^{\mathcal{N}}(t)$.

We distinguish the enabledness of transitions at decision sets and at markings. A transition $t \in \mathcal{T}$ is *enabled at a decision set* D if $\mathit{pre}^{\mathcal{N}}(t) \subseteq M(D_t)$, where D_t is the multiset consisting of all triples $(p, \mathit{type}\text{-}2, T) \in D$ with $p \in \mathit{pre}^{\mathcal{N}}(t)$ and $t \in T$. Thus t has to occur in the commitments of all triples in D with places in t's precondition. If a transition is enabled at D, it is also enabled at the marking $M(D)$, but not vice versa.

- D *contains a bad place* if $\exists\, q : q \in M(D) \cap \mathcal{B}$.
- D is a *deadlock* if a transition of $\mathcal{N}$ is enabled at $M(D)$, but no transition is enabled at D.
- D is *terminating* if no transition of $\mathcal{N}$ is enabled at $M(D)$.
- D *is type2-incorrect* if the multiset of places that are flagged as type-2 is not a correct type2-marking, i.e., $M_2(D) \notin \mathcal{M}_2$, where $\mathcal{M}_2$ is defined in Subsection 6.3 and $M_2(D)$ is defined as follows:

$$M_2(D)(p) = \sum_{T \in 2^T \cup \{\top\}} D(p, \mathit{true}, T).$$

- D is *nondeterministic* if two separate transitions t_1 and t_2 of $\mathcal{N}$ that share a system place in their precondition are enabled at D, or if a single transition t of $\mathcal{N}$ with a system place in its precondition is enabled at D such that $\mathit{pre}^{\mathcal{N}}(t) \neq M(D_t)$. In the second case, two different instances of t can fire at $M(D)$.

Now we can continue with the definition of $\mathbb{G}$.

- The set V_1 of states belonging to Player 1 consists of all decision sets of V that correspond to an mcut.
- The set V_0 of states belonging to Player 0 consists of all other decision sets: $V_0 = V \setminus V_1$.
- The set I of initial states consists of all decisions sets D with $M(D) = \mathit{In}$, the initial marking of $\mathcal{N}$, such that for all triples $(p, \mathit{type}\text{-}2, T)$ the type-2 is a Boolean flag indicating a type-2 place, and the commitment T consists of all outgoing transitions of p in $\mathcal{N}$ if $p \in \mathcal{P}_E$ and an arbitrary subset of the outgoing transitions of p in $\mathcal{N}$ if $p \in \mathcal{P}_S$.
- The set E of edges is defined as follows. If D contains a bad state, or is a deadlock, terminating, type2-incorrect, or nondeterministic, the game enters a self-loop. From all other states, the edges are defined below.

  First consider a V_1-state of Player 1, i.e., a decision set D that corresponds to an mcut, say with the environment place $p \in M(D)$. An edge from D is obtained by executing one environment transition t, i.e., with $p \in \mathit{pre}^{\mathcal{N}}(t)$, that is enabled at D. Let M' be the marking with $M(D)[t\rangle M'$ in $\mathcal{N}$, i.e., $M' = M(D) - \mathit{pre}^{\mathcal{N}}(t) + \mathit{post}^{\mathcal{N}}(t)$. From M' construct a state D' by the following annotations. For the environment place $p' \in M'$, set the type-2 flag to false and the commitment T to the set of all outgoing transitions from p'. For each system place $q \in \mathit{post}^{\mathcal{N}}(t)$, take the symbol $\top$ and set the type-2 flag to true if it was set on some system place in the precondition, and to false otherwise. For each system place $q \in M(D) - \mathit{post}^{\mathcal{N}}(t)$ keep the annotations of D. Then add an edge from D to D'.

  Next consider a V_0-state of Player 0, i.e., a decision set D, say with the environment place $p \in M(D)$, that does *not* correspond to an mcut. We distinguish two cases. If D contains a $\top$ symbol in some triple $(q, \mathit{type}\text{-}2, \top)$ of a system place q, choose a set $T \subseteq \mathit{post}^{\mathcal{N}}(q)$ of transitions, and add an edge from D to a state D' that results from D by replacing $(q, \mathit{type}\text{-}2, \top)$ with $(q, \mathit{type}\text{-}2', T)$, where $\mathit{type}\text{-}2'$ must be *true* if type-2 is *true* and can be *true* or *false* otherwise. Otherwise, execute one system transition t that is enabled at D such that $\mathit{pre}^{\mathcal{N}}(t)$ contains neither p nor any type-2 place. Again, let M' be the marking with $M(D)[t\rangle M'$ in $\mathcal{N}$, i.e., $M' = M(D) - \mathit{pre}^{\mathcal{N}}(t) + \mathit{post}^{\mathcal{N}}(t)$. From M' construct a state D' by the following annotations. For each system place $q \in \mathit{post}^{\mathcal{N}}(t)$ take the symbol $\top$ and set the type-2 flag to true if it was set on some system place in the precondition, and to false otherwise. For each place $q \in M(D) - \mathit{post}^{\mathcal{N}}(t)$ keep the annotations of D. Then add an edge from D to D'.
- The set F of accepting states consists of all decision sets that are terminating, contain a type-2 place or correspond to an mcut, and are not a deadlock, are not nondeterministic, do not contain a bad place, and are not type-2 incorrect.

6.3. Type-2 Analysis

We now describe how to compute the candidates for type-2 places in winning strategies. Recall that these are the places from which no further interaction with the environment is needed. Consider a Petri game $\mathcal{G} = (\mathcal{P}_S, \mathcal{P}_E, \mathcal{T}, \mathcal{F}, In, \mathcal{B})$ with underlying net $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, In)$ and places $\mathcal{P} = \mathcal{P}_S \cup \mathcal{P}_E$. For a marking M of $\mathcal{N}$, we define the modified game $\mathcal{G}_M$ by $\mathcal{G}_M = (\mathcal{P}_S, \mathcal{P}_E, \mathcal{T}, \mathcal{F}, M, \mathcal{B})$, where the initial marking is replaced by M.

A *type-2 strategy* of $\mathcal{G}$ is a non-terminating strategy that does not contain any environment places. A *correct type-2 marking* is a marking M of $\mathcal{N}$, such that the modified game $\mathcal{G}_M$ has a winning *type-2 strategy*. Let $\mathcal{M}_2$ be the set of correct type-2 markings.

A set $\mathcal{M}$ of markings of $\mathcal{N}$ is called *closed* if it does not contain environment and bad places, such that, for every marking $M \in \mathcal{M}$, there is a system transition t and a marking $M' \in \mathcal{M}$ with $M[t\rangle M'$. Let $\mathcal{M}^c$ be the largest closed subset of markings of $\mathcal{N}$.

Lemma 6.3. *The two sets of markings coincide: $\mathcal{M}_2 = \mathcal{M}^c$.*

Proof: (1) $\mathcal{M}_2 \subseteq \mathcal{M}^c$. We show that $\mathcal{M}_2$ is closed. Consider some $M \in \mathcal{M}_2$. Then there exists a winning type-2 strategy σ in the modified game $\mathcal{G}_M$. Since σ is non-terminating and does not contain environment places, there exists a system transition t from some place in M in the initial cut of σ. Let M' be the successor marking, i.e., $M[t\rangle M'$. Then $M' \in \mathcal{M}_2$ as well, because the sub-strategy of σ that starts in the successor cut reached after firing t is a winning type-2 strategy in the modified game $\mathcal{G}_{M'} = (\mathcal{P}_S, \mathcal{P}_E, \mathcal{T}, \mathcal{F}, M', \mathcal{B})$.

(2) $\mathcal{M}^c \subseteq \mathcal{M}_2$. Suppose now, by way of contradiction, that $\mathcal{M}^c \not\subseteq \mathcal{M}_2$. Then there is a marking $M \in \mathcal{M}^c$ but $M \notin \mathcal{M}_2$. We construct a winning type-2 strategy for the modified game $\mathcal{G}_M$ and thus prove that M is, contrary to the assumption, a correct type-2 marking. The strategy is constructed inductively with the initial marking M as the initial cut of the strategy. Suppose that the strategy has been constructed up to cut C with $\lambda[C] \in \mathcal{M}_2$. Then there is a system transition t and a marking M' with $M[t\rangle M'$. We extend the strategy with new places NP such that $\lambda[NP] = post(t)$ and continue with the cut through the new places NP and the places in C that did not participate in t. The resulting process σ is a strategy: by construction, every place has at most one outgoing transition; hence, σ is deterministic at every system place (condition S1), and for every forbidden instance of a transition there exists a system place in the precondition where all instances of the transition are forbidden (condition S3), namely the place that has a different outgoing transition; finally, σ does not restrict any local environment transitions (condition S2), because σ does not contain any environment places. The strategy σ is winning, because it does not include any bad places, and is non-terminating, and, hence, deadlock avoiding, by construction. □

Lemma 6.4. *The set of correct type-2 markings $\mathcal{M}_2$ can be computed in exponential time.*

Proof: By Lemma 6.3, $\mathcal{M}_2 = \mathcal{M}^c$ holds. The set $\mathcal{M}^c$ can be constructed in a largest fixed point iteration, starting with the set of markings of $\mathcal{N}$ that do not contain bad places, and eliminating, in each step of the iteration, those markings from which there is no system transition back into the set. Since there are exponentially many markings, and each step of the iteration eliminates at least one marking, the running time of this construction is at most exponential. □

Lemma 6.5. *Let the Petri game $\mathcal{G}$ be k-bounded. Then for every correct type-2 marking M of the underlying net of $\mathcal{G}$, there is a representation of a winning type-2 strategy for the modified game $\mathcal{G}_M$ of at most exponential size.*

Proof: Since $\mathcal{G}$ is k-bounded, the set of markings in $\mathcal{G}_M$ consists of the (exponentially many) functions from $\mathcal{P}_S$ to $\{0, 1, \ldots, k\}$. We modify the construction of the winning type-2 strategy in the proof of Lemma 6.3, inclusion (2), such that only places from the finite set $\mathcal{P}_S \times (\mathcal{P}_S \to \{0, 1, \ldots, k\})$ are used. The inductive construction begins with the places in $M \times \{M\}$, where $\lambda(m, M) = m$ for all $(m, M) \in M \times \{M\}$. Suppose that the strategy has been constructed up to cut C with $\lambda[C] \in \mathcal{M}_2$. Then there is a system transition t and a marking M' with $M[t\rangle M'$. We add an instance of transition t leading to places $NP = post(t) \times \{M'\}$. The construction stops when the same marking is encountered for the second time. Since there are only exponentially many places, the constructed strategy has at most exponential size. As in Lemma 6.3, the resulting process σ is a strategy: by construction, every place has at most one outgoing transition; hence, σ is deterministic at every system place (condition S1), and for every forbidden instance of a transition there exists a system place in the precondition where all instances of the transition are forbidden (condition S3), namely the place that has a different outgoing transition; finally, σ does not restrict

any local environment transitions (condition S2), because σ does not contain any environment places. The strategy σ is winning, because it does not include any bad places, and is non-terminating, and, hence, deadlock avoiding, by construction. □

6.4. Properties of the Büchi Game

We now analyze the properties of the Büchi game constructed above.

Lemma 6.6 (From Büchi Game Strategies to Petri Game Strategies). *If Player 0 has a winning strategy in the Büchi game then the system players have a deadlock avoiding winning strategy in the Petri game.*

Proof: We translate a winning strategy for Player 0 in the Büchi game into a deadlock avoiding winning strategy for the system players in the Petri game. Let f be a winning strategy for Player 0 of the Büchi game $\mathbb{G} = (V, V_0, V_1, I, E, W_0, W_1)$ simulating the Petri game $\mathcal{G}$. Starting from an initial decision set $D_0 \in I$, the strategy f generates a (possibly infinite) tree T_f, where the nodes are labeled with states from V. The root is labeled with D_0. A node labeled with a V_1-state D has for each successor state D' in the edge relation E one successor node in T_f labeled with D'. A node N labeled with a V_0-state D has a unique successor node labeled with the state $f(w)$, where w is the sequence of labels from the root to N.

We traverse the nodes of T_f in a breadth-first order. Thereby we inductively construct a strategy $\sigma = (\mathcal{N}^\sigma, \lambda)$ for $\mathcal{G}$ and associate to each node labeled with a state D a cut C in the strategy σ under construction such that $\lambda[C] = M(D)$ holds. To D_0 we associate a cut C_0 labeled with the initial marking of $\mathcal{N}$, i.e., $\lambda[C_0] = In$. Then $\lambda[C_0] = M(D_0)$. Suppose now the breadth-first traversal has reached a node in T_f labeled with a V_1-state D to which the cut C is associated. Then for each successor node in T_f labeled with D' we consider the corresponding environment transition t and extend the strategy with new places NP such that $\lambda[NP] = post(t)$. We associate to D' the cut C' through the new places NP and the places in C that did not participate in t.

If the traversal has reached a node in T_f labeled with a V_0-state D with associated cut C, consider the unique successor node in T_f labeled with D'. If the edge from D to D' resolves a $\top$ symbol in D, nothing is added to the strategy, and C is also associated to D'. If the edge from D to D' corresponds to a transition t, we extend the strategy as above, arriving at a cut C' associated to D'. If C' contains places that are flagged as type-2 in D' and are reached for the first time, we add at C' the winning type-2 strategy computed in Lemma 6.5.

We show that σ is *deterministic* at every system place (condition S1). Suppose there exists a system place $p \in \mathcal{P}_S^\sigma$ such that σ is not deterministic at p. By definition, there exist a reachable marking M in $\mathcal{N}^\sigma$ and two different transitions $t_1, t_2 \in \mathcal{T}^\sigma$ that share p in their precondition and are enabled at M.

Case 1. Suppose there exists one cut C encountered during the construction of σ with $pre(t_1), pre(t_2) \subseteq C$.

C corresponds to a decision set D with $\lambda(p) \in M(D)$. If $\lambda(t_1), \lambda(t_2)$ are two different transitions sharing the system place $\lambda(p)$ in their precondition that are enabled at D. Thus D is nondeterministic. Contradiction.

If t_1 and t_2 are instances of the same transition $t \in \mathcal{T}^\mathcal{N}$, i.e., $\lambda(t_1) = \lambda(t_2) = t$, with $\lambda(p) \in pre^\mathcal{N}(t)$ then $pre(t_1) \neq pre(t_2)$ by the definition of unfolding, but $\lambda[pre(t_1)] = \lambda(pre[t_2])$. Then $\exists\, q' : q' \in pre(t_1) \wedge q' \notin pre(t_2)$ (or vice versa for t_1 and t_2) and $\exists\, q'' \in pre(t_2) : \lambda(q') = \lambda(q'') \wedge q' \neq q''$. Thus in C there are two different places q' and q'' with $\lambda(q') = \lambda(q'')$ and $q' \in pre(t_1)$ and $q'' \in pre(t_2)$. Hence in $\lambda[C]$ there is a place $q = \lambda(q') = \lambda(q'') \in pre^\mathcal{N}(t)$ with multiplicity of at least 2. Then the transition t is enabled at D with $pre^\mathcal{N}(t) \neq M(D_t)$, i.e., $M(D_t)$ has more tokens than needed to enable t, so t can fire in two different ways. Thus D is nondeterministic. Contradiction.

Case 2. Suppose there exist two different cuts C_1 and C_2 such that during the construction of σ, first t_1 was introduced at C_1 and then t_2 was introduced at C_2 with $pre(t_1) \subseteq C_1$ and $pre(t_2) \subseteq C_2$, but $pre(t_1) \not\subseteq C_2$ and $pre(t_2) \not\subseteq C_1$.

We again distinguish two cases. *Case 2a* concerns the situation that the node in T_f that produced C_2 is *below* the node that produced C_1. Suppose the Büchi game corresponds to executing transitions $t'_1, \ldots, t'_n$ with $n \geq 1$ and $C_1 = C'_0[t'_1\rangle \ldots [t'_n\rangle C'_n = C_2$. If this sequence removes a token from $pre(t_1)$ and puts it on one place of $pre(t_2)$ to enable t_2 at C_2 then $pre(t_1)$ and $pre(t_2)$ are in conflict. Then there cannot exist a reachable marking M, where both t_1 and t_2 are enabled. Contradiction. Thus if this sequence removes a token from $pre(t_1)$, it puts it to a place outside of $pre(t_2)$. Consider the transition t'_i with $1 \leq i < n$ that first removes a token from a place $p' \in pre(t_1)$. Then p' cannot be an environment place because then t'_i would be an environment transition, which can only fire at an mcut. However, C'_{i-1} is not an mcut because there exist system transitions after t'_i producing C_2. So p' is a system place. Then another nondeterminism in σ is encountered at place p' with $pre(t_1), pre(t'_i) \subseteq C'_i$. Then we proceed as in Case 1.

Case 2b concerns the situation that the node n_2 that produced C_2 is in on a different branch of T_f than the node n_1 that produced C_1. Let C be the cut at the last common ancestor n of n_1 and n_2, and let t'_1 be the transition from the environment place in C towards C_1, and t'_2 the transition from the environment place in C towards C_2. All transitions added in branches leading from n to n_1 depend on t'_1, all transitions added in branches leading from n to n_2 depend on t'_2. Hence, all the transitions added in branches leading from n to n_1 are in conflict with all transitions added in branches leading from n to n_2. In particular, t_1 and t_2 are in conflict with each other and can thus not be enabled in the same marking.

The requirement that σ does not restrict local transitions at an environment place (condition S2) is satisfied, because the decision set corresponding to the mcut in which the local environment transition could be taken would be a deadlock. The requirement that if an instance of a transition t is forbidden by σ, then from some place p in the precondition of p, σ uniformly forbids all instances t' of this transition (condition S3), is satisfied for the following reason: if t is not in σ, then there must be a place p in its precondition, such that on all traversals that reach a decision set D and a cut C such that t is enabled in C, D forbids $\lambda(t)$ and hence also $\lambda(t')$.

The strategy σ is *winning* because the plays that conform to f avoid bad places. We show that σ is also *deadlock avoiding*. The winning strategy f of the Büchi game enables an infinite play that Player 0 wins. Hence infinitely many transitions are visited in the breadth-first processing of the tree T_f and added to the Petri game strategy σ. All transitions in σ can fire. Thus at any cut visited in σ, a transition can fire. Since the precondition $pre(t)$ of any transition t is included in one cut, there cannot be any reachable marking in σ where no transition can fire. Some transitions could stem from self-loops. However, in the winning play of Player 0, a self-loop is only possible if the play is visiting a terminating state, which corresponds to a terminating marking in σ. So, σ is indeed deadlock avoiding. □

Lemma 6.7 (From Petri Game Strategies to Büchi Game Strategies). *If the system players have a deadlock avoiding winning strategy in the Petri game then Player 0 has a winning strategy in the Büchi game.*

Proof: We translate a deadlock avoiding winning strategy for the system players in the Petri game into a winning strategy for Player 0 in the Büchi game. Given a deadlock avoiding winning strategy $\sigma = (\mathcal{N}^\sigma, \lambda)$ of the Petri game $\mathcal{G}$ and a prefix $w \in V^* \cdot V_0$ of a play of the Büchi game $\mathbb{G}$ simulating $\mathcal{G}$, we compute the choice $f(w)$ of a strategy f for $\mathbb{G}$ as follows. Let w^- result from w by deleting all states containing a $\top$ symbol.

Starting with the initial marking of $\mathcal{N}^\sigma$ and firing the sequence of transitions corresponding to w^- in σ, we arrive at a cut C of σ, which by the definition of V_0 does not correspond to an mcut.

Case 1. If $last(w) \in V_0$ contains a $\top$ symbol, perform a choice step to the decision set

$$dec[C] = \{(\lambda(p), type(p), \lambda(post^\sigma(p))) \mid p \in C\}.$$

Case 2. If $last(w) \in V_0$ contains no $\top$ symbol, then $last(w) = dec[C]$. Since C is not an mcut, some system transition is enabled at C. Choose one of the enabled system transitions, say t, as the next step after w in $\mathbb{G}$. Let t lead to a successor cut C'. Take as decision set of the new state the multiset

$$dec[C'] = \{(\lambda(p), type(p), \lambda(post^\sigma(p))) \mid p \in C'\}.$$

We now show that every play w of $\mathbb{G}$ that conforms to the constructed strategy f is winning, i.e., it visits some state from F infinitely often.

Since σ is winning, every play π conforming to σ avoids any bad place. Also, by definition, σ is deterministic at every system place, and by assumption, σ is deadlock avoiding. The play w therefore avoids the rejecting self-loops on states with nondeterministic decision sets and decision sets that are a deadlock or contain bad places. If w reaches a terminating decision set, then the play enters an accepting self-loop. Suppose now, by way of contradiction, that none of these conditions apply and w is still not winning, i.e., there is a position $m \geq 0$ such that for all positions $n \geq m$, $w(n)$ is not an mcut and none of the places in $w(n)$ is a type-2 place.

Let, for each $i \in \mathbb{N}$, C_i be the cut used in the construction of f to obtain $w(i)$. Let q be the environment place in C_n. Since none of the cuts C_n for $n \geq m$ is an mcut, and, hence, Player 1 never gets to make a move, q is the environment

place of all these cuts C_n. Also, since tokens on type-1 places cannot move unless the environment token moves, there must be an infinite sequence $p_m, p_{m+1}, \ldots$ of system places $p_m \in C_m$, $p_{m+1} \in C_{m+1}$, …, such that the places are neither type-1 nor type-2 and for all $n \geq m$, p_n and p_{n+1} are either the same place or connected by a transition, and for infinitely many n, p_n and p_{n+1} are different places connected by a transition.

We now show that the size of the cuts C_n grows beyond any bound. Since p_m is not a type-2 place, there exists an environment place $q' \not\leq p_m$ and a place $x \in fut^{\sigma}(p_m)$, i.e., with $p_m \leq x$, such that $q' \leq x$.

We first show that $q \leq q'$. Let $k \leq m$ be the first position where $q \in C_k$. If $k = 0$, then $q \leq q'$ obviously holds: since there is only one environment token, q is in the causal past of every environment place. If $k > 0$, then let $q_{k-1} \neq q$ be the environment place in C_{k-1}. Since the environment token has moved from C_{k-1} to C_k, C_{k-1} must be an mcut. All places in C_{k-1} must therefore be type-1 or type-2. Since p_m is not type-2, it cannot be the case that p_m is in the future of some type-2 place in C_{k-1}; there must, hence, exist some type-1 place $p_{k-1} \in C_{k-1}$ with $p_{k-1} \leq p_m$. By the definition of type-1, we therefore have that $q_{k-1} \leq p_m$, and, by transitivity, $q_{k-1} \leq x$. Since both q_{k-1} and q' are in the past of x, they cannot be in conflict. Since $q' \not\leq p_m$, also $q' \not\leq q_{k-1}$. Hence, $q_{k-1} \leq q'$ and $q_{k-1} \neq q'$. Since q is the single successor environment place of q_{k-1}, we therefore have that $q \leq q'$ also for the case of $k > 0$.

Having established $q \leq q'$, consider now the unique transition t in the precondition of x. This transition is in the future of p_m, in the future of q', and in the past of x. Since q is in all cuts $C_m, C_{m+1}, \ldots$ and $q \leq q'$, there is never a token on q'. Thus the transition t never fires. Since decision sets are deterministic, p_m has one outgoing transition, say t_m. Firing t_m puts a token on a place closer to t. If that place is p_{m+1} we repeat this process. Eventually, we reach a place p' that does not move any further, it remains in all cuts from some point m' onward. This place p' is in the intersection of the past of x and the future of p_m. We now choose an index m'' such that x is no longer in the future of $p_{m''}$ (this must be possible, because the sequence of p_n's is infinite and the past of x is finite).

Now we repeat the argument from $p_{m''}$ showing that there is *another* place p'' that remains in all cuts from some point m''' onward. p'' is different from p' because p'' is in the future of $p_{m''}$ while p' is not. By repeating this argument infinitely often, we show that the number of places in the cuts grows beyond any bound, which contradicts the assumption that $\mathcal{N}$ is bounded.

Thus w must be winning. Hence, f is winning from the decision set of the initial cut in 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 [9]. The details are given in the proof of the following theorem.

Theorem 6.8 (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 deadlock avoiding winning strategy is EXPTIME-complete. If a winning strategy for the system players exists, it can be constructed in exponential time.*

Proof: The *upper bound* is established by estimating the size of the set V of states in the Büchi game. Assume the underlying Petri net is k-bounded by some $k \geq 1$. For a given place p we represent a triple $(p, type\text{-}2, T)$ by a Boolean variable for *type*-2, a Boolean variable indicating the presence of $\top$, and for each transition $t \in \mathcal{T}$ a Boolean variable indicating the presence of t in the commitment set T. Considering all valuations of these variables, we see that for each place p there are $2^{|\mathcal{T}|+2}$ possible triples. Since the underlying Petri net is k-bounded, each place can have at most k tokens. So we have $k \cdot |\mathcal{P}|$ copies of places to consider in the decision sets. Thus the size of the set V of states of the Büchi game is bounded by $k \cdot |\mathcal{P}| \cdot 2^{|\mathcal{T}|+2}$, an exponential number of states in the size of the Petri game. Since Büchi games can be solved in polynomial time [18], the total time required to construct and solve the Büchi game is exponential in the size of the Petri game. Büchi games are memoryless determined. The winning strategy can therefore be represented as a finite graph G_f whose size is bounded by the size of the Büchi game. We construct a finite representation of a deadlock avoiding winning strategy for the Petri game following the construction from the proof of Lemma 6.6, using G_f instead of the infinite strategy tree T_f.

The *lower bound* is established by a simple reduction from the combinatorial G_5 game [9]. (Our construction is similar to a reduction given in [19].) In a G_5 game, two players take turns in setting the truth values of boolean variables, one at a time. The boolean variables are partitioned into a set X of variables, which Player I can manipulate, and a set Y of variables that Player II can manipulate. A subset of the variables $Z \subseteq X \cup Y$ is initially set to *true*. Player I wins the game if a given Boolean formula φ over Z never becomes true; otherwise Player II wins.

We reduce a given G_5 game to a Petri game such that the system players have a winning strategy iff Player I has a winning strategy in the G_5 game. We assume w.l.o.g. that φ is in negation normal form, i.e., negations only occur in front of variables. We represent each variable $v \in Z$ by a pair of system places v_0, v_1 indicating the current truth value of v; additionally, there are two environment places t_I, t_{II}, indicating if it is the turn of Player I or Player II. When it is Player I's turn, Player I can move the token for one of the variables in X and must, simultaneously, move the environment token to t_{II}. From there, the environment first chooses whether it wishes to stop the game and evaluate φ, or to update some variable in Y, or to pass; after the update is carried out, the environment player can decide to stop the game and evaluate φ or to return to t_I. If the environment decides to evaluate φ, it traverses the structure of φ, choosing one disjunct in case of a disjunction, or both conjuncts (sequentially) in case of a conjunction, and synchronizing with v_1 for a variable v and with v_0 for the negation of a variable. The initial marking places one token on v_1 for all $v \in Z$ and one token on v_0 for all $v \in (X \cup Y) \setminus Z$. □

Although the reachability problem is decidable also for unbounded Petri nets [20], we cannot decide unbounded Petri games. This is an immediate consequence of the undecidability of VASS (Vector Addition Systems with States) games [21].

Theorem 6.9. *For unbounded Petri games, the question whether the system players have a deadlock-avoiding winning strategy is undecidable. This holds even under the restriction that there is at most one environment token.*

Proof: A 2-dimensional VASS game is a tuple $\mathcal{V} = (S, S_A, S_B, T, S_F)$, where S is a finite set of *control states*, partitioned into S_A and S_B, T is a finite set of *transitions*, and $S_F \subseteq S$ is the set of *final states*. Each transition is a tuple $(s, (a, b), s') \in S \times (\mathbb{Z} \times \mathbb{Z}) \times S$. The game has two players, player A and player B, who are both fully informed. The game is played on a graph of *configurations* $(s, x, y) \in S \times \mathbb{N} \times \mathbb{N}$, where the next transition is chosen by player A if $s \in S_A$ and by player B if $s \in S_B$. Transition $(s, (a, b), s')$ leads from a configuration (s, x, y) to a configuration $(s', x + a, y + b)$, where it is assumed that $x + a \geq 0$ and $y + b \geq 0$, otherwise the transition is blocked. The goal of player A is to avoid the final states S_F, no matter what player B does.

We translate a given VASS game $\mathcal{V} = (S, S_A, S_B, T, S_F)$ into the Petri game $\mathcal{G} = (\mathcal{P}_S, \mathcal{P}_E, \mathcal{T}, \mathcal{F}, In, \mathcal{B})$ defined in the following. In $\mathcal{G}$, the values of x and y are represented as the number of system tokens on two designated places $\hat{x}$ and $\hat{y}$, each control state $s \in S$ is represented by a place $\hat{s}$, and each transition $t = (s, (a, b), s') \in T$ is split into two transitions in $\mathcal{T}$ representing the choice and the subsequent execution of t. The choice of t is represented in $\mathcal{G}$ by a transition called $\widetilde{(s,t)}$ that leads from place $\hat{s}$ to a new intermediate place denoted by $\widehat{(s,t)}$, the execution of t is represented in $\mathcal{G}$ by a transition called $\tilde{t}$ leading from $\widehat{(s,t)}$ to $\widehat{s'}$ and withdrawing or adding the appropriate number of tokens in $\hat{x}$ and $\hat{y}$ according to the values a and b.

Formally, the system places $\mathcal{P}_S = \{\hat{s} \mid s \in S_A\} \cup \{\widehat{(s,t)} \mid s \in S_A, t \in T\} \cup \{\hat{x}, \hat{y}\}$ consist of places for the control states owned by player A and for pairs of such control states and transitions, and, additionally, one place each for x and y. The environment places $\mathcal{P}_E = \{\hat{s} \mid s \in S_B\} \cup \{\widehat{(s,t)} \mid s \in S_B, t \in T\}$ consist of all control states owned by player B and pairs of such control states and transitions. The set $\mathcal{T}$ contains, for each $t = (s, (a, b), s') \in T$, a transition $\widetilde{(s,t)}$ and a transition $\tilde{t}$. Thus $\mathcal{T} = \{\widetilde{(s,t)}, \tilde{t} \mid \exists a, b \in \mathbb{Z}, s' \in S : t = (s, (a, b), s') \in T\}$. For $t = (s, (a, b), s') \in T$, the flow relation $\mathcal{F}$ for these transitions is as follows: $\mathcal{F}(\hat{s}, \widetilde{(s,t)}) = 1$, $\mathcal{F}(\widetilde{(s,t)}, \widehat{(s,t)}) = 1$, $\mathcal{F}(\widehat{(s,t)}, \tilde{t}) = 1$, $\mathcal{F}(\tilde{t}, \widehat{s'}) = 1$, $\mathcal{F}(\hat{x}, \tilde{t}) = (-1) \cdot a$ if $a < 0$ and 0 otherwise, $\mathcal{F}(\hat{y}, \tilde{t}) = (-1) \cdot b$ if $b < 0$ and 0 otherwise, $\mathcal{F}(\tilde{t}, \hat{x}) = a$ if $a > 0$ and 0 otherwise, and $\mathcal{F}(\tilde{t}, \hat{y}) = b$ if $b > 0$ and 0 otherwise. For an illustration see Fig. 13.

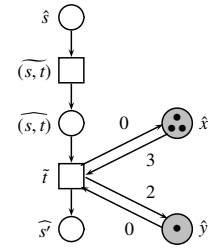

Figure 13. Representation of the transition $t = (s, (-3, 2), s')$ of $\mathcal{V}$ with $s, s' \in S_B$ by two transitions $\widetilde{(s,t)}$ and $\tilde{t}$ in $\mathcal{G}$ involving the environment places $\hat{s}$, $\widehat{(s,t)}$, $\widehat{s'}$ and system places $\hat{x}$, $\hat{y}$.

The idea is that for a place $\hat{s} \in \mathcal{P}_E$ each outgoing choice transition $\widetilde{(s,t)}$ is purely under the control of the environment and thus cannot be blocked by the system players. By contrast, each execution transition $\tilde{t}$ depends on the system places $\hat{x}$ and $\hat{y}$, so the system players can block $\tilde{t}$, but then a deadlock occurs and the system players lose.

We translate a given start configuration $c = (s_0, x_0, y_0)$ of $\mathcal{V}$ into the initial marking In of $\mathcal{G}$, by placing one token on $\widehat{s_0}$, x_0 tokens on $\hat{x}$, and y_0 tokens on $\hat{y}$. In every reachable marking, there is exactly one token on one of the places $\hat{s}$ representing the control states or the places $\widehat{(s,t)}$ representing the decisions.

Since all environment places are of this type, there is *at most one* environment token. However, the number of system tokens in $\hat{x}$ and $\hat{y}$ is unbounded.

We translate the final states S_F of $\mathcal{V}$ into the bad places $\mathcal{B} = \{\hat{s} \mid s \in S_F\}$ of $\mathcal{G}$. Player A has a strategy to avoid the final states in the VASS game $\mathcal{V}$ starting in c iff the system players have a deadlock-avoiding winning strategy in the Petri game $\mathcal{G}$. Since it is undecidable whether player A has a strategy to avoid the final states [21], it is also undecidable whether the system players in a Petri game have a deadlock-avoiding winning strategy. □

7. Application: Traffic Scenario

We consider a traffic scenario with two vehicles, $V1$ and $V2$, represented by system players, and an environment that unpredictably chooses which route is available for the vehicles. If the *safe route* is available, $V1$ passes through an area s_1 and $V2$ through an area s_2, disjoint from s_1. However, if the environment blocks this route, the vehicles have to pass through a *critical section*, where the two vehicles would collide if they attempt to pass it simultaneously. Therefore they have to communicate with each other in order pass the critical section in mutual exclusion. The aim is a strategy for the two vehicles that avoids the bad state of a collision if they have to take this *critical route*.

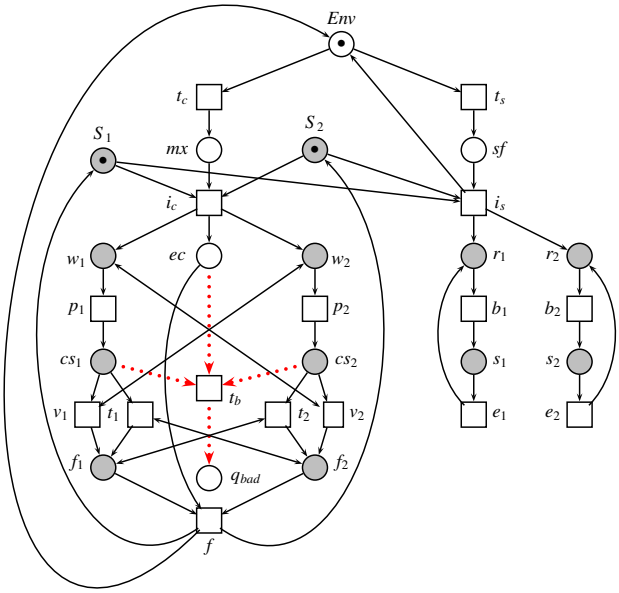

Figure 14. Petri game modeling a traffic scenario with two vehicles represented by two system players, initially on the places S_1 and S_2.

We model this scenario by the Petri game shown in Fig. 14. Initially, the token of the environment player is in the place *Env* and the tokens of the system players (representing the vehicles) are in the places S_1 and S_2. The environment decision for the safe route is represented by the transition t_s, putting the token on place *sf* marking the safe route. The environment decision for the critical route is represented by the transition t_c, putting the token on place *mx* indicating that mutual exclusion needs to be established. Once the environment has chosen which route is available, the system players act accordingly by initiating the safe route via the joint transition i_s, which puts the tokens on the places r_1 and r_2 representing requests for entering the safe route, or the critical route via the joint transition i_c, which puts the tokens on the system places w_1 and w_2 representing waiting for entering the critical route and the environment place e_c standing for environment check.

Let us first look how the game continues in case of the safe route. Here the two system players proceed independently of each other and of the environment. From the place r_i, with $i \in \{1, 2\}$, each system player i begins its safe route s_i via the transition b_i and ends it via the transition e_i. For simplicity, the Petri game models that this can continue forever for each system player so that the places r_1 and r_2 are type-2 places.

Let us now look at the more complex case of the critical route. From the place w_i, with $i \in \{1, 2\}$, each system player i can begin its route through the critical section cs_i via the transition p_i. However, when both players do the same, they may reach their critical sections cs_1 and cs_2 simultaneously, which enables the environment to fire transition t_b and thus put a token onto the bad place q_{bad} that models a collision and represents a loss for the system players. To avoid reaching the bad place, a strategy for the system players will temporarily block one of the transitions p_1 or p_2 and thus let the corresponding token wait on place w_1 or w_2, respectively. The idea is that one system player, say player 1, goes ahead and when it has successfully passed the critical section it should signal the system player 2 that the critical section is now free again. To this end, we connected the transition v_1 exiting the critical section cs_1 with the place w_2, and vice versa, the transition v_2 with the place w_1. Note that firing transition v_1 will not remove the token from w_2. So system player 2 can now fire its transition p_2 to access its critical section cs_2. To terminate this critical section, it has to take the transition t_2 because the transition v_2 (without any token on place w_1) is not enabled any more. Then both system tokens will be on the places f_1 and f_2 indicating that the system players have individually finished accessing their critical sections. Firing the joint transition f finishes one round through the critical route and places the system tokens and the environment token back to their initial positions in the places S_1, S_2 and Env.

Note that, if the system forbids transition t_b to q_{bad}, a deadlock occurs if the critical sections cs_1 and cs_2 are reached simultaneously. By definition, also this represents a loss for the system players.

To discuss a winning strategy for the two system players formally, we look at (an initial part of) the (infinite) unfolding of the Petri game shown in Fig. 15. Notice that the transitions v_2 and v_1 lead to new copies of the places w_1 and w_2, representing the knowledge that now the other system player has left its critical section. When we remove all dashed arrows together with their connected places and transitions, we obtain a winning strategy for the two system players. It is the one informally discussed above. This strategy gives priority to system player 1, but symmetrically, we could have shown a winning strategy that gives priority to player 2. In the strategy shown in Fig. 15, the places labeled with r_1, r_2 and s_1, s_2 are type-2, and all other places are type-1 as their future behavior requires some synchronization with the environment.

8. Related Work

There is a significant body of work on synthesis and control based on Petri nets (cf. [22, 23, 24, 25]). 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 [26]. A general game model for these types of realizability problems are Walukiewicz and Mohalik's *distributed games* [27]. While undecidable in general [26], the distributed synthesis problem can be solved in the Pnueli/Rosner setting for a number of interesting architectures, including pipelines [28], rings [29], and generally all architectures where the processes can be *ordered* according to their informedness [30]. Unfortunately, all these decision procedures have nonelementary complexity. Another important line of work concerns the alternating-time temporal logics, which are interpreted over concurrent game structures [31]. 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 [6, 12, 7, 14] 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 [13, 15, 8]. 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. They also generate the auxiliary places and transitions needed in the strategies.

Related to Petri games is the control problem of Zielonka's asynchronous automata [32]. A (deterministic) *asynchronous automaton* $\mathcal{A}$ is defined for a set P of processes, which are each equipped with a finite set S_p (for $p \in P$) of states. The automaton runs on actions from an alphabet Σ, which is distributed in the sense that a localization function

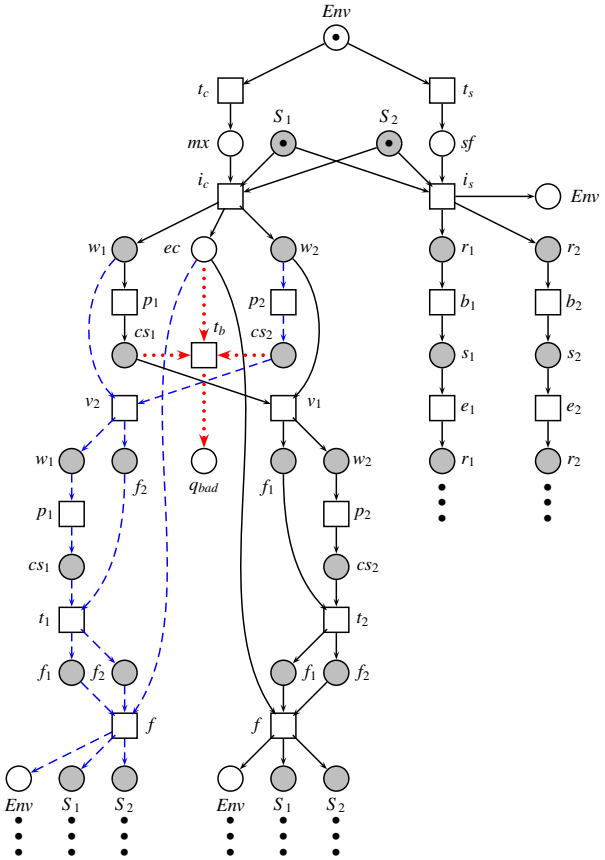

Figure 15. Part of the unfolding of the Petri game in Fig. 14. By removing the dashed parts, a winning strategy for the system players is obtained.

$dom : \Sigma \to (2^P \setminus \emptyset)$ assigns to each action a a set $dom(a)$ of processes that need to synchronize in order to perform the action. Determinism means that every action $a \in \Sigma$ has a partial transition function $\delta_a : \prod_{p \in dom(a)} S_p \nrightarrow \prod_{p \in dom(a)} S_p$, which modifies the state of the processes identified by the localization function. Processes participating in a shared action can exchange complete information about their causal past.

Such an automaton $\mathcal{A}$ corresponds to a labeled, finite Petri net $\mathcal{N}(A)$ that is safe and concurrency-preserving. W.l.o.g. we may assume that $\mathcal{N}(A)$ is reachable. Then each process of $\mathcal{A}$ corresponds to a slice of $\mathcal{N}(A)$ according to Lemma 4.1, i.e., to the behavior of a single token. An action a of $\mathcal{A}$ corresponds to the set of transitions in $\mathcal{N}(A)$ with the label a. Determinism of $\mathcal{A}$ translates into the requirement for $\mathcal{N}(A)$ that any two transitions with the same precondition and the same label are identical. This corresponds to the last condition for occurrence nets: see p. 5.

By this correspondence, it is clear that both in asynchronous automata and in Petri games a causal view is chosen: processes in asynchronous automata and players in Petri games exchange complete information about their causal past when synchronizing on a shared action or a common transition, respectively.

The control problem is now defined with respect to a partitioning of the set of actions into a subset Σ^{sys} of controllable system actions and a subset Σ^{env} of uncontrollable environment actions. In the *action-based* control problem, studied in [3, 33], the controller is another asynchronous automaton over the same alphabet, which can have an arbitrary transition function δ', except that δ' cannot restrict environment actions, i.e., δ_a is a total function for every $a \in \Sigma^{env}$. In the *process-based* control problem, studied in [4], the controller consists of a strategy for each process that decides, based on the local view of the process, which actions should be enabled. An environment action can always be taken, a system action only if it is allowed by the strategies of all involved processes. The action-based

controllers are more powerful than process-based controllers, because for the decision which actions should be enabled, the controller can consider the states of the participating processes together, rather than separately. As a result, the process-based controller can be converted into an action-based controller, but not the other way around [34]. Madhusudan et al. [4] showed the decidability of the process-based control problem under strong assumptions on the synchronization behavior. Genest et al. [5] showed the decidability for this setting for specific classes of architectures such as trees. Gastin et al. [3] showed the decidability of the action-based control problem under restrictions on the dependencies on the actions. Finally, Muscholl and Walukiewicz [33] showed the decidability for this setting for the special case of acyclic architectures.

Unlike asynchronous automata, Petri games are not limited to safe or concurrency-preserving nets. As the examples in Fig. 5 illustrate, Petri games can therefore elegantly model resource allocation problems and situations where the number of processes varies over time, such as fork and join structures. The restriction to a single environment token in the decision procedure of this paper is incomparable to the restrictions of the architecture studied in [5, 33], and it is not obvious how to transfer the result to asynchronous automata.

9. 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 [23], business work flows [35], and other distributed applications.

The decision procedure presented in this paper has been implemented in ADAM [36], a tool for the automatic synthesis of distributed systems with multiple concurrent processes. ADAM uses a BDD-based fixed point iteration to solve the Büchi game from Section 6 symbolically. ADAM has been applied successively in case studies with up to 38 system processes.

In future work, we will investigate further winning conditions, such as reachability, Büchi, and parity, and develop other types of game solving methods, such as SAT-based methods. A first step in this direction is [37], where a technique for finding winning strategies based on the bounded synthesis approach is presented. In bounded synthesis, the size of the strategy is a priori limited. The existence of such a bounded strategy can be encoded as a quantified boolean formula (QBF). This approach is useful to find small strategies, but does not provide a decision procedure.

A major challenge is to extend our decision procedure to a more general class of Petri games. One open question is whether Petri games with more than one environment token are decidable; if so, what is the precise complexity? Another open question is whether there are 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.

Acknowledgements. We thank Manuel Gieseking and Jesko Hecking-Harbusch for their comments on earlier versions of this paper and two anonymous reviewers for their helpful suggestions.

References

[1] B. Finkbeiner, E.-R. Olderog, Petri games: Synthesis of distributed systems with causal memory, in: A. Peron, C. Piazza (Eds.), Proc. Fifth Intern. Symp. on Games, Automata, Logics and Formal Verification (GandALF), Vol. 161 of EPTCS, 2014, pp. 217–230. doi:10.4204/EPTCS.161.19.
URL http://dx.doi.org/10.4204/EPTCS.161.19

[2] W. Zielonka, Asynchronous automata, in: G. Rozenberg, V. Diekert (Eds.), Book of Traces, World Scientific, 1995, pp. 205–248. doi:10.1142/9789814261456_0007.

[3] P. Gastin, B. Lerman, M. Zeitoun, Distributed games with causal memory are decidable for series-parallel systems, in: Proc. FSTTCS, 2004, pp. 275–286. doi:10.1007/978-3-540-30538-5_23.

[4] P. Madhusudan, P. S. Thiagarajan, S. Yang, The MSO theory of connectedly communicating processes, in: Proc. FSTTCS'05, Vol. 3821 of LNCS, Springer, 2005, pp. 201–212. doi:10.1007/11590156_16.

[5] B. Genest, H. Gimbert, A. Muscholl, I. Walukiewicz, Asynchronous games over tree architectures, in: Proc. ICALP'13, Part II, Vol. 7966 of LNCS, Springer, 2013, pp. 275–286. doi:10.1007/978-3-642-39212-2_26.

[6] M. Nielsen, G. D. Plotkin, G. Winskel, Petri nets, event structures and domains, Part I, Theor. Comput. Sci. 13 (1981) 85–108. doi:10.1016/0304-3975(81)90112-2.

[7] J. Engelfriet, Branching processes of Petri nets, Acta Informatica 28 (6) (1991) 575–591. doi:10.1007/BF01463946.
[8] J. Esparza, K. Heljanko, Unfoldings – A Partial-Order Approach to Model Checking, Springer, 2008. doi:10.1007/978-3-540-77426-6.
[9] L. J. Stockmeyer, A. K. Chandra, Provably difficult combinatorial games, SIAM J. Comput. 8 (2) (1979) 151–174. doi:10.1137/0208013.
[10] W. Reisig, Petri Nets – An Introduction, Springer, 1985. doi:10.1007/978-3-642-69968-9.
[11] W. Reisig, Elements of Distributed Algorithms – Modeling and Analysis with Petri Nets, Springer, 1998.
[12] E. Best, C. Fernández, Nonsequential Processes, Springer, 1988. doi:10.1007/978-3-642-73483-0.
[13] J. Esparza, Model checking using net unfoldings, Science of Comp. Programming 23 (1994) 151–195. doi:10.1016/0167-6423(94)00019-0.
[14] J. Meseguer, U. Montanari, V. Sassone, Process versus unfolding semantics for place/transition Petri nets, TCS 153 (1996) 171–210. doi:10.1016/0304-3975(95)00121-2.
[15] V. Khomenko, M. Koutny, W. Vogler, Canonical prefixes of Petri net unfoldings, Acta Informatica 40 (2003) 95–118. doi:10.1007/3-540-45657-0_49.
[16] C. A. R. Hoare, Communicating Sequential Processes, Prentice Hall, 1985. doi:10.1145/359576.359585.
[17] E.-R. Olderog, Nets, Terms and Formulas: Three Views of Concurrent Processes and Their Relationship, Cambridge University Press, 1991. doi:10.1017/CBO9780511526589.
[18] K. Chatterjee, M. Henzinger, An $O(n^2)$ time algorithm for alternating Büchi games, in: Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '12, SIAM, 2012, pp. 1386–1399.
URL http://dl.acm.org/citation.cfm?id=2095116.2095225
[19] G. Katz, D. Peled, S. Schewe, Synthesis of distributed control through knowledge accumulation, in: Proc. CAV, Vol. 6806 of LNCS, Springer, 2011, pp. 510–525.
[20] E. W. Mayr, An algorithm for the general petri net reachability problem, in: Proc. 13th ACM STOC, ACM, 1981, pp. 238–246. doi:10.1145/800076.802477.
[21] P. A. Abdulla, A. Bouajjani, J. d'Orso, Deciding monotonic games, in: Proc. CSL, Vol. 2803 of LNCS, Springer, 2003, pp. 1–14. doi:10.1007/978-3-540-45220-1_1.
[22] A. Giua, Petri nets as discrete event models for supervisory control, Ph.D. thesis, Rensselaer Polytechnic Institute (1992).
[23] Q. Zhou, M. Wang, S. P. Dutta, Generation of optimal control policy for flexible manufacturing cells: A Petri net approach, Intern. Journal of Advanced Manufacturing Technology 10 (1995) 59–65. doi:10.1007/BF01184279.
[24] J.-F. Raskin, M. Samuelides, L. V. Begin, Petri games are monotone but difficult to decide, Technical report, Université Libre De Bruxelles (2003).
[25] U. Buy, H. Darabi, M. Lehene, V. Venepally, Supervisory control of time Petri nets using net unfolding, Annual International Computer Software and Applications Conference 2 (2005) 97–100. doi:10.1109/COMPSAC.2005.148.
[26] A. Pnueli, R. Rosner, Distributed reactive systems are hard to synthesize, in: Proc. FOCS, IEEE Computer Society Press, 1990, pp. 746–757. doi:10.1109/FSCS.1990.89597.
[27] I. Walukiewicz, S. Mohalik, Distributed games, in: Proc. FSTTCS'03, Vol. 2914 of LNCS, 2003, pp. 338–351. doi:10.1007/978-3-540-24597-1_29.
[28] R. Rosner, Modular synthesis of reactive systems, Ph.D. thesis, Weizmann Institute of Science, Rehovot, Israel (1992).
[29] O. Kupferman, M. Y. Vardi, Synthesizing distributed systems, in: Proc. LICS, IEEE Computer Society Press, 2001, pp. 389–398. doi:10.1109/LICS.2001.932514.
[30] B. Finkbeiner, S. Schewe, Uniform distributed synthesis, in: Proc. LICS, IEEE Computer Society Press, 2005, pp. 321–330. doi:10.1109/LICS.2005.53.
[31] R. Alur, T. A. Henzinger, O. Kupferman, Alternating-time temporal logic, Journal of the ACM 49 (5) (2002) 672–713. doi:10.1145/585265.585270.
[32] W. Zielonka, Notes on finite asynchronous automata, RAIRO - Theoretical Informatics and Applications - Informatique Thorique et Applications 21 (2) (1987) 99–135.
URL http://eudml.org/doc/92285
[33] A. Muscholl, I. Walukiewicz, Distributed synthesis for acyclic architectures, in: V. Raman, S. P. Suresh (Eds.), Proc. FSTTCS, Vol. 29 of Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2014, pp. 639–651. doi:http://dx.doi.org/10.4230/LIPIcs.FSTTCS.2014.639.
[34] A. Muscholl, I. Walukiewicz, M. Zeitoun, A look at the control of asynchronous automata, in: Perspectives in Concurrency Theory, CRC Press, ISBN 978-81-7371-652-2, 2009.
[35] W. M. P. van der Aalst, The application of Petri nets to workflow management, J. of Circuits, Systems and Computers 8 (1998) 21–66. doi:10.1142/S0218126698000043.
[36] B. Finkbeiner, M. Gieseking, E.-R. Olderog, Adam: Causality-based synthesis of distributed systems, in: D. Kroening, C. S. Pasareanu (Eds.), Computer Aided Verification (CAV), Part I, Vol. 9206 of LNCS, Springer, 2015, pp. 433–439.
[37] B. Finkbeiner, Bounded synthesis for Petri games, in: R. Meyer, A. Platzer, H. Wehrheim (Eds.), Correct System Design, Vol. 9360 of LNCS, Springer, 2015, pp. 223–237.

Appendix A. Functions and Multisets

For functions f and g let $g \circ f$ denote their composition defined by $(g \circ f)(a) = g(f(a))$. For sets A and B let $f(A)$ denote the image $f(A) = \{f(a) \mid a \in A\}$ and $f^{-1}(B)$ the inverse image $f^{-1}(B) = \{a \mid f(a) \in B\}$. Further, let $f \restriction A$ denote the *restriction* of f to A.

A *multiset* M over a set S is a function $M : S \to \mathbb{N}$. We write $s \in M$ if $M(s) > 0$. We identify $\{0,1\}$-valued multisets with sets and, vice versa, extend some set notation to multisets. We use $\emptyset$ to denote the *empty*

multiset, i.e., with $\emptyset(s) = 0$ for all $s \in S$. For multisets M, N over S let $M \subseteq N$ denote *multiset inclusion*, i.e., $M(s) \leq N(s)$ for all $s \in S$, $M + N$ *multiset addition*, i.e., $(M + N)(s) = M(s) + N(s)$ for all $s \in S$, and $M - N$ multiset difference, i.e., $(M - N)(s) = max(0, M(s) - N(s))$ for all $s \in S$. A multiset M over S is called *finite* if its support $sup(M) = \{s \in S \mid M(s) > 0\}$ is a finite subset of S. For a finite multiset M let $|M|$ denote its cardinality defined by $|M| = \Sigma_{s \in sup(M)}\, M(s)$. Any function $f : S \to S'$ is *freely extended* to multisets M over S: we write $f[M]$ to denote the multiset over S' defined for every $s' \in S'$ as follows:

$$f[M](s') = \Sigma_{s \in f^{-1}(\{s'\})}\, M(s).$$