Translating Asynchronous Games for Distributed Synthesis

Raven Beutner
Saarland University, Saarbrücken, Germany

Bernd Finkbeiner
Saarland University, Saarbrücken, Germany

Jesko Hecking-Harbusch
Saarland University, Saarbrücken, Germany

Abstract

In distributed synthesis, a set of process implementations is generated, which together, accomplish an objective against all possible behaviors of the environment. A lot of recent work has focussed on systems with causal memory, i.e., sets of asynchronous processes that exchange their causal histories upon synchronization. Decidability results for this problem have been stated either in terms of control games, which extend Zielonka's asynchronous automata by partitioning the actions into controllable and uncontrollable, or in terms of Petri games, which extend Petri nets by partitioning the tokens into system and environment players. The precise connection between these two models was so far, however, an open question.

In this paper, we provide the first formal connection between control games and Petri games. We establish the equivalence of the two game types based on weak bisimulations between their strategies. For both directions, we show that a game of one type can be translated into an equivalent game of the other type. We provide exponential upper and lower bounds for the translations. Our translations allow to transfer and combine decidability results between the two types of games. Exemplarily, we translate decidability in acyclic communication architectures, originally obtained for control games, to Petri games, and decidability in single-process systems, originally obtained for Petri games, to control games.

2012 ACM Subject Classification Theory of computation → Algorithmic game theory

Keywords and phrases synthesis, distributed systems, causal memory, Petri games, control games

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2019.26

Related Version A full version of the paper is available at `https://arxiv.org/abs/1907.00829` [3].

Funding Supported by the German Research Foundation (DFG) Grant Petri Games (392735815) and the Collaborative Research Center "Foundations of Perspicuous Software Systems" (TRR 248, 389792660), and by the European Research Council (ERC) Grant OSARES (683300).

1 Introduction

Synthesis is the task of automatically generating an implementation fulfilling a given objective or proving that no such implementation can exist. Synthesis can be viewed as a game between the *system* and the *environment* with winning strategies for the system being correct implementations [4]. We call a class of games *decidable* if we can determine the existence of a winning strategy. A distributed system consists of local processes, that possess incomplete information about the global system state. *Distributed synthesis* searches for distributed strategies that govern the local processes such that the system as a whole satisfies an objective, independently of the inputs that are received from the environment.

© Raven Beutner, Bernd Finkbeiner, and Jesko Hecking-Harbusch;
licensed under Creative Commons License CC-BY

After some early results on *synchronous* distributed systems [24], most work has focussed on the synthesis of *asynchronous* distributed systems with *causal memory* [12, 13, 20, 14, 11, 10]. Causal memory means that two processes share no information while they run independently; during every synchronization, however, they exchange their complete local histories. The study of the synthesis problem with causal memory has, so far, been carried out, independently of each other, in two different models: control games and Petri games.

Control Games and Petri Games. *Control games* [13] are based on Zielonka's asynchronous automata [26], which are compositions of local processes. The actions of the asynchronous automaton are partitioned as either controllable or uncontrollable. Hence, each process can have both controllable and uncontrollable behavior. A strategy comprises a family of one individual controller for each process that can restrict controllable actions based on the causal past of the process but has to account for all uncontrollable actions. Together, the local controllers aim to fulfill an objective against all possible unrestricted behavior. There are non-elementary decidability results for acyclic communication architectures [13, 20]. Decidability has also been obtained for restrictions on the dependencies of actions [12] or on the synchronization behavior [16, 17] and, recently, for decomposable games [14].

Petri games [11] are based on Petri nets. They partition the places of the underlying Petri net into system places and environment places and, thereby, group the tokens into system players and environment players. For tokens in system places, the outgoing transitions can be restricted by the strategy whereas tokens in environment places cannot be controlled, i.e., every possible transition has to be accounted for. Strategies are defined as restrictions of the global unfolding and aim to fulfill an objective against all possible unrestricted behavior. Petri games are EXPTIME-complete for a bounded number of system players and one environment player [11] as well as for one system player and a bounded number of environment players [10]. Both models are based on causal information: Control games utilize local views whereas Petri games utilize unfoldings.

Translations. The precise connection between control games and Petri games, and hence, the question whether results can be transferred between them, was, so far, open. We translate control games into Petri games, and vice versa. Both game types admit strategies based on causal information but the formalisms for the possibilities of system and environment differ. In control games, an action is either controllable or uncontrollable and therefore can be restricted by either all or none of the involved players. From the same state of a process, both controllable and uncontrollable behavior is possible. By contrast, Petri games utilize a partitioning into system and environment places. While this offers more precise information about which player can control a shared transition, a given place can no longer comprise both system and environment behavior. The challenge is to resolve the controllability while preserving the causal information in the game. For both translations, we adopt the concept of *commitment sets*: The local players do not enable behavior directly but move to a state or place that explicitly encodes their decision of what to enable. Using this explicit representation, we can express the controllability aspects of one game in the respective other one, i.e., make actions in a control game controllable by only a subset of players and allow places in Petri games that comprise both environment and system behavior.

Our translations preserve the structure of winning strategies in a weak bisimilar way. In addition to the upper bounds established by our exponential translations, we provide matching lower bounds. The translations show that contrasting formalisms can be overcome whereas our lower bounds highlight an intrinsic difficultly to achieve this. The equivalence

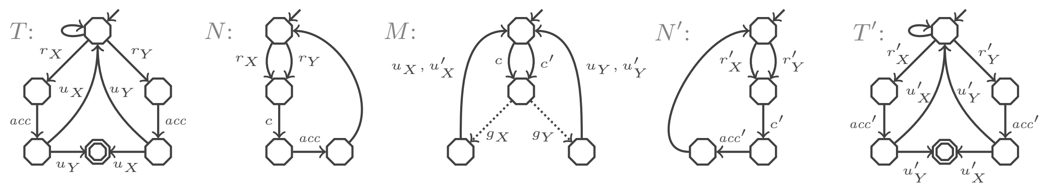
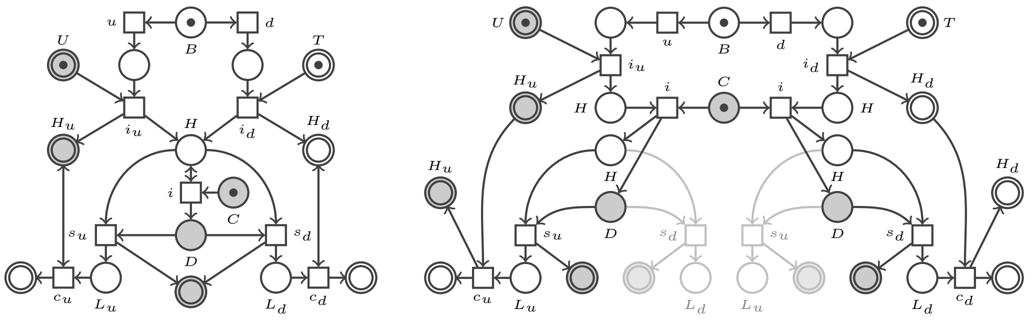


Figure 1 A control game for a manager M of resources X and Y between threads T and T' with networks N and N' is depicted. Communication occurs by synchronization on shared actions. Dotted actions are controllable, all others are uncontrollable. Losing states are double circles.



(a) Petri game for a police strategy. **(b)** Unfolding (with grayed parts) and winning strategy (without).

Figure 2 A Petri game, an unfolding, and a winning strategy are given. Gray places belong to the system whereas white places belong to the environment. Winning places are double circles.

of both models, as witnessed by our results, gives rise to more practical applications by allowing the transfer of existing decidability results between both models. As an example, we can transfer decidability of single-process systems for Petri games [10] to control games and decidability for acyclic communication architectures for control games [13] to Petri games.

2 Examples

We illustrate the models with two examples. The examples demonstrate the use of control games and Petri games and their differences, which our translations overcome. Both examples highlight decidable classes [10, 13], that are transferable through the results of the paper.

As a control game, consider the example of a manager for resources in Fig. 1. The control game consists of five players: A manager M and two pairs of thread and network connection (T, N and T', N'). Both pairs of thread and network connection are identical but act on disjoint actions (primed and not). There are two resources X and Y that are managed by M. Each thread (T, T') can request access to one of them (r_X, r_Y) and afterwards wait for the acknowledgement from its network connection (acc). After the acknowledgement, the thread can use one of the resources (u_X, u_Y). Each network connection (N, N') synchronizes with its thread on the actions for requests and synchronizes with the manager for communication (c). Afterwards, each network connection sends the acknowledgement to its thread. The manager is the only process that comprises controllable actions. Upon communication with one of the two network connections, the manager can grant access to the resources X or Y using the controllable actions g_X or g_Y. The enabled resource can afterwards be accessed and used (u_X, u_Y). A losing state can be reached for either thread if an unwanted resource is enabled, i.e., after the acknowledgement, the requested and granted resource do not match.

This control game can be won by the system. After every communication with a network connection, the manager enables the resource that the respective thread requested. A winning controller relies on the information transfer associated with every synchronization. The request of the process is transferred to the manager upon communication with the network connection. Then, the correct resource can be enabled. This control game falls into a decidable class by our translation to Petri games as it is a single-process system with bad places [10]. Note that the control game has a cyclic communication architecture.

As a Petri game, consider the example of a burglary in Fig. 2a. A crime boss in environment place B decides to either burgle up- or downtown by firing transition u or d. Depending on the choice, an undercover agent in system place U or a thug in environment place T is instructed by transition i_u or i_d and commits the burglary, i.e., moves to place H_u or H_d. This returns the crime boss to her hideout H where she gets caught and interrogated (i) by a cop in system place C. Afterwards, the cop can send (s_u, s_d) the flipped crime boss up- or downtown to place L_u or L_d in order to intercept the burglary (c_u, c_d).

Causal past is key for the existence of winning strategies. Only upon synchronization players exchange all information about their past. After the crime boss instructs for a location to burgle, only she and the respective burglar know about the decision. The cop learns about the location of the burglary after catching the crime boss. A winning strategy for the cop catches and interrogates the crime boss and then uses the obtained information to send the flipped crime boss to the correct location. For this Petri game, our translation results in a control game with acyclic communication architecture [13]. Note that the Petri game has two system and two environment players.

3 Background

We recall asynchronous automata [26], control games [13], Petri nets [25], and Petri games [11].

3.1 Zielonka's Asynchronous Automata

An *asynchronous automaton* [26] is a family of finite automata, called *processes*, synchronizing on shared actions. Our definitions follow [13]. The finite set of processes of an asynchronous automaton is defined as $\mathcal{P}$. A *distributed alphabet* (Σ, dom) consists of a finite set of *actions* Σ and a *domain* function $\mathit{dom} : \Sigma \to 2^{\mathcal{P}} \setminus \{\emptyset\}$. For an action $a \in \Sigma$, $\mathit{dom}(a)$ are all processes that have to synchronize on a. For a process $p \in \mathcal{P}$, $\Sigma_p = \{a \in \Sigma \mid p \in \mathit{dom}(a)\}$ denotes all actions p is involved in. A (deterministic) asynchronous automaton $\mathcal{A} = (\{S_p\}_{p\in\mathcal{P}}, s_{in}, \{\delta_a\}_{a\in\Sigma})$ is defined by a finite set of local states S_p for every process $p \in \mathcal{P}$, the initial state $s_{in} \in \prod_{p\in\mathcal{P}} S_p$, and a partial function $\delta_a : \prod_{p\in \mathit{dom}(a)} S_p \dashrightarrow \prod_{p\in \mathit{dom}(a)} S_p$. We call an element $\{s_p\}_{p\in\mathcal{P}} \in \prod_{p\in\mathcal{P}} S_p$ *a global state.* For a set of processes $R \subseteq \mathcal{P}$, we abbreviate $s_R = \{s_p\}_{p\in R}$ as the restriction of the global state to R. We denote that a local state $s' \in S_p$ is part of a global state s_R by $s' \in s_R$. For a local state s', we define the set of outgoing actions by $\mathit{act}(s') = \{a \in \Sigma \mid \exists s_{\mathit{dom}(a)} \in \mathit{domain}(\delta_a) : s' \in s_{\mathit{dom}(a)}\}$. We can view an asynchronous automaton as a sequential automaton with state space $\prod_{p\in\mathcal{P}} S_p$ and transitions $s \xrightarrow{a} s'$ if $(s_{\mathit{dom}(a)}, s'_{\mathit{dom}(a)}) \in \delta_a$ and $s_{\mathcal{P}\setminus \mathit{dom}(a)} = s'_{\mathcal{P}\setminus \mathit{dom}(a)}$. By $\mathit{Plays}(\mathcal{A}) \in \Sigma^* \cup \Sigma^\omega$, we denote the set of finite and infinite sequences in this global automaton. For a finite $u \in \mathit{Plays}(\mathcal{A})$, $\mathit{state}(u)$ denotes the global state after playing u and $\mathit{state}_p(u)$ the local state of process p.

The domain function dom induces an independence relation I: Two actions $a, b \in \Sigma$ are independent, denoted by $(a, b) \in I$, if they involve different processes, i.e., $\mathit{dom}(a) \cap \mathit{dom}(b) = \emptyset$. Adjoint independent actions of sequences of actions can be swapped. This leads to an equivalence relation $\sim_I$ between sequences, where $u \sim_I w$ if u and w are identical up to multiple swaps of consecutive independent actions. The equivalence classes of $\sim_I$ are called *traces* and denoted by $[u]_I$ for a sequence u. Given the definition of asynchronous automata, it is natural to abstract from concrete sequences and consider $\mathit{Plays}(\mathcal{A})$ as a set of traces.

In our translation, an alternative characterization of a subset of asynchronous automata turns out to be practical: We describe every process p by a finite local automaton $\mathfrak{N}_p = (Q_p, s_{0,p}, \vartheta_p)$ acting on actions from Σ_p. Here, Q_p is a finite set of states, $s_{0,p}$ the initial state and $\vartheta_p \subseteq Q_p \times \Sigma_p \times Q_p$ a deterministic transition relation. For a family of local processes $\{\mathfrak{N}_p\}_{p\in\mathfrak{P}}$, we define the *parallel composition* $\bigotimes_{p\in\mathfrak{P}} \mathfrak{N}_p$ as an asynchronous automaton with **(1)** $\forall p \in \mathfrak{P} : S_p = Q_p$, **(2)** $s_{in} = \{s_{0,p}\}_{p\in\mathfrak{P}}$, and **(3)** $\delta_a(\{s_p\}_{p\in dom(a)})$: If for all $p \in dom(a)$, there exists a state $s'_p \in S_p$ with $(s_p, a, s'_p) \in \vartheta_p$ then define $\delta_a(\{s_p\}_{p\in dom(a)}) = \{s'_p\}_{p\in dom(a)}$, otherwise it is undefined. Figure 1 is an example of such a parallel composition. Note that not every asynchronous automaton can be described as a composition of local automata.

3.2 Control Games

A *control game* [13] $\mathcal{C} = (\mathcal{A}, \Sigma^{sys}, \Sigma^{env}, \{\mathcal{S}_p\}_{p\in\mathfrak{P}})$ consists of an asynchronous automaton $\mathcal{A}$ as a game arena, a distribution of actions into *controllable* actions Σ^{sys} and *uncontrollable* actions Σ^{env}, and *special states* $\{\mathcal{S}_p\}_{p\in\mathfrak{P}}$ for a winning objective. We define the set of plays in the game as $\mathit{Plays}(\mathcal{C}) = \mathit{Plays}(\mathcal{A})$. Intuitively, a strategy for $\mathcal{C}$ can restrict controllable actions but cannot prohibit uncontrollable actions. Given a play u, a process p only observes parts of it. The local p-view, denoted by $\mathit{view}_p(u)$, is the shortest trace $[v]_I$ such that $u \sim_I v\,w$ for some w not containing any actions from Σ_p. The p-view describes the *causal past* of process p and contains all actions the process is involved in and all actions it learns about via communication. We define the set of p-views as $\mathit{Plays}_p(\mathcal{C}) = \{\mathit{view}_p(u) \mid u \in \mathit{Plays}(\mathcal{C})\}$.

To avoid confusion with Petri games, we refer to strategies for control games as controllers. A *controller* for $\mathcal{C}$ is a family of local controllers for all processes $\varrho = \{f_p\}_{p\in\mathfrak{P}}$. A local controller for a process p is a function $f_p : \mathit{Plays}_p(\mathcal{C}) \to \Sigma^{sys} \cap \Sigma_p$. $\mathit{Plays}(\mathcal{C}, \varrho)$ denotes the set of plays respecting ϱ. It is defined as the smallest set containing the empty play ϵ and such that for every $u \in \mathit{Plays}(\mathcal{C}, \varrho)$: **(1)** if $a \in \Sigma^{env}$ and $ua \in \mathit{Plays}(\mathcal{C})$ then $ua \in \mathit{Plays}(\mathcal{C}, \varrho)$ and **(2)** if $a \in \Sigma^{sys}$, $ua \in \mathit{Plays}(\mathcal{C})$, and $\forall p \in dom(a) : a \in f_p(\mathit{view}_p(u))$ then $ua \in \mathit{Plays}(\mathcal{C}, \varrho)$. Environment actions are always possible whereas system actions are only possible if allowed by the local controllers of *all* participating processes. Local controllers base their decisions on their local view and thereby act only on their causal past.

We define the (possibly empty) set of *final plays* $\mathit{Plays}^F(\mathcal{C}, \varrho)$ as all finite plays $u \in \mathit{Plays}(\mathcal{C}, \varrho)$ such that there is no a with $u\,a \in \mathit{Plays}(\mathcal{C}, \varrho)$. We consider either reachability or safety objectives for the system. Therefore, $\{\mathcal{S}_p\}_{p\in\mathfrak{P}}$ describes sets of winning $(\mathcal{W}_p)$ or losing $(\mathcal{B}_p)$ states. A controller ϱ is *reachability-winning* if it only admits finite plays and on each final play all processes terminate in a winning state. For safety objectives, we need to ensure progress. A controller ϱ is *deadlock-avoiding* if $\mathit{Plays}^F(\mathcal{C}, \varrho) \subseteq \mathit{Plays}^F(\mathcal{C}, \top)$ for the controller $\top$ allowing all actions, i.e., the controller only terminates if the asynchronous automaton does. A controller ϱ is *safety-winning* if it is deadlock-avoiding and no play in $\mathit{Plays}(\mathcal{C}, \varrho)$ visits any local, losing state from $\bigcup_{p\in\mathfrak{P}} \mathcal{B}_p$.

3.3 Petri Nets

A *Petri net* [25, 22] $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, \mathit{In})$ consists of disjoint sets of *places* $\mathcal{P}$ and *transitions* $\mathcal{T}$, the *flow relation* $\mathcal{F}$ as multiset over $(\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$, and the *initial marking* In as multiset over $\mathcal{P}$. We call elements in $\mathcal{P} \cup \mathcal{T}$ *nodes* and $\mathcal{N}$ *finite* if the set of nodes is finite. For node x, the *precondition* (written $\mathit{pre}(x)$) is the multiset defined by $\mathit{pre}(x)(y) = \mathcal{F}(y, x)$ and *postcondition* (written $post(x)$) the multiset defined by $post(x)(y) = \mathcal{F}(x, y)$. For multiple nets $\mathcal{N}^\sigma, \mathcal{N}^1, \cdots$, we refer to the components by $\mathcal{P}^{\mathcal{N}^\sigma}$ and write $pre^{\mathcal{N}^\sigma}(x)$ unless clear from the context. Configurations of Petri nets are represented by multisets over places, called *markings*. *In* is the initial marking. For a transition t, $pre(t)$ is the multiset of places from which tokens are consumed. A transition t is *enabled* in marking M if $pre(t) \subseteq M$, i.e., every place in M contains at least as many tokens as required by t. If no transition is enabled from marking M then we call M *final*. An enabled transition t can *fire* from a marking M resulting in the successor marking $M' = M - pre(t) + post(t)$ (denoted $M\ [\,t\,\rangle\ M'$). For markings M and M', we write $M\ [\,{t_0,\ldots,t_{n-1}}\,\rangle\ M'$ if there exist markings $M = M_0, \ldots, M_n = M'$ s.t. $M_i\ [\,t_i\,\rangle\ M_{i+1}$ for all $0 \leq i \leq n-1$. The set of *reachable markings* of $\mathcal{N}$ is defined as $\mathcal{R}(\mathcal{N}) = \{M \mid \exists n \in \mathbb{N}, t_0, \ldots, t_{n-1} \in \mathcal{T} : In\ [\,{t_0,\ldots,t_{n-1}}\,\rangle\ M\}$. A net $\mathcal{N}'$ is a *subnet* of $\mathcal{N}$ (written $\mathcal{N}' \sqsubseteq \mathcal{N}$) if $\mathcal{P}' \subseteq \mathcal{P}$, $\mathcal{T}' \subseteq \mathcal{T}$, $In' \subseteq In$, and $\mathcal{F}' = \mathcal{F} \restriction (\mathcal{P}' \times \mathcal{T}') \cup (\mathcal{T}' \times \mathcal{P}')$. A Petri net is *1-bounded* if every reachable marking contains at most one token per place. It is *concurrency-preserving* if $|pre(t)| = |post(t)|$ for all transitions t.

For nodes x and y, we write $x \lessdot y$ if $x \in pre(y)$, i.e., there is an arc from x to y. With $\leq$, we denote the reflexive, transitive closure of $\lessdot$. The *causal past* of x is $past(x) = \{y \mid y \leq x\}$. x and y are *causally related* if $x \leq y \vee y \leq x$. They are *in conflict* (written $x \,\sharp\, y$) if there exists a place $q \in \mathcal{P} \setminus \{x, y\}$ and two distinct transitions $t_1, t_2 \in post(q)$ s.t. $t_1 \leq x$ and $t_2 \leq y$. Node x is in *self-conflict* if $x \,\sharp\, x$. We call x and y *concurrent* if they are neither causally related nor in conflict. An *occurrence net* is a Petri net $\mathcal{N}$, where the pre- and postcondition of all transitions are sets, the initial marking coincides with places without ingoing transitions ($\forall q \in \mathcal{P} : q \in In \Leftrightarrow |pre(q)| = 0$), all other places have exactly one ingoing transition ($\forall q \in \mathcal{P} \setminus In : |pre(q)| = 1$), $\leq$ is *well-founded* (no infinite path following the inverse flow relation exists), and no transition is in self-conflict. An *initial homomorphism* from $\mathcal{N}$ to $\mathcal{N}'$ is a function $\lambda : \mathcal{P} \cup \mathcal{T} \to \mathcal{P}' \cup \mathcal{T}'$ that respects node types ($\lambda(\mathcal{P}) \subseteq \mathcal{P}' \wedge \lambda(\mathcal{T}) \subseteq \mathcal{T}'$), is structure-preserving on transitions ($\forall t \in \mathcal{T} : \lambda[pre^{\mathcal{N}}(t)] = pre^{\mathcal{N}'}(\lambda(t)) \wedge \lambda[post^{\mathcal{N}}(t)] = post^{\mathcal{N}'}(\lambda(t))$), and agrees on the initial markings ($\lambda[In] = In'$).

A branching process [5, 18, 6] describes parts of the behavior of a Petri net. Formally, an *(initial) branching process* of a Petri net $\mathcal{N}$ is a pair $\iota = (\mathcal{N}^\iota, \lambda^\iota)$ where $\mathcal{N}^\iota$ is an occurrence net and $\lambda^\iota : \mathcal{P}^\iota \cup \mathcal{T}^\iota \to \mathcal{P} \cup \mathcal{T}$ is an initial homomorphism from $\mathcal{N}^\iota$ to $\mathcal{N}$ that is injective on transitions with the same precondition ($\forall t, t' \in \mathcal{T}^\iota : (pre^{\mathcal{N}^\iota}(t) = pre^{\mathcal{N}^\iota}(t') \wedge \lambda^\iota(t) = \lambda^\iota(t')) \Rightarrow t = t'$). A branching process describes subsets of possible behaviors of a Petri net. Whenever a place or transition can be reached on two distinct paths it is split up. λ can be thought of as label of the copies into nodes of $\mathcal{N}$. The injectivity condition avoids additional unnecessary splits: Each transition must either be labelled differently or occur from different preconditions. The *unfolding* $\mathfrak{U}$ of $\mathcal{N}$ is the maximal branching process: Whenever there is a set of pairwise concurrent places C s.t. $\lambda[C] = pre^{\mathcal{N}}(t)$ for some transition t then there exists t' with $\lambda(t') = t$ and $pre^{\mathcal{N}^{\mathfrak{U}}}(t') = C$. It represents ever possible behavior of $\mathcal{N}$.

3.4 Petri Games

A *Petri game* [11] is a tuple $\mathcal{G} = (\mathcal{P}_\mathcal{S}, \mathcal{P}_\mathcal{E}, \mathcal{T}, \mathcal{F}, In, Sp)$. The *system places* $\mathcal{P}_\mathcal{S}$ and *environment places* $\mathcal{P}_\mathcal{E}$ partition the places of the underlying, finite net $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, In)$ with $\mathcal{P} = \mathcal{P}_\mathcal{S} \uplus \mathcal{P}_\mathcal{E}$. We extend notation from the underlying net to $\mathcal{G}$ by, e.g., defining $pre^{\mathcal{G}}(\cdot) = pre^{\mathcal{N}}(\cdot)$ and $\mathcal{P}^{\mathcal{G}} = \mathcal{P}^{\mathcal{N}}$. The game progresses by firing transitions in the underlying net. Intuitively, a strategy can control the behavior of tokens on system places by deciding which transitions to allow. Tokens on environment places belong to the environment and cannot be restricted by strategies. $Sp \subseteq \mathcal{P}$ denotes *special places* used to pose a winning objective. For graphical representation, we depict a Petri game as the underlying net and color system places gray, environment places white, and special places as double circles (cf. Fig. 2).

A *strategy* for $\mathcal{G}$ is an initial branching process $\sigma = (\mathcal{N}^\sigma, \lambda^\sigma)$ satisfying *justified refusal*: If there is a set of pairwise concurrent places C in $\mathcal{N}^\sigma$ and a transition $t \in \mathcal{T}^\mathcal{G}$ with $\lambda[C] = pre^\mathcal{G}(t)$ then there either is a transition t' with $\lambda(t') = t$ and $C = pre^{\mathcal{N}^\sigma}(t')$ or there is a system place $q \in C \cap \lambda^{-1}[\mathcal{P_S}]$ with $t \notin \lambda[post^{\mathcal{N}^\sigma}(q)]$. Since a branching process describes subsets of the behavior of a Petri net, a strategy is a restriction of possible moves in the game. Justified refusal enforces that only system places can prohibit transitions based on their causal past. From every situation in the game, a transition possible in the underlying net is either allowed, i.e., in the strategy, or there is a *system* place that never allows it. In particular, transitions involving only environment places are always possible. A strategy σ is *reachability-winning* for a set of winning places $Sp = \mathcal{W}$ if $\mathcal{N}^\sigma$ is a finite net and in each final, reachable marking every token is on a winning place. A strategy is *deadlock-avoiding* if for every final, reachable marking M in the strategy, $\lambda[M]$ is final as well, i.e., the strategy is only allowed to terminate if the underlying Petri net does so. A strategy σ is *safety-winning* for bad places $Sp = \mathcal{B}$ if it is deadlock-avoiding and no reachable marking contains a bad place. For both objectives, we can require σ to be *deterministic*: For every reachable marking M and system place $q \in M$ there is at most one transition from $post^{\mathcal{N}^\sigma}(q)$ enabled in M. In Fig. 2b, the unfolding of Fig. 2a is depicted labeled by λ. Excluding the grayed parts, this is a winning strategy for the system.

For safety as winning objective, unbounded Petri games are undecidable in general [11] whereas bounded ones with either one system player [10] or one environment player [11] are EXPTIME-complete. Bounded synthesis is a semi-decision procedure to find winning strategies [7, 8, 15]. Both approaches are implemented in the tool ADAM [9, 8].

4 Game Equivalence

A minimum requirement for translations between games is to be *winning-equivalent*. The system has a winning strategy in one game if and only if it has a winning strategy in the translated other one. One trivial translation fulfilling this is to solve the game and to return a minimal winning-equivalent game. Such a translation is not desirable, especially since decidability in both control games and Petri games is still an open question [19, 11]. Instead, our translations preserve the underlying structure of the games. We propose *strategy-equivalence* as an adequate equivalence notion. Our notion is based on weak bisimulation which is popular and powerful to relate concurrent systems represented as Petri nets [2, 1, 23].

For our purpose, a bisimulation between the underlying Petri net and the asynchronous automaton is not sufficient. Instead, we want to express that any strategy can be matched by a strategy that allows equivalent (bisimilar) behavior, i.e., allows identical actions/transitions. In both models, strategies are defined based on the causal past of the players. A Petri game $\mathcal{G}$ utilizes unfoldings whereas a control game $\mathcal{C}$ utilizes local views. We consider a strategy and a controller equivalent if there is a weak bisimulation between the branching process of the strategy and the plays that are compatible with the controller. We base our definition on a set of shared actions and transitions between the Petri game and the control game. We refer to them as *observable*. All non-shared transitions and actions are considered *internal* (τ). If we, e.g., translate a Petri game to a control game we aim for a control game that contains all transitions as observable actions but might add internal ones.

▶ **Definition 1.** *A strategy σ for $\mathcal{G}$ and controller ϱ for $\mathcal{C}$ are* bisimilar *if there exists a relation $\approx_\mathfrak{B} \subseteq \mathcal{R}(\mathcal{N}^\sigma) \times Plays(\mathcal{A}, \varrho)$ s.t. $In^\sigma \approx_\mathfrak{B} \epsilon$ and all following conditions hold:*

- *If $M \approx_\mathfrak{B} u$ and $M\,[\,a\,\rangle\,M'$ there exists $u' \in Plays(\mathcal{A}, \varrho)$ with $u' = u\tau^*a\tau^*$ and $M' \approx_\mathfrak{B} u'$*
- *If $M \approx_\mathfrak{B} u$ and $M\,[\,\tau\,\rangle\,M'$ there exists $u' \in Plays(\mathcal{A}, \varrho)$ with $u' = u\tau^*$ and $M' \approx_\mathfrak{B} u'$*
- *If $M \approx_\mathfrak{B} u$ and $u' = u\,a$ there exists $M' \in \mathcal{R}(\mathcal{N}^\sigma)$ with $M\,[\,\tau^*a\tau^*\,\rangle\,M'$ and $M' \approx_\mathfrak{B} u'$*
- *If $M \approx_\mathfrak{B} u$ and $u' = u\,\tau$ there exists $M' \in \mathcal{R}(\mathcal{N}^\sigma)$ with $M\,[\,\tau^*\,\rangle\,M'$ and $M' \approx_\mathfrak{B} u'$*

A Petri game $\mathcal{G}$ and a control game $\mathcal{C}$ are called *strategy-equivalent* if for every winning strategy σ for $\mathcal{G}$ there exists a bisimilar winning controller ϱ_σ for $\mathcal{C}$ and for every winning controller ϱ for $\mathcal{C}$ there exists a bisimilar winning strategy σ_ϱ for $\mathcal{G}$.

5 Translating Petri Games to Control Games

We give our translation from Petri games to control games and prove that it yields strategy-equivalent (and therefore winning-equivalent) games. Moreover, we provide an exponential lower bound, showing that our translation is asymptotically optimal when requiring strategy-equivalence. We present the translation for reachability objectives. Due to space restrictions, all proofs and further details can be found in the full version of the paper [3].

5.1 Construction

We describe the construction of our translation for a restrictive class of Petri games called *sliceable*. In Sec. 5.3, the construction is generalized to *concurrency-preserving* Petri games.

Slices. A Petri game describes the *global* behavior of the players. By contrast, a control game is defined in terms of *local* processes. Similarly, a Petri game strategy is a global branching process opposed to a family of local controllers for control games. The first difference our translation needs to overcome is to distribute a Petri game into parts describing the local behavior of players. Therefore, we dismantle the Petri game into *slices* for each token.

▶ **Definition 2.** *A slice of a Petri net $\mathcal{N}$ is a Petri net $\varsigma = (\mathcal{P}^\varsigma, \mathcal{T}^\varsigma, \mathcal{F}^\varsigma, In^\varsigma)$ s.t.,* **(1)** $\varsigma \sqsubseteq \mathcal{N}$*,* **(2)** $|In^\varsigma| = 1$*,* **(3)** $\forall t \in \mathcal{T}^\varsigma : |pre^\varsigma(t)| = |post^\varsigma(t)| = 1$*,* **(4)** $\forall q \in \mathcal{P}^\varsigma : post^{\mathcal{N}}(q) \subseteq \mathcal{T}^\varsigma$

A slice is a subnet of $\mathcal{N}$ **(1)** that describes the course of exactly one token **(2, 3)** and includes every possible move of this token **(4)**. A slice characterizes the exact behavior of a single token in the global net $\mathcal{N}$. For a family of slices $\{\varsigma\}_{\varsigma \in \mathfrak{S}}$, the parallel composition $\|_{\varsigma \in \mathfrak{S}}\ \varsigma$ is the Petri net with places $\biguplus_{\varsigma \in \mathfrak{S}} \mathcal{P}^\varsigma$, transitions $\bigcup_{\varsigma \in \mathfrak{S}} \mathcal{T}^\varsigma$, flow relation $\biguplus_{\varsigma \in \mathfrak{S}} \mathcal{F}^\varsigma$, and initial marking $\biguplus_{\varsigma \in \mathfrak{S}} In^\varsigma$. All unions, except for the union of transitions, are disjoint. Transitions can be shared between multiple slices, creating synchronization. A Petri net $\mathcal{N}$ is *sliceable* if there is a family of slices $\{\varsigma\}_{\varsigma \in \mathfrak{S}}$ s.t. $\mathcal{N} = \|_{\varsigma \in \mathfrak{S}}\ \varsigma$ and $\biguplus_{\varsigma \in \mathfrak{S}} \mathcal{P}^\varsigma$ is a partition of $\mathcal{P}^{\mathcal{N}}$, i.e., $\mathcal{N}$ can be described by the local movements of tokens. Sliceable Petri nets are concurrency-preserving and 1-bounded. We extend slices to Petri games in the natural way by distinguishing system, environment, and special places. Figure 4 depicts a Petri game (a) and a possible distribution into slices (b). Note that even concurrency-preserving and 1-bounded Petri games must not be sliceable and that a distribution in slices is not unique.

Commitment Sets. In control games, actions are either controllable or uncontrollable whereas, in Petri games, players are distributed between the system and the environment. In our construction, we represent transitions as actions and need to guarantee that only certain players can control them. In control games, this cannot be expressed directly. We overcome this difference by using commitment sets. Each process that should be able to control an action chooses a commitment set, i.e., moves to a state that explicitly encodes its decision.

We fix a sliceable game $\mathcal{G} = (\mathcal{P}_\mathcal{S}, \mathcal{P}_\mathcal{E}, \mathcal{T}, \mathcal{F}, In, \mathcal{W})$ and a distribution in slices $\{\varsigma\}_{\varsigma \in \mathfrak{S}}$. We begin by defining a control game $\mathcal{C}_\mathcal{G}$. Afterwards, we describe a possible modification $\widehat{\mathcal{C}_\mathcal{G}}$, that enforces determinism. The construction is depicted in Fig. 3.

Define $\mathfrak{P} = \mathfrak{S}$ and the distributed alphabet as (Σ, dom) with:

$$\Sigma = \mathcal{T} \,\cup\, \{\tau_{(q,A)} \mid q \in \mathcal{P}_{\mathcal{S}} \;\wedge\; A \subseteq \mathit{post}^{\mathcal{G}}(q)\}$$
$$\cup\, \{\, \lightning^{(q,A)}_{[t_1,t_2]} \mid q \in \mathcal{P}_{\mathcal{S}} \;\wedge\; A \subseteq \mathit{post}^{\mathcal{G}}(q) \;\wedge\; t_1, t_2 \in A \;\wedge\; t_1 \neq t_2\}$$

and $\mathit{dom} : \Sigma \to 2^{\mathfrak{P}} \setminus \{\emptyset\}$:

$$\mathit{dom}(\,t\,) = \{\varsigma \in \mathfrak{S} \mid t \in \mathcal{T}^{\varsigma}\} \quad \text{for } t \in \mathcal{T}$$
$$\mathit{dom}(\,\tau_{(q,A)}\,) = \{\varsigma\} \text{ where } \varsigma \in \mathfrak{S} \text{ is the unique slice s.t. } q \in \mathcal{P}^{\varsigma}$$
$$\mathit{dom}(\,\lightning^{(q,A)}_{[t_1,t_2]}\,) = \{\varsigma \in \mathfrak{S} \mid t_1 \in \mathcal{T}^{\varsigma} \;\vee\; t_2 \in \mathcal{T}^{\varsigma}\}$$

For each slice $\varsigma = (\mathcal{P}^{\varsigma}, \mathcal{T}^{\varsigma}, \mathcal{F}^{\varsigma}, \mathit{In}^{\varsigma}) \in \mathfrak{S}$, we define a local process $\mathscr{A}_{\varsigma} = (Q_{\varsigma}, q_{0,\varsigma}, \vartheta_{\varsigma})$ with $\vartheta_{\varsigma} \subseteq Q_{\varsigma} \times \Sigma_{\varsigma} \times Q_{\varsigma}$ as:

- $Q_{\varsigma} = \mathcal{P}^{\varsigma} \cup \{(q, A) \mid q \in \mathcal{P}^{\varsigma} \cap \mathcal{P}_{\mathcal{S}} \;\wedge\; A \subseteq \mathit{post}^{\varsigma}(q)\} \cup \{\bot_{\varsigma}\}$
- $q_{0,\varsigma}$ is the unique state s.t. $\mathit{In}^{\varsigma} = \{q_{0,\varsigma}\}$

and ϑ_{ς} is given by:

(1) $q \xmapsto{\tau_{(q,A)}} (q, A)$	**(2)** $q \xmapsto{t} q'$	**(3)** $(q, A) \xmapsto{t} q'$
$q \in \mathcal{P}_{\mathcal{S}} \,\wedge$ $A \subseteq \mathit{post}^{\varsigma}(q)$	$q \in \mathcal{P}_{\mathcal{E}} \;\wedge\; t \in \mathcal{T} \,\wedge$ $q \in \mathit{pre}^{\varsigma}(t) \;\wedge\; q' \in \mathit{post}^{\varsigma}(t)$	$q \in \mathcal{P}_{\mathcal{S}} \;\wedge\; t \in A \,\wedge$ $q \in \mathit{pre}^{\varsigma}(t) \;\wedge\; q' \in \mathit{post}^{\varsigma}(t)$
(4) $(q, A) \xmapsto{\lightning^{(q,A)}_{[t_1,t_2]}} \bot_{\varsigma}$	**(5)** $q \xmapsto{\lightning^{(q',A')}_{[t_1,t_2]}} \bot_{\varsigma}$	**(6)** $(q, A) \xmapsto{\lightning^{(q',A')}_{[t_1,t_2]}} \bot_{\varsigma}$
	$q' \notin Q_{\varsigma} \;\wedge\; q \in \mathcal{P}_{\mathcal{E}} \,\wedge$ $(t_1 \in \mathcal{T}^{\varsigma} \Rightarrow t_1 \in \mathit{post}^{\varsigma}(q)) \,\wedge$ $(t_2 \in \mathcal{T}^{\varsigma} \Rightarrow t_2 \in \mathit{post}^{\varsigma}(q))$	$q' \notin Q_{\varsigma} \,\wedge$ $(t_1 \in \mathcal{T}^{\varsigma} \Rightarrow t_1 \in A) \,\wedge$ $(t_2 \in \mathcal{T}^{\varsigma} \Rightarrow t_2 \in A)$

Define $\mathcal{A}_{\mathcal{G}} = \bigotimes_{\varsigma \in \mathfrak{S}} \mathscr{A}_{\varsigma}$ and the control game as $\mathcal{C}_{\mathcal{G}} = (\mathcal{A}_{\mathcal{G}}, \Sigma^{sys}, \Sigma^{env}, \{\mathcal{W}_{\varsigma}\}_{\varsigma \in \mathfrak{S}})$ where

- $\Sigma^{sys} = \{\tau_{(q,A)} \mid q \in \mathcal{P}_{\mathcal{S}} \;\wedge\; A \subseteq \mathit{post}^{\mathcal{G}}(q)\}$
- $\Sigma^{env} = \mathcal{T} \,\cup\, \{\, \lightning^{(q,A)}_{[t_1,t_2]} \mid q \in \mathcal{P}_{\mathcal{S}} \;\wedge\; A \subseteq \mathit{post}^{\mathcal{G}}(q) \;\wedge\; t_1, t_2 \in A \;\wedge\; t_1 \neq t_2\}$
- $\mathcal{W}_{\varsigma} = (\mathcal{W} \,\cap\, \mathcal{P}^{\varsigma}) \,\cup\, \{(q, A) \mid q \in (\mathcal{W} \cap \mathcal{P}^{\varsigma}) \;\wedge\; A \subseteq \mathit{post}^{\mathcal{G}}(q)\}$

Figure 3 The construction of the translated control game for a Petri game $\mathcal{G} = (\mathcal{P}_{\mathcal{S}}, \mathcal{P}_{\mathcal{E}}, \mathcal{T}, \mathcal{F}, \mathit{In}, \mathcal{W})$, distributed in slices $\{\varsigma\}_{\varsigma \in \mathfrak{S}}$, is depicted. Excluding the red parts, this is the definition of $\mathcal{C}_{\mathcal{G}}$. Including the red parts, this is the definition of $\widehat{\mathcal{C}_{\mathcal{G}}}$.

We transform every slice ς into a process that is described by a local automaton $\mathscr{A}_{\varsigma}$. Hence, we use the terms slice and process interchangeably. Every place in ς becomes a local state in $\mathscr{A}_{\varsigma}$. The process starts in the state that corresponds to the initial place of the slice. For every *system* place q, we furthermore add the aforementioned commitment sets. These are states (q, A) representing every possible commitment, i.e., every $A \subseteq \mathit{post}^{\mathcal{G}}(q)$.

Every transition t is added as an uncontrollable action. Action t involves all processes with slices synchronizing on t. To choose a commitment set, we furthermore add controllable actions (τ-actions) that are local to each process. We assume that each process chooses at most one commitment set. The transition relation ϑ_{ς} is given by three rules: From every *system* place $q \in \mathcal{P}_{\mathcal{S}}$, a process can choose a commitment set using the corresponding τ-action **(1)**. From an environment place $q \in \mathcal{P}_{\mathcal{E}}$, t can fire if q is in the precondition of t ($q \in \mathit{pre}^{\varsigma}(t)$). The process is then moved to the state q' that corresponds to the place that is reached when

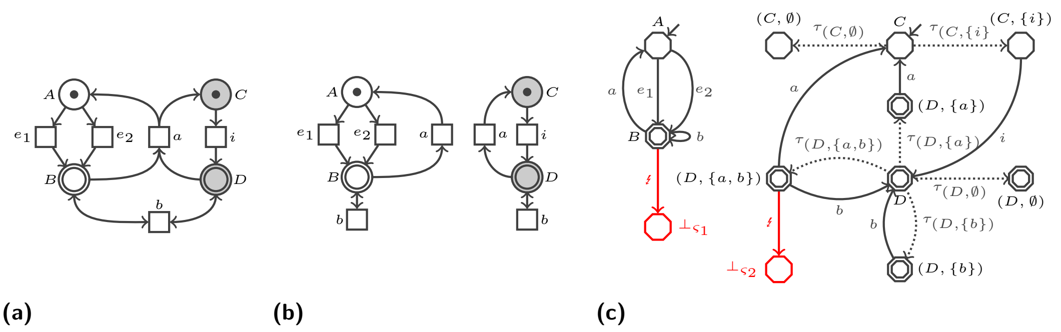


Figure 4 A sliceable Petri game $\mathcal{G}$ (a), a possible (in this case unique) distribution in slices $(\varsigma_1, \varsigma_2)$ (b), and the asynchronous automaton $\mathcal{C}_\mathcal{G}$ obtained by our translation (c). $\widehat{\mathcal{C}_\mathcal{G}}$ comprises additional $\bot$-states and one $\lightning^{(D,\{a,b\})}_{[a,b]}$-action (named $\lightning$) depicted in red.

firing t in the slice ($q' \in post^\varsigma(t)$) **(2)**. A process on an environment place can hence never restrict any actions; as in Petri games. For a system place, the rule is almost identical but only admits t if a commitment set has been chosen, that contains t **(3)**. Therefore, a process on a system place can control actions by choosing commitment sets; as in Petri games. States corresponding to winning places become winning states.

An example translation is depicted in Fig. 4. The Petri game (a) comprise two players starting in A and C. They can move to B and D using e_1, e_2, or i and afterwards synchronize on a or b. The Petri game can be distributed into slices (b). In the control game from our construction (c), the slice containing only environment places results in the local process on the left. For the system places in the other slice, commitment sets are added as states $\{C\} \times 2^{\{i\}}$ and $\{D\} \times 2^{\{a,b\}}$. The process can choose them using controllable τ-actions and the actions a, b, and i can only occur if included in the current set. The construction guarantees that only the second process can control transitions a and b, as in the Petri game.

Non-Determinism. In deterministic strategies, every system place allows transitions s.t. in every situation, there is at most one of them enabled. In $\mathcal{C}_\mathcal{G}$, the controller can choose arbitrary commitment sets and, thus, a winning controller can result in a *non-deterministic* strategy for $\mathcal{G}$. To ensure deterministic strategies, we want to penalize situations where a commitment set in $\mathcal{C}_\mathcal{G}$ is chosen s.t. two or more distinct actions from this set can be taken.

To achieve this, we define the modified game $\widehat{\mathcal{C}_\mathcal{G}}$. We equip each process with a $\bot$-state from which no winning configurations are reachable. Uncontrollable $\lightning$-actions move processes to $\bot$-states and thereby cause the system to lose. The situation to be covered comprises a process that has chosen a commitment set, i.e., is in a state (q, A), and two distinct actions t_1 and t_2 in A. For every such combination, we add a $\lightning^{(q,A)}_{[t_1,t_2]}$-action that involves all processes participating in t_1 or t_2 and can be taken exactly if (q, A) is a current state and both t_1 *and* t_2 could occur from the current global state. The three rules in ϑ add the $\lightning^{(q,A)}_{[t_1,t_2]}$-action to each process. It fires if one process is in state (q, A) **(4)** and all other involved processes are in states such that both $t_1, t_2 \in A$ are possible **(4, 5)**. To ensure that t_1 and t_2 can both be taken, we distinguish between system and environment places: Every process ς on an environment place needs to be in the right state, i.e., if ς is involved in t_i ($t_i \in \mathcal{T}^\varsigma$), then t_i is in the postcondition of its current place for $i = 1, 2$ **(5)**. If on a system place, t_i must not only be in the postcondition but also in the currently chosen commitment set **(6)**.

Size. In both $\mathcal{C}_{\mathcal{G}}$ and $\widehat{\mathcal{C}_{\mathcal{G}}}$, the size of the alphabet and number of local states is exponential in the number of transitions and linear in the number of places. For $\mathcal{C}_{\mathcal{G}}$, the blow-up in the alphabet can be kept polynomial by using a tree construction to choose commitment sets. For a bound on the number of outgoing transitions, both $\mathcal{C}_{\mathcal{G}}$ and $\widehat{\mathcal{C}_{\mathcal{G}}}$ are of polynomial size.

5.2 Correctness

We show that our translation yields strategy-equivalent games by outlining the translation of winning strategies and controllers between $\mathcal{G}$ and $\mathcal{C}_{\mathcal{G}}$. Both game types rely on causal information, i.e., a strategy/controller bases its decisions on every action/transition it took part in as well as all information it received upon communication. In $\mathcal{G}$, the information is carried by individual tokens. In our translation, we transform each slice for a token into a process that is involved in exactly the transitions that the slice it is build from takes part in, i.e., we preserve the communication architecture. At every point, all processes in $\mathcal{C}_{\mathcal{G}}$ possess the same information as their counterpart slices. Using the commitment set, our translation ensures that only processes on a state based on a system place can control any behavior. Therefore, a process and its counterpart slice have the same possibilities for control.

Translating a Strategy for $\mathcal{G}$ to a Controller for $\mathcal{C}_{\mathcal{G}}$. Given a winning strategy σ, we construct a controller ϱ_σ. The only states from which a process p can control any behavior (in terms of controllable actions) are of the form $q \in \mathcal{P}_{\mathcal{S}}$. σ decides for every system place which transitions to enable. Due to our construction, p can copy the decision of σ by choosing an appropriate commitment set. Therefore, ϱ_σ allows the same behavior as σ. If σ is deterministic then the commitments sets are chosen such that no $\lightning$-actions are possible.

Translating a Controller for $\mathcal{C}_{\mathcal{G}}$ to a Strategy for $\mathcal{G}$. Given a winning controller ϱ, we incrementally construct a strategy σ_ϱ. Every system place q in the partially constructed strategy can control which transitions are enabled. The place q belongs to some process. If on a state corresponding to a system place, this process can control all actions using its commitment sets. q enables exactly the transitions that the process has chosen as a commitment set. An environment place cannot control any behavior and neither can the process it belongs to. Hence, ϱ and σ_ϱ allow the same actions and transitions. A winning controller for $\widehat{\mathcal{C}_{\mathcal{G}}}$ additionally avoids any uncontrollable $\lightning$-actions and results in a deterministic strategy.

We obtain that $\mathcal{G}$ and $\mathcal{C}_{\mathcal{G}}$ are strategy-equivalent and that $\mathcal{G}$ and $\widehat{\mathcal{C}_{\mathcal{G}}}$ are strategy-equivalent if we require deterministic strategies for Petri games.

5.3 Generalization to Concurrency-Preserving Games

Our translation builds processes from a slice distribution of the Petri game. This limits the translation to sliceable games. The notion of slices is too strict: Our translation only requires to distribute the global behavior of the Petri game into local behavior, a *partitioning* of the places is not necessarily needed. We introduce the new concept of *singular nets* (SN). Similar to a slice, an SN describes the course of one token. Instead of being a subnet, it is equipped with a labeling function assigning to each node in the singular net a node in the original net. This labeling allows us to split up places and transitions by equally labelled copies enabling us to distribute every concurrency-preserving Petri net and game into singular nets. We can build our previous translation with an SN-distribution instead of a slice-distribution.

Define $\mathcal{G}_{\mathcal{C}} = (\mathcal{P}_{\mathcal{S}}, \mathcal{P}_{\mathcal{E}}, \mathcal{T}, \mathcal{F}, In, \mathcal{B})$ where

- $\mathcal{P}_{\mathcal{S}} = \bigcup_{p\in\mathfrak{P}} S_p, \quad \mathcal{P}_{\mathcal{E}} = \{\, (s, A) \mid s \in \bigcup_{p\in\mathfrak{P}} S_p,\ A \subseteq act(s) \cap \Sigma^{sys}\} \cup \{\bot^p_{DL} \mid p \in \mathfrak{P}\}$
- $\mathcal{T} = \{\, (a, B, \{A_s\}_{s\in B}) \mid a \in \Sigma^{env},\ B \in domain(\delta_a),\ A_s \subseteq act(s) \cap \Sigma^{sys}\} \cup$ **(1)**

  $\{\, (a, B, \{A_s\}_{s\in B}) \mid a \in \Sigma^{sys},\ B \in domain(\delta_a),\ A_s \subseteq act(s) \cap \Sigma^{sys},\ a \in A_s\} \cup$ **(2)**

  $\{\, \tau_{(s,A)} \mid s \in \bigcup_{p\in\mathfrak{P}} S_p,\ A \subseteq act(s) \cap \Sigma^{sys}\} \cup \{t^M_{DL} \mid M \in \mathfrak{D}_{DL}\}$ **(3), (7)**
- $\mathcal{F} = \{\, \big((s, A)\, ,\, (a, B, \{A_s\}_{s\in B})\big) \mid s \in B,\ A_s = A\} \cup$ **(4)**

  $\{\, \big((a, B, \{A_s\}_{s\in B})\, ,\, s'\,\big) \mid s' \in \delta_a(B)\} \cup$ **(5)**

  $\{\, \big(s, \tau_{(s,A)}\big)\} \cup \{\big(\tau_{(s,A)}, (s, A)\big)\,\} \cup$ **(6)**

  $\{\, \big(q, t^M_{DL}\big) \mid q \in M\} \cup \{\, \big(t^M_{DL}, \bot^p_{DL}\big) \mid p \in \mathfrak{P}\}$ **(8)**
- $In = s^{\mathcal{A}}_{in}$ and $\mathcal{B} = \bigcup_{p\in\mathfrak{P}} \mathcal{B}_p \cup \bigcup_{p\in\mathfrak{P}} \bot^p_{DL}$

Figure 5 We give the construction of the translated Petri game $\mathcal{G}_{\mathcal{C}}$ for a control game $\mathcal{C} = (\mathcal{A}, \Sigma^{sys}, \Sigma^{env}, \{\mathcal{B}_p\}_{p\in\mathfrak{P}})$ where $\mathcal{A} = (\{S_p\}_{p\in\mathfrak{P}}, s^{\mathcal{A}}_{in}, \{\delta_a\}_{a\in\Sigma})$. The initial state $s^{\mathcal{A}}_{in}$ is viewed as a set. The gray parts penalize the artificial deadlocks, i.e., all markings in $\mathfrak{D}_{DL}$.

▶ **Theorem 3.** *For every concurrency-preserving Petri game $\mathcal{G}$, there exist control games $\mathcal{C}_{\mathcal{G}}$ and $\widehat{\mathcal{C}_{\mathcal{G}}}$ with an equal number of players such that* **(1)** *$\mathcal{G}$ and $\mathcal{C}_{\mathcal{G}}$ are strategy-equivalent and* **(2)** *$\mathcal{G}$ and $\widehat{\mathcal{C}_{\mathcal{G}}}$ are strategy-equivalent if we require deterministic Petri game strategies.*

5.4 Lower Bound

We can show that there is a family of Petri games such that every strategy-equivalent control game must have exponentially many local states. In a control game, either all or none of the players can restrict an action. By contrast, Petri games offer a finer granularity of control by allowing only some players to restrict a transition. The insight for the lower bound is to create a situation where a transition is shared between players but can only be controlled by one of them. Using careful reasoning, we can show that in any strategy-equivalent control game there must be actions that can only be controlled by a single process, resulting in exponentially many local states. Our translation shows that the difference between both formalism can be overcome but our lower bound shows an intrinsic difficulty to achieve this.

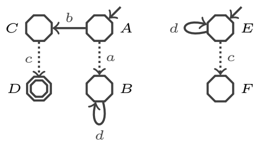
▶ **Theorem 4.** *There is a family of Petri games such that every strategy-equivalent control game (with an equal number of players) must have at least $\Omega(d^n)$ local states for $d > 1$.*

6 Translating Control Games to Petri Games

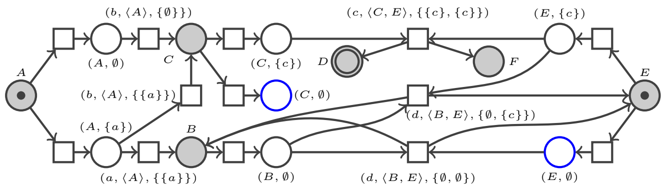
We give our translation from control games to Petri games, prove that it yields strategy-equivalent games, and give an exponential lower bound. We present our translation for safety objectives. All proofs and further details can be found in the full version of this paper [3].

6.1 Construction

We fix a control game $\mathcal{C} = (\mathcal{A}, \Sigma^{sys}, \Sigma^{env}, \{\mathcal{B}_p\}_{p\in\mathfrak{P}})$ with safety objective. The translation to $\mathcal{G}_{\mathcal{C}}$ is depicted in Fig. 5. We represent each local state s as a system place. We add environment places (s, A), which encode every possible commitment set of actions that can be allowed by a controller ($A \subseteq act(s) \cap \Sigma^{sys}$). From each system place, the player can move to



(a)


$(b, \langle A\rangle, \{\emptyset\}\})$
$(c, \langle C, E\rangle, \{\{c\}, \{c\}\})$
$(E, \{c\})$
$(A, \emptyset)$
C
$(C, \{c\})$
D
F
A
E
$(b, \langle A\rangle, \{\{a\}\})$
$(C, \emptyset)$
$(d, \langle B, E\rangle, \{\emptyset, \{c\}\})$
$(A, \{a\})$
B
$(a, \langle A\rangle, \{\{a\}\})$
$(B, \emptyset)$
$(d, \langle B, E\rangle, \{\emptyset, \emptyset\})$
$(E, \emptyset)$

(b)

Figure 6 Control game $\mathcal{C}$ (a) and translated Petri game $\mathcal{G}_{\mathcal{C}}$ (b) are given. Commitment sets without outgoing transitions are omitted. The set of artificial deadlocks $\mathfrak{D}_{DL}$ comprises every final marking that contains at least one blue place. The resulting t_{DL}^{M}-transitions are omitted.

places for the commitment sets using a $\tau_{(s,A)}$-transition **(3, 6)**. Each action a in $\mathcal{C}$ can occur from different configurations of the processes in $dom(a)$, i.e., all states in $domain(\delta_a)$, whereas in Petri games transitions fire from fixed preconditions. We want to represent a as a transition that fires from places representing commitment sets that correspond to configurations from which a can occur in $\mathcal{C}$. We hence duplicate a into multiple transitions to account for every configuration in $domain(\delta_a)$ and for every combination of commitment sets. Transitions have the form $(a, B, \{A_s\}_{s\in B})$ where a is the action in the control game, $B \in domain(\delta_a)$ is the configuration from which a can fire, and $\{A_s\}_{s\in B}$ are the involved commitment sets. If action a is uncontrollable the corresponding transitions are added independently of the commitment sets **(1)**. If a is controllable a transition is only added if a is in the commitment sets of all involved players, i.e., $a \in A_s$ for every $s \in B$ **(2)**. If $(a, B, \{A_s\}_{s\in B})$ is added it fires from precisely the precondition that is encoded in it, i.e., the places (s, A) where $s \in B$ and $A_s = A$, and moves every token to the system place that corresponds to the resulting local state when firing a in $\mathcal{C}$ **(4, 5)**. A strategy can restrict controllable actions by moving to an appropriate commitment set but cannot forbid uncontrollable ones, since they can occur from every combination of commitment sets. If a system player decides to refuse any commitment set it could prohibit transitions that correspond to uncontrollable actions. In Sec. 6.3, we show how to force the system to always choose a commitment set.

In safety games, every winning strategy must avoid deadlocks. By introducing explicit commitment sets, we add *artificial* deadlocks, i.e., configurations that are deadlocked in $\mathcal{G}_{\mathcal{C}}$ but where the corresponding state in $\mathcal{C}$ could still act. This permits trivial strategies that, e.g., always choose the empty commitment set. We define $\mathfrak{D}_{DL}$ as the set of all reachable markings that are final in $\mathcal{G}_{\mathcal{C}}$ but where the corresponding global state in $\mathcal{C}$ can still perform an action, i.e., all artificial deadlocks. Similar to the $\not{\iota}$-actions, we introduce t_{DL}^{M}-transitions that fire from every marking M in $\mathfrak{D}_{DL}$ and move every token to a losing place $\bot_{DL}$ **(7, 8)**. The mechanism to detect artificial deadlocks is depicted as the gray parts in Fig. 5.

Figure 6 depicts an example translation. The system cannot win this game: The uncontrollable action b can always happen, independent of the commitment set for place A. If one of the two tokens refuses c (moves to a blue place) a (losing) transition t_{DL} can fire.

6.2 Correctness

We show strategy-equivalence of $\mathcal{C}$ and $\mathcal{G}_{\mathcal{C}}$ by translating strategies (that always commit) and controllers between both of them. We observe that each token moves on the local states of one process and takes part in precisely the actions of the process. At every point, a token hence possesses the same local information as the process. A token can restrict the controllable actions using the commitment sets but cannot restrict the uncontrollable ones. The token therefore has the same possibilities as the process counterpart.

Translating Controllers to Strategies. Given a winning controller ϱ, we incrementally build a (possibly infinite) winning, deterministic strategy σ_ϱ. Every system place q in a partially constructed strategy can choose one of the commitment sets. q copies ϱ by committing to exactly the actions that the process it belongs to has allowed. The commitment sets can only restrict controllable actions, as the process can. Hence, σ_ϱ allows the same behavior as ϱ.

Translating Strategies to Controllers. Given a winning, deterministic strategy σ, we construct a winning controller ϱ_σ. A process p that resides on a local state s can decide which of the controllable actions should be allowed. Every token in σ can decide for a commitment set and therefore implicitly chooses which controllable actions should be enabled. p allows exactly the actions that σ chooses as a commitment set. Both can only restrict controllable actions and, by copying, ϱ_σ achieves the same behavior as σ.

▶ **Theorem 5.** *$\mathcal{C}$ and $\mathcal{G}_\mathcal{C}$ are strategy-equivalent.*

6.3 Enforcing Commitment

Our construction assumes wining strategies to always choose a commitment set. We can modify $\mathcal{G}_\mathcal{C}$ such that every non-committing strategy cannot win. The insight is to use the deadlock-avoidance of winning strategies. Deadlocks define a *global* situation of the game. To enforce commitment, we require *local* deadlock-avoidance in the sense that every token has to choose a commitment set. This is not prevented by global deadlock-avoidance, where, e.g., a single player being able to play locally enables every other player to refuse to commit without being deadlocked. We reduce local to global deadlocks by adding transitions to challenge the players to have reached a local deadlock. Using challenge transitions, every player currently residing on a place that corresponds to a chosen commitment set moves to a terminating place. Every player that has chosen commitment sets can terminate, resulting in the players that are locally deadlocked to cause a global deadlock. Although the challenge is always possible, the scheduler decides the point of challenge. The game with the added challenger has a winning strategy iff $\mathcal{G}_\mathcal{C}$ has a winning strategy that always commits.

6.4 Lower Bounds

We can provide a family of control games where every strategy-equivalent Petri game must be of exponential size. In control games, both controllable and uncontrollable actions can occur from the same state. In Petri games, a given place can either restrict all transitions (system place) or none. A control game where both actions types are possible already results in Petri games of exponential size. We assume the absence of infinite τ-sequences.

▶ **Theorem 6.** *There is a family of control games such that every strategy-equivalent Petri game (with an equal number of players) must have at least $\Omega(d^n)$ places for $d > 1$.*

7 New Decidable Classes

We exemplarily show one transferrable class of decidability for both control games and Petri games to highlight the applicability of our translations.

New Decidable Control Games. A process in a control game is an *environment process* if all its action are uncontrollable. A *system process* is one that is not an environment process. We can modify our second translation by not adding system places if there are no outgoing controllable actions. Therefore, environment processes do not add system places to the Petri game and we can use the results from [10].

▶ **Corollary 7.** *Control games with safety objectives and one system process are decidable.*

New Decidable Petri Games. Given a Petri game $\mathcal{G}$ and a distribution into slices (or SNs) $\{\varsigma\}_{\varsigma \in \mathbb{S}}$, we analyze the communication structure between the slices by building the undirected graph (V, E) where $V = \mathbb{S}$ and $E = \{(\varsigma_1, \varsigma_2) \mid \mathcal{T}^{\varsigma_1} \cap \mathcal{T}^{\varsigma_2} \neq \emptyset\}$. (V, E) is isomorphic to the *communication architecture* of the constructed asynchronous automaton $\mathcal{C}_{\mathcal{G}}$ (as introduced in [13]). We define $\mathcal{G}_{\circledast}$ as every Petri game that has a distribution $\{\varsigma\}_{\varsigma \in \mathbb{S}}$ where (V, E) is *acyclic*. We can show that such distributions are hard to find. From [13], we obtain decidability.

▶ **Lemma 8.** *Deciding whether a Petri net has an acyclic slice-distribution is* NP*-complete.*

▶ **Corollary 9.** *Petri games in $\mathcal{G}_{\circledast}$ with reachability objectives are decidable.*

8 Conclusion

We have provided the first formal connection between control games and Petri games by showing that both are equivalent. This indicates that synthesis models for asynchronous systems with causal memory are stable under the concrete formalisms of system and environment responsibilities for the two most common models. Conversely, our lower bounds show an intrinsic difference between control games and Petri games. By our translations, existing and future decidability results can be combined and transferred between both game types. Our translations could be adapted to other winning objectives. An interesting direction for future work is to investigate how action-based control games [21] relate to Petri games and to study unified models that combine features from control games and Petri games.

References

1 Cyril Autant and Philippe Schnoebelen. Place Bisimulations in Petri Nets. In *Proceedings of Application and Theory of Petri Nets*, pages 45–61, 1992. doi:10.1007/3-540-55676-1_3.

2 Eike Best, Raymond R. Devillers, Astrid Kiehn, and Lucia Pomello. Concurrent Bisimulations in Petri Nets. *Acta Inf.*, 28(3):231–264, 1991. doi:10.1007/BF01178506.

3 Raven Beutner, Bernd Finkbeiner, and Jesko Hecking-Harbusch. Translating Asynchronous Games for Distributed Synthesis (Full Version). *arXiv preprint*, 2019. arXiv:1907.00829.

4 J. Richard Buchi and Lawrence H. Landweber. Solving sequential conditions by finite state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.

5 Joost Engelfriet. Branching Processes of Petri Nets. *Acta Inf.*, 28(6):575–591, 1991. doi:10.1007/BF01463946.

6 Javier Esparza and Keijo Heljanko. *Unfoldings – A Partial-Order Approach to Model Checking*. Springer, 2008. doi:10.1007/978-3-540-77426-6.

7 Bernd Finkbeiner. Bounded Synthesis for Petri Games. In *Proceedings of Correct System Design - Symposium in Honor of Ernst-Rüdiger Olderog on the Occasion of His 60th Birthday*, pages 223–237, 2015. doi:10.1007/978-3-319-23506-6_15.

8 Bernd Finkbeiner, Manuel Gieseking, Jesko Hecking-Harbusch, and Ernst-Rüdiger Olderog. Symbolic vs. Bounded Synthesis for Petri Games. In *Proceedings of SYNT@CAV*, pages 23–43, 2017. doi:10.4204/EPTCS.260.5.

9 Bernd Finkbeiner, Manuel Gieseking, and Ernst-Rüdiger Olderog. Adam: Causality-Based Synthesis of Distributed Systems. In *Proceedings of CAV*, pages 433–439, 2015. doi:10.1007/978-3-319-21690-4_25.

10 Bernd Finkbeiner and Paul Gölz. Synthesis in Distributed Environments. In *Proceedings of FSTTCS*, pages 28:1–28:14, 2017. doi:10.4230/LIPIcs.FSTTCS.2017.28.

11 Bernd Finkbeiner and Ernst-Rüdiger Olderog. Petri games: Synthesis of distributed systems with causal memory. *Inf. Comput.*, 253:181–203, 2017. doi:10.1016/j.ic.2016.07.006.

12 Paul Gastin, Benjamin Lerman, and Marc Zeitoun. Distributed Games with Causal Memory Are Decidable for Series-Parallel Systems. In *Proceedings of FSTTCS*, pages 275–286, 2004. doi:10.1007/978-3-540-30538-5_23.

13 Blaise Genest, Hugo Gimbert, Anca Muscholl, and Igor Walukiewicz. Asynchronous Games over Tree Architectures. In *Proceedings of ICALP*, pages 275–286, 2013. doi:10.1007/978-3-642-39212-2_26.

14 Hugo Gimbert. On the Control of Asynchronous Automata. In *Proceedings of FSTTCS*, pages 30:1–30:15, 2017. doi:10.4230/LIPIcs.FSTTCS.2017.30.

15 Jesko Hecking-Harbusch and Niklas O. Metzger. Efficient Trace Encodings of Bounded Synthesis for Asynchronous Distributed Systems. In *Proceedings of ATVA*, 2019.

16 P. Madhusudan and P. S. Thiagarajan. A Decidable Class of Asynchronous Distributed Controllers. In *Proceedings of CONCUR*, pages 145–160, 2002. doi:10.1007/3-540-45694-5_11.

17 P. Madhusudan, P. S. Thiagarajan, and Shaofa Yang. The MSO Theory of Connectedly Communicating Processes. In *Proceedings of FSTTCS*, pages 201–212, 2005. doi:10.1007/11590156_16.

18 José Meseguer, Ugo Montanari, and Vladimiro Sassone. Process versus Unfolding Semantics for Place/Transition Petri Nets. *Theor. Comput. Sci.*, 153(1&2):171–210, 1996. doi:10.1016/0304-3975(95)00121-2.

19 Anca Muscholl. Automated Synthesis of Distributed Controllers. In *Proceedings of ICALP*, pages 11–27, 2015. doi:10.1007/978-3-662-47666-6_2.

20 Anca Muscholl and Igor Walukiewicz. Distributed Synthesis for Acyclic Architectures. In *Proceedings of FSTTCS*, pages 639–651, 2014. doi:10.4230/LIPIcs.FSTTCS.2014.639.

21 Anca Muscholl, Igor Walukiewicz, and Marc Zeitoun. A look at the control of asynchronous automata. *Perspectives in Concurrency Theory*, pages 356–371, 2009.

22 Mogens Nielsen, Gordon D. Plotkin, and Glynn Winskel. Petri Nets, Event Structures and Domains, Part I. *Theor. Comput. Sci.*, 13:85–108, 1981. doi:10.1016/0304-3975(81)90112-2.

23 Ernst-Rüdiger Olderog. *Nets, terms and formulas: three views of concurrent processes and their relationship*, volume 23. Cambridge University Press, 2005.

24 Amir Pnueli and Roni Rosner. Distributed Reactive Systems Are Hard to Synthesize. In *31st Annual Symposium on Foundations of Computer Science, 1990, Volume II*, pages 746–757, 1990. doi:10.1109/FSCS.1990.89597.

25 Wolfgang Reisig. *Petri Nets: An Introduction.* Springer, 1985. doi:10.1007/978-3-642-69968-9.

26 Wieslaw Zielonka. Notes on Finite Asynchronous Automata. *ITA*, 21(2):99–135, 1987. doi:10.1051/ita/1987210200991.