(Un)Decidability Bounds of the Synthesis Problem for Petri Games

Paul Hannibal
University of Oldenburg,
Lower Saxony, Germany
paul.jonathan.hannibal1@uni-oldenburg.de

Petri games are a multi-player game model for the automatic synthesis of distributed systems, where the players are represented as tokens on a Petri net and are grouped into environment players and system players. 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 entire causal history of the other players.

We show that the synthesis problem for two-player Petri games under a global safety condition is NP-complete and it can be solved within a non-deterministic exponential upper bound in the case of up to 4 players. Furthermore, we show the undecidability of the synthesis problem for Petri games with at least 6 players under a local safety condition.

1 Introduction

A Petri game is a model for distributed, reactive systems. It is played on a Petri net where each place is either a system place or an environment place. The tokens on system places are system players and they control which transitions to take next. The tokens on environment places are uncontrollable environment players. Essential for Petri games is the informedness of the players. As long as the players move in independent parts of the net, they do not know of each other; when taking a joint transition they exchange information about their complete causal history.

A winning strategy of the system players reacts to all options of the environment players while satisfying a winning condition. Thereby, a decision of a system player is based on its causal history, which grows infinitely in a Petri net with loops. Different causal histories allow different decisions. The *synthesis problem* asks whether there is a winning strategy of the system players. There have been several positive results on deciding the synthesis problem for Petri games, obtained by restricting the number of players [8, 7] or restricting the concurrent behaviour [13]. Also, an approach to bounded synthesis has been proposed [6]. These papers considered as winning conditions either local safety conditions where some 'bad' places must be avoided or global safety conditions requiring that some sets of places are considered as 'bad' markings that must not be reached simultaneously.

Petri games are related to the model of control games played by multiple processes on Zielonka automata. These games are also based on the causal memory of their processes. A control game is a composition of local processes. The processes communicate via shared actions that are either controllable or uncontrollable. A strategy consists of one local controller for each process that can restrict controllable actions based on the causal past of the process. As in Petri games, a strategy must take into account all uncontrollable behaviour in order to win. Unlike Petri games, one of the most common winning conditions is a local termination condition. A formal relationship of the two models has been presented in [1], where translations from Petri games into control games and back have been presented

A. Achilleos and D. Della Monica (Eds.): Fourteenth International Symposium on Games, Automata, Logics, and Formal Verification (GandALF 2023).
EPTCS 390, 2023, pp. 115–131, doi:10.4204/EPTCS.390.8

such that there is a weak bisimulation between the winning strategies of the two models. When translating a control game into a Petri game, the processes are turned one-by-one into players. These players switch between system and environment players. The winning condition stays the same. The number of places (or states) blows up exponentially when a game is translated in either direction.

Decidability results for control games have been obtained by restricting the communication architecture [17, 10] or restricting the concurrent behaviour [15, 16]. Another class of decidable control games are decomposable games [11] that come with a local termination condition. Decomposable games are decidable with up to 4 players [11]. In Sec. 3, we show that the synthesis problem for Petri games under a global safety condition with up to 4 players is decidable within an exponential upper time bound, and for two-player Petri games this problem is NP-complete.

In [12], it has been shown that the synthesis problem is undecidable for control games with at least 6 processes under a local termination condition. This result, together with the translation into Petri games in [1], implies that there are 6-player Petri games that are equipped with a local termination condition for which the synthesis problem is undecidable. In Sec. 4, we show in a direct proof that the synthesis problem for Petri games with 6 players is also undecidable under a local safety condition.

The synthesis problem is of great interest because it automates the error-prone implementation process while delivering implementations that are correct by construction. It was first introduced in [3]. Pnueli and Rosner introduced a setting of synchronous processes that communicate via shared variables [19]. For a single process, this setting is known to be decidable [2, 18]. For multiple processes, the setting of Pnueli and Rosner is known to be undecidable [19]. There have been positive decidability results on specific architectures with multiple processes, including pipelines [20], rings [14], and acyclic architectures [9]. However, all the positive results for multiple processes have non-elementary complexity.

2 Foundations

In this section, we define branching processes and unfoldings as in [4]. Also, we define Petri games and their winning strategies as in [8].

Some notation: the *power set* of a set A is denoted by $2^A = \{B \mid B \subseteq A\}$, the *set of nonempty finite subsets* of A by $2^A_{nf} = \{B \mid B \subseteq A \wedge B \neq \emptyset \wedge B \text{ is finite}\}$, and the *set of finite subsets* of A by 2^A_f.

A *Petri net* or simply *net* is a structure $N = (\mathscr{P}, \mathscr{T}, pre, post, In)$, where $\mathscr{P}$ is the (possibly infinite) set of *places*, $\mathscr{T}$ is the (possibly infinite) set of *transitions*, *pre* and *post* are *flow mappings*, $In \subseteq \mathscr{P}$ is the *initial marking*, and the following properties hold: $\mathscr{P} \cap \mathscr{T} = \emptyset$, $pre : \mathscr{T} \to 2^{\mathscr{P}}_{nf}$, $post : \mathscr{T} \to 2^{\mathscr{P}}_f$. A Petri net is called finite if $\mathscr{P} \cup \mathscr{T}$ is a finite set. The flow mappings *pre* and *post* are extended to places as usual: $\forall p \in \mathscr{P} : pre(p) = \{t \in \mathscr{T} \mid p \in post(t)\}$ and $\forall p \in \mathscr{P} : post(p) = \{t \in \mathscr{T} \mid p \in pre(t)\}$. A *marking* M of a Petri net N is a *multiset* over $\mathscr{P}$. In particular, In is a marking. By convention, a net named N has the components $(\mathscr{P}, \mathscr{T}, pre, post, In)$, and analogously for net with decorated names like N_0, N_1, N_2, where the components are equally decorated.

A transition $t \in \mathscr{T}$ is *enabled* at marking M if $pre(t) \subseteq M$. If t is enabled, the transition t can be *fired*, such that the new marking is $M' = M - pre(t) + post(t)$. This is denoted as $M|t\rangle M'$. This notation is extended to sequences of enabled transitions $M|t_1 \dots t_n\rangle M'$. A marking M is *reachable* if there exists a sequence of successively enabled transitions $(t_k)_{k\in\{1,\dots,n\}}$ and $In|t_1 \dots t_n\rangle M$. This sequence can be empty. The set of all reachable markings of a net N is denoted as $\mathscr{R}(N)$. A Petri net N is called *safe*, if for all reachable markings $M(p) \leq 1$ holds for all $p \in \mathscr{P}$. Then, M is a subset of P. All Petri nets considered in this paper are safe.

A *node* x is a place or a transition $x \in \mathscr{P} \cup \mathscr{T}$. The binary *flow relation* $\mathscr{F}$ on nodes is defined as follows: $x \mathscr{F} y$ if $x \in pre(y)$. A node x is a *causal predecessor* of y, denoted as $x \leq y$, if $x \mathscr{F}^+ y$. Furthermore, $x \leq x$ holds for all $x \in \mathscr{P} \cup \mathscr{T}$. Two nodes $x, y \in \mathscr{P} \cup \mathscr{T}$ are *causally related*, if $x \leq y$ or $y \leq x$ holds. We say x is a *causal successor* of y, if $y \leq x$ holds.

Two nodes $x_1, x_2 \in \mathscr{P} \cup \mathscr{T}$ are *in conflict*, denoted $x_1 \# x_2$, if there exist two transitions $t_1, t_2 \in \mathscr{T}$, $t_1 \neq t_2$ with $pre(t_1) \cap pre(t_2) \neq \emptyset$ and $t_i \leq x_i$, $i = 1,2$. A node $x \in \mathscr{P} \cup \mathscr{T}$ is in self-conflict if $x \# x$. Informally speaking, two nodes are in conflict if two transitions exist that share some place in their presets and each node is a causal successor of one of those transitions. Two nodes $x, y \in \mathscr{P} \cup \mathscr{T}$ are *concurrent*, denoted $x || y$, if they are neither causally related nor in conflict.

A Petri net N is *finitely preceded*, if for every node $x \in \mathscr{P} \cup \mathscr{T}$ the set $\{y \in \mathscr{P} \cup \mathscr{T} \mid y \leq x\}$ is finite. That set is the causal history of a node. A Petri net N is *acyclic*, if the directed graph $(\mathscr{P} \cup \mathscr{T}, \mathscr{F})$ is acyclic. The following definitions lead to the definition of a branching process.

An *occurrence net* is a Petri net N with the following properties: N is acyclic, finitely preceded, $\forall p \in \mathscr{P} : |pre(p)| \leq 1$, no transition $t \in \mathscr{T}$ is in self-conflict, and $In = \{p \in \mathscr{P} \mid pre(p) = \emptyset\}$.

A homomorphism from one Petri net to another maps each node to a node such that the preset and postset relations are preserved including the initial marking. Formally, let N_1 and N_2 be two Petri nets. Then a *homomorphism* from N_1 to N_2 is a mapping $h : \mathscr{P}_1 \cup \mathscr{T}_1 \to \mathscr{P}_2 \cup \mathscr{T}_2$ with following properties: $h(\mathscr{P}_1) \subseteq \mathscr{P}_2$ and $h(\mathscr{T}_1) \subseteq \mathscr{T}_2$, for all transitions $t \in \mathscr{T}_1$, h restricted to $pre_1(t)$ is a bijection between $pre_1(t)$ and $pre_2(h(t))$, for all transitions $t \in \mathscr{T}_1$, h restricted to $post_1(t)$ is a bijection between $post_1(t)$ and $post_2(h(t))$, and the restriction of h to In_1 is a bijection between In_1 and In_2. An *isomorphism* is a bijective homomorphism.

A run is represented by a (possibly infinite) firing sequence of transitions. A branching process of a Petri net represents (possibly) multiple runs of the underlying Petri net.

Branching process. A *branching process* of a net N_0 is a pair $B = (N, \pi)$, where N is an occurrence net and π a homomorphism from N to N_0 such that: $(*)$ For all $t_1, t_2 \in \mathscr{T}$: if $pre(t_1) = pre(t_2)$ and $\pi(t_1) = \pi(t_2)$, then $t_1 = t_2$.

An example of a Petri net and a branching process is shown in Fig. 1.

The notion of the set of all reachable markings of a branching process $B = (N, \pi)$ is extended to $\mathscr{R}(B) = \mathscr{R}(N)$. By convention, a branching Process B has the components (N_B, π_B). Throughout this paper, B_1 and B_2 are branching processes of an underlying net N_0. The property $(*)$ of the definition of a branching process ensures that every run of the Petri net is represented at most once. Informally speaking, a run only consists of concurrent and causally related nodes and a node can be part of multiple runs. Nodes that are in conflict, cannot belong to the same run.

Homomorphism on branching processes. A *homomorphism* from B_1 to B_2 is a homomorphism h from N_1 to N_2 such that $\pi_2 \circ h = \pi_1$. The branching processes B_1 and B_2 are *isomorphic* if there exists an isomorphism from B_1 to B_2 which is denoted as $B_1 \cong B_2$.

A natural partial order on branching processes is defined in the following.

Subprocess relation of branching processes. B_1 approximates B_2, denoted by $B_1 \leq B_2$, if there exists an injective homomorphism from B_1 to B_2.

Now we define the unfolding of a net as the maximal branching process that contains all (possibly infinite) runs of a net. Loosely speaking, the unfolding of a net is an acyclic net with the same behaviour as the original net, but where each place and transition has a unique causal history.

Unfolding. The *unfolding* $unf(N_0)$ of a Petri net N_0 is the maximal branching process with respect to the subprocess relation $\leq$ of branching processes. This definition is unique up to isomorphism. We refer to the components of the unfolding as $\mathscr{T}_{unf(N_0)}$, $\mathscr{P}_{unf(N_0)}$, $pre_{unf(N_0)}$, $post_{unf(N_0)}$, and $In_{unf(N_0)}$.

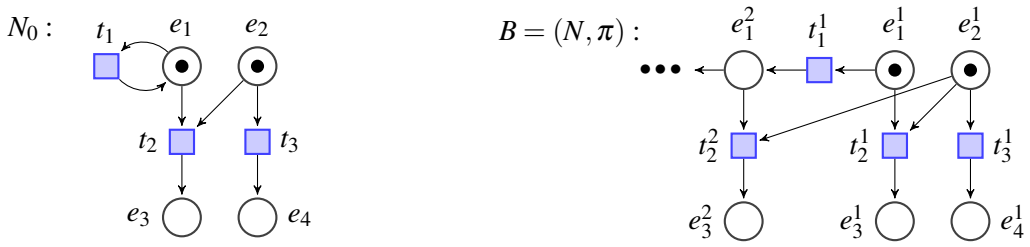

Figure 1: A Petri net N_0 on the left and a branching process $B = (N, \pi)$ of N_0 on the right. Places are shown as circles, transitions as boxes and the preset and postset relations as arrows. The initial marking $\{e_1, e_2\}$ is represented by the black dots, the *tokens*. The homomorphism π from N to N_0 is given as $\pi(e_j^i) = e_j$ and $\pi(t_j^i) = t_j$. The transitions t_1^1 and t_2^1 are in conflict, i.e., $t_1^1 \# t_2^1$, and also $t_2^2 \# t_2^1$, $t_3^1 \# t_2^1$, $t_2^2 \# t_3^1$.

In Fig. 1, the branching process B is the unfolding of the Petri net N_0 assuming that the dots to the left of the place e_1^2 indicate that the branching process continues infinitely in the same way.

We continue with the definition of a Petri game. A Petri game is played on a finite and safe Petri net. Tokens may transit from a system place to an environment place and vice versa.

Petri game. A *Petri game* on an underlying finite and safe Petri net N_0 is a tuple $G = (\mathscr{P}_0^S, \mathscr{P}_0^E, \mathscr{T}_0, pre_0, post_0, In_0, \mathscr{B})$, where the set $\mathscr{P}_0$ of places of N_0 is partitioned into disjoint sets of *system places* $\mathscr{P}_0^S$, and *environment places* $\mathscr{P}_0^E$, and where $\mathscr{B} \in 2^{\mathscr{P}_0}$ is the set of *bad markings*.

A winning strategy of a Petri game is a branching process of the underlying Petri net of the game. A strategy must satisfy 4 properties that are reasonable properties of implementations of processes. Beside a *safety* property, each process must act deterministically, *determinism*, and at least one process must enable a transition if possible, *deadlock avoiding*. Loosely speaking, the *justified refusal* property forces the system players to commit to transitions that are always allowed on their current place. Strategies for Petri games are obtained by cutting out parts of the unfolding.

Winning strategy A *winning strategy* σ of a Petri-game $G = (\mathscr{P}_0^S, \mathscr{P}_0^E, \mathscr{T}_0, pre_0, post_0, In_0, \mathscr{B})$ with underlying Petri-net N_0 is a branching process $\sigma = (N, \pi)$ of N_0 satisfying the following properties:

1. **Justified refusal**: Let $C \subseteq \mathscr{P}$ be a set of pairwise concurrent places and $t \in \mathscr{T}_0$ a transition with $\pi(C) = pre_0(t)$. If no $t' \in \mathscr{T}$ with $\pi(t') = t$ and $pre(t') = C$ exists, then there exists a place $p \in C$ with $\pi(p) \in \mathscr{P}_0^S$, such that $t \notin \pi(post(p))$.
2. **Safety**: For all reachable markings $M \in \mathscr{R}(N)$ it holds that $\pi(M) \notin \mathscr{B}$.
3. **Determinism**: For all $p \in \mathscr{P}$ with $\pi(p) \in \mathscr{P}_0^S$ and for all reachable markings M in N with $p \in M$ there exists at most one transition $t \in post(p)$, which is enabled in M.
4. **Deadlock avoiding**: For all reachable markings M in N there exists an enabled transition, if a transition is enabled in $\pi(M)$ in the underlying Petri-net N_0.

We refer to a token on a system place as a system player, and a token on an environment place as an environment player. The *justified refusal* property ensures that a system player allows all instances of an outgoing transition or no instance at all. The global *safety* property ensures that no bad markings are reachable. The *determinism* property ensures that for each system place at most one transition is enabled in every reachable marking. The *deadlock avoiding* property ensures that the system allows at least one transition in every reachable marking if an enabled transition exists in that marking, e.g. the system players cannot add deadlocks to the existing deadlocks in the Petri net by forbidding transitions.

The *unfolding* $unf(G)$ of a Petri game G is like the unfolding $unf(N_0) = (N, \pi)$ of the underlying Petri net N_0 of G, additionally keeping the distinction between system and environment: a place p in N is a system place if $\pi(p) \in \mathscr{P}_0^S$ and an environment place if $\pi(p) \in \mathscr{P}_0^E$. A winning strategy σ_G of G can be seen as a subprocess of $unf(G)$.

Fig. 2 shows a Petri game with 4 players modelling the control of a room with two doors that must not be opened at the same time so that the two places of the bad marking $\{O^1, O^2\}$ are not reached simultaneously. After receiving a request to open door via transition r^1 the first system player on place S^{12} can decide to proceed with communicating (transition t) or without communicating (transition n^1) to the second system player, then on place S^{22}. On place S^{13} the system player chooses between opening the door (o^1) or denying the request (d^1) for the environment player waiting on W^1. From place S^{14} the system player can close the door it has opened before (c^1). The second system player has the same options after receiving a request via r^2. After that, if both players open their doors the bad marking $\{O^1, O^2\}$ is reached. At least one system player has to deny the request to win the game.

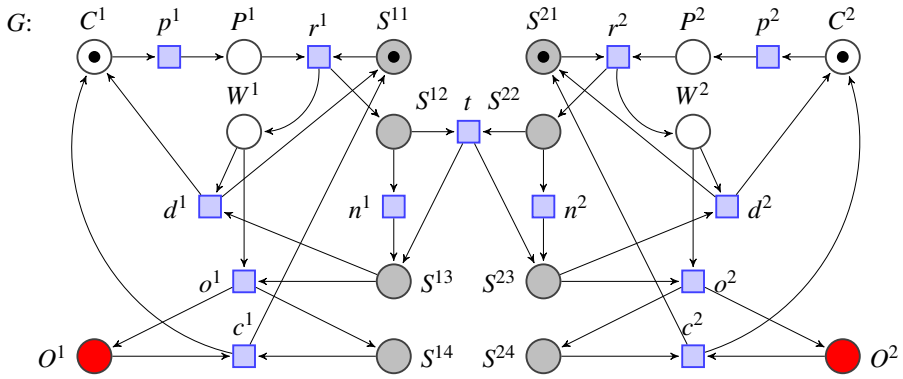

Figure 2: A Petri game G: the grey places belong to the system players and the white places to the environment players. There are two doors and two system players taking requests to open the door. After a request the system players decide whether to communicate and open their door afterwards. The bad marking $\{O^1, O^2\}$ is reached if the system players decide to open both doors simultaneously.

Fig. 3 shows an initial part of the unfolding of the Petri game in Fig. 2. The parts that are not greyed out are the initial parts of a winning strategy. Here, the system player agree on communicating via t_1. The first player decides to open its door via transition o_2^1 while the other door remains closed via transition d_2^2. After the transition c_2^1 the first door is closed again. The following is not shown in Fig. 3 anymore: after both players have received another opening request via r_2^1 and r_2^2, respectively, the winning strategy could continue opening door 2 and keeping door 1 closed since the different causal histories allow different decisions.

According to the definition of a winning strategy it is also possible that a winning strategy denies all requests to open a door. The example is chosen with foresight for the content in Section 3.

3 Decidability Results

In this section, we show that the synthesis problem for Petri games with up to 4 players under a global safety condition is decidable in non-deterministic exponential time (NEXP), and NP-complete in the 2-player case. In [11], a related result is shown that the synthesis problem for 4-process control games

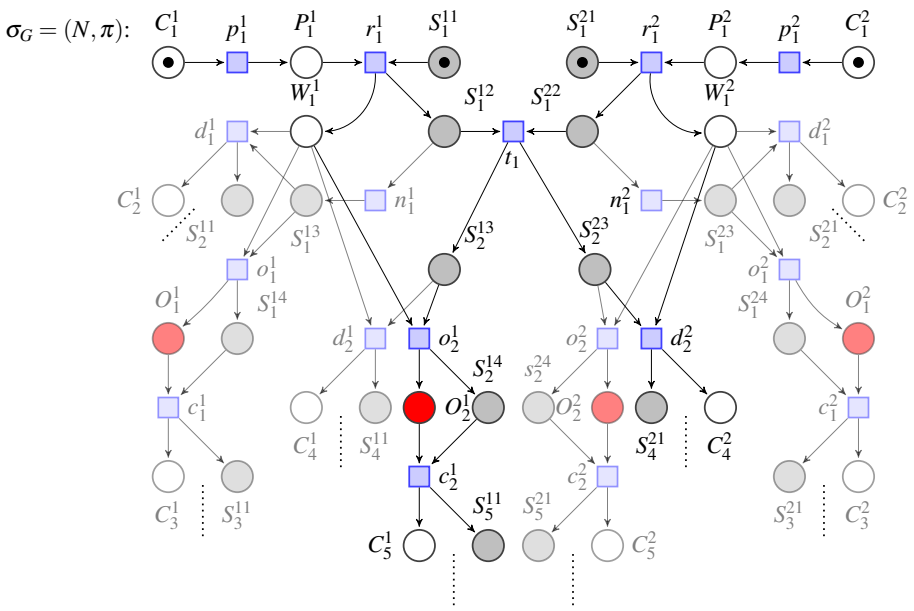

Figure 3: An initial part of the unfolding of the Petri game G in Fig. 2. The nodes that are not greyed out form an initial part of a winning strategy σ_G where door 1 gets opened and door 2 remains closed after the system players have communicated. The homomorphism π is defined analogously to that in Fig. 1.

with a local termination condition is decidable. The translation [1] of this result to Petri games gives a decidability result for Petri games with 4 players with a local termination condition, without process generation and deletion. There are no complexity bounds for the 4-player case given in [11], and the translation to Petri games already generates an exponential blow up in the number of nodes [1].

A Petri game G is called a *K-player Petri game*, $K \in \mathbb{N}$, if and only if for all reachable markings $M \in \mathscr{R}(G)$ it holds that $|M| \leq K$. Two-player Petri games can be seen as a natural generalisation of infinite games on graphs with a safety winning condition. NP-hardness is established by a reduction of the 3-SAT problem to 2-player Petri games.

Lemma 3.1. *There exists a polynomial-time reduction of the 3-SAT problem to the synthesis problem of two-player Petri games.*

Proof. Let $F = (x_1^1 \vee x_2^1 \vee x_3^1) \wedge \ldots \wedge (x_1^n \vee x_2^n \vee x_3^n)$ be an instance of the 3-SAT problem in conjunctive normal form where all clauses consist of exactly three literals and where x_j^i, $i = 1, \ldots, n$ and $j = 1, 2, 3$, is a positive or negative (with overline) literal from the set $\{x_1, \bar{x}_1, \ldots x_m, \bar{x}_m\}$. Fig. 4 shows the Petri game of F. If F is satisfiable, the top system player chooses the transitions according to the boolean assignment that satisfies F, for example $\bar{x}_1$ if the truth value of x_1 is false under the boolean assignment. The bottom system player chooses the literal that is true under the boolean assignment in each clause, for example if $x_1^1 = \bar{x}_1$, she may choose x_1^1 since the top system player chooses it. Both system players do not know how when the transition of the other player are fired such that it has to be correct for all possible orders of execution, for example the top player could have chosen the truth value for x_n already before the bottom player chose the literal for the first clause.

Conversely, the top player's choices yield a boolean assignment that satisfies F if the Petri game shown has a winning strategy since the bottom player chooses one literal of each clause that must be true

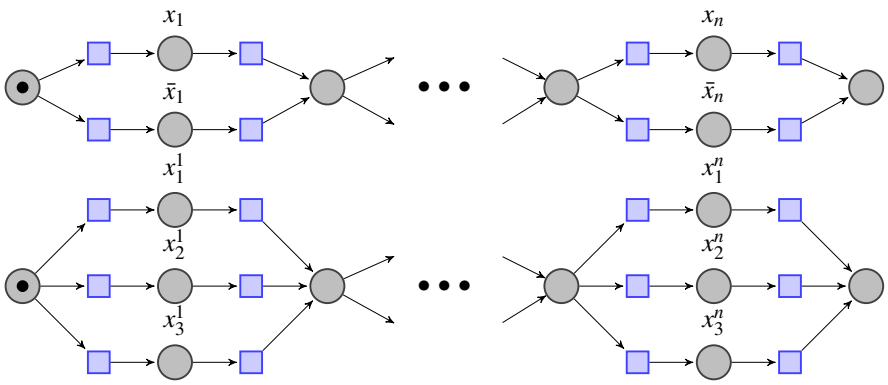

Figure 4: 3-SAT problem F as a Petri game. The top system player chooses sequentially the truth value for $x_1, \ldots x_n$. The bottom system player has to choose a literal for each clause according to the truth values chosen by the top player. The bad markings are $\{\{x_k, x_j^i\} \mid x_j^i = \bar{x}_k\} \cup \{\{\bar{x}_k, x_j^i\} \mid x_j^i = x_k\}$. So every time the bottom player chooses a literal that is not chosen by the top system player a bad marking is reached, e.g. the top player chooses x_1 and the bottom player x_3^1 where $x_3^1 = \bar{x}_1$.

under the boolean assignment. □

Note that the 2-player case is NP-hard even without an environment player. The matching upper bound is established later, along with the complexity of the 4-player case.

The idea of solving 4-player Petri games is to find game states where a winning strategy can be repeated and still win. A few definitions are necessary to formalise repeating a part of a winning strategy. The following definitions about branching processes are equivalent to those in [5].

Cut. The reachable markings in a branching process B are called *cuts*. A cut is a maximal set of pairwise concurrent places [7].

Future of a cut. We define the branching process $Fut(B,C)$ of a cut $C \subseteq \mathscr{P}_B$ as follows: $\mathscr{P}_{Fut(B,C)} \cup \mathscr{T}_{Fut(B,C)} = \{x \in \mathscr{P}_B \cup \mathscr{T}_B \mid \forall p \in C : p \leq x \vee p || x\}$, the mappings *preset* and *post* are the mappings pre_B and $post_B$ restricted to $\mathscr{T}_{Fut(B,C)}$, and $In_{Fut(B,C)} = C$. Generally, the *future* $Fut(B,C)$ is not a branching process of the underlying net N_0 of B (i.e. $\pi_B : \mathscr{T}_B \cup \mathscr{P}_B \to \mathscr{T}_0 \cup \mathscr{P}_0$) but it is a branching process of the net $(\mathscr{P}_0, \mathscr{T}_0, pre_0, post_0, \pi_B(C))$, i.e. $Fut(B,C)$ is a branching process of N_0 if $\pi_B(C) = In_0$. (This definition is equal to the definition of $\Uparrow$*Configuration* in [5].)

Informally speaking, the definition of the future of a cut coincides with the intuition that it is the branching process that follows after that cut.

Last known cut and last known marking. The *last known cut* $lkc(t)$ of a transition t of a branching process B is defined as $lkc(t) = \{p \in \mathscr{P}_B \mid p \not< t \wedge \forall t' \in pre_B(p) : t' \leq t\}$. Informally speaking, this cut is reached by firing all transitions that are causal predecessors of t. The *last known marking* $lkm(t)$ is defined as $\pi_B(lkc(t))$, which is the marking reached in the underlying Petri net. (The *lkc* is the cut of a *local configuration* [5])

A *cut* and *glue* operation formalises the process of copying the future of one cut to another cut of a strategy. Later, the actual requirements for when to copy are defined. In the following, B and B' are branching processes of (possibly different) nets.

Cut. $B - B' = (\mathscr{P}_B \setminus (\mathscr{P}_{B'} \setminus In_{B'}), \mathscr{T}_B \setminus \mathscr{T}_{B'}, pre_B \restriction_{\mathscr{T}_{B-B'}}, post_B \restriction_{\mathscr{T}_{B-B'}}, In_B)$

If B' is the future of a cut of B, the cut operation removes B' from B leaving only the initial marking of B' in B.

Glue. $B+B' = (\mathscr{P}_B \cup \mathscr{P}_{B'}, \mathscr{T}_B \cup \mathscr{T}_{B'}, pre_{B+B'}, post_{B+B'}, In_B)$, where $pre_{B+B'}(t) = \begin{cases} pre_B(t) & t \in \mathscr{T}_B \\ pre_{B'}(t) & t \in \mathscr{T}_{B'} \end{cases}$, and analogously $post_{B+B'}$.

If the initial marking of B' is a cut in B, B' gets glued to that cut. Generally, $B-B'$ and $B+B'$ are not branching processes. However, if there are two cuts $C_1, C_2 \subseteq \mathscr{P}_B$ with $\pi_B(C_1) = \pi_B(C_2)$ cutting out the future of C_2 and glueing an isomorphic copy (this is just a necessary renaming) of the future of C_1 to C_2 yields a branching process.

Cut and glued branching process. Let $Fut(B,C_1)'$ be an isomorphic copy of $Fut(B,C_1)$, $Fut(B,C_1)' \cong Fut(B,C_1)$, such that $In_{Fut(B,C_1)'} = C_2$ and $(\mathscr{P}_B \cup \mathscr{T}_B) \cap (\mathscr{P}_{Fut(B,C_1)'} \cup \mathscr{T}_{Fut(B,C_1)'}) = C_2$. The cut and glued branching process $B_{C_1 \to C_2}$ is the branching process $B_{C_1 \to C_2} = B - Fut(B,C_2) + Fut(B,C_1)'$. This definition is unique up to isomorphism.

Definition 3.1 (Imitable cuts)**.** Let σ be a winning strategy of a Petri game G. A cut C_2 is imitable by a cut C_1 if $\sigma_{C_1 \to C_2}$ is a winning strategy.

Now we define the actual cuts that are imitable. The first kind of these cuts are cuts where a subset of players take transitions without communicating to the remaining players until all players of this subset synchronise at a joint transition for the second time. In the following, we fix σ as a winning strategy of a Petri game G.

Definition 3.2 (Partial repetition cuts)**.** Let $t_1, t_2 \in \mathscr{T}_\sigma$. The cuts $lkc(t_1)$ and $lkc(t_2)$ are *partial repetition cuts*, denoted $prc(t_1,t_2)$, if $lkm(t_1) = lkm(t_2) \wedge lkc(t_1) \setminus post(t_1) = lkc(t_2) \setminus post(t_2)$. The partial repetitions cuts $prc(t_1,t_2)$ are called a *loop*, denoted $prc_{loop}(t_1,t_2)$ if $t_1 < t_2$.

Lemma 3.2 (Partial repetition cuts are imitable)**.** *For transitions $t_1, t_2 \in \mathscr{T}_\sigma$, $prc(t_1,t_2)$ implies that $lkc(t_2)$ is imitable by $lkc(t_1)$ and vice versa.*

Proof by contradiction. We show that $\sigma_{lkc(t_1) \to lkc(t_2)} = \sigma - Fut(\sigma, lkc(t_2)) + Fut(B, lkc(t_1))'$ is a winning strategy. Suppose that there exists a cut $C \subseteq \mathscr{P}_{\sigma_{lkc(t_1) \to lkc(t_2)}}$ such that one of the properties of the winning strategy is violated.

We distinguish two cases: The first case is that there exists a place $p \in C$ such that $t_2 \leq p$. Then, $C \subseteq \mathscr{P}_{Fut(\sigma_{lkc(t_1) \to lkc(t_2)}, lkc(t_2))}$, holds. Since $Fut(\sigma_{lkc(t_1) \to lkc(t_2)}, lkc(t_2)) = Fut(\sigma, lkc(t_1))' \cong Fut(\sigma, lkc(t_1))$ it follows directly that σ is not a winning strategy if $\sigma_{lkc(t_1) \to lkc(t_2)}$ is not winning.

The second case is that there exists no place $p \in C$ such that $t_2 \leq p$, i.e. $\forall p \in C : p \leq t_2 \vee p \# t_2 \vee p || t_2$ holds. Since the last known cuts of t_1 and t_2 are equal except for $post(t_1)$ and $post(t_2)$ the branching processes $Fut(\sigma, lkc(t_2))$ and $Fut(\sigma, lkc(t_1))$ are isomorphic up to the nodes that are causal successors of t_1 and t_2 respectively. This means that all transitions and places that are not causal successors of t_2 are only renamed by constructing $\sigma_{lkc(t_1) \to lkc(t_2)}$. Since there is no place in C that is a causal successor of t_2 there is no transition enabled in C that could have been added or removed. It follows that σ is not a winning strategy, if $\sigma_{lkc(t_1) \to lkc(t_2)}$ is not a winning strategy. □

In Fig. 5 is an example of partial repetition cuts. Note that not all of those partial repetition cuts are loops as the players can reach the same places in the underlying Petri net by taking different transitions. Partial repetition cuts are defined regardless of the number of players in a Petri game, i.e, these cuts are imitable in any Petri game.

Maximally repeated strategy. Since the nodes of a branching process are countable we can construct by induction over the partial repetition cuts of a winning strategy σ a winning strategy σ^{pr} where $prc(t_1,t_2)$ implies that $Fut(\sigma^{pr}, t_1) \cong Fut(\sigma^{pr}, t_2)$. σ^{pr} denotes a *maximally repeated strategy* of σ.

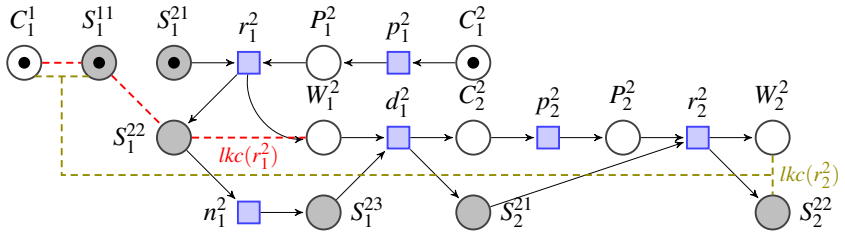

Figure 5: Example of partial repetition cuts. Assume that this is a part of a winning strategy where the second system player denies (d_1^2) the first request (r_1^2) to open the door without communicating with the other system player via transition t. Here, the last known cuts of r_1^2 and r_2^2 are $lkc(r_1^2) = \{C_1^1, S_1^{11}, S_1^{22}, W_1^2\}$ and $lkc(r_2^2) = \{C_1^1, S_1^{11}, S_2^{22}, W_2^2\}$, so $lkm(r_1^2) = lkm(r_2^2)$. Thus, a partial repetition cut (that is a loop) occurs $prc_{loop}(r_1^2, r_2^2)$.

The partial repetition cuts have a useful implication for the case with up to 4 players: if two players do not take any joint transition with one of the two remaining players they will get to a partial repetition cut eventually. The idea of the following definition of a synchronisation segment is based on this implication. Informally speaking, a synchronisation segment describes the part starting from the last known cut of a transition t until all other players are causal successors of this transition or the strategy can be repeated due to the partial repetition cuts. Here, the idea of when to repeat a strategy is that if all players play identically starting from the last known cut of transition t until they get to know of transition t the strategy can be repeated after transition t.

Definition 3.3 (Synchronisation segment)**.** The *synchronisation segment* $Seg(t)$ of a transition $t \in \mathscr{T}_B$ of a branching process B is defined as a smallest branching process (with respect to $\leq$) such that
(1.) $Seg(t) \leq Fut(B, lkc(t))$ and
(2.) $\forall t' \in \mathscr{T}_{Fut(B,lkc(t))} : t' \notin \mathscr{T}_{Seg(t)} \Rightarrow$ (a) $(\forall p \in pre_B(t') : t \leq p) \vee$
(b) $(\exists t_1, t_2, t_3 \in \mathscr{T}_{Seg(t)} : prc_{loop}(t_1, t_2) \wedge t_3 \leq t_2 \wedge prc(t', t_3))$

For a transition $t' \in Fut(B, lkc(t))$ that is not in $\mathscr{T}_{Seg(t)}$, all places in its preset are causal successors of t (2.a) or there is a transition t_3 with $prc(t', t_3)$ within a loop (2.b). So, the parts of a winning strategy that get repeated due to partial repetition cuts are included in the synchronisation segment. An example of a synchronisation segment is shown in Fig. 6 and Fig. 7. *Synchronisation equivalent cuts* are those cuts where the synchronisation segments are isomorphic.

Definition 3.4 (Synchronisation equivalent cuts)**.** Let σ^{pr} be a maximally repeated winning strategy of a Petri game G and $t_1, t_2 \in \mathscr{T}_{\sigma^{pr}}$. The cuts $lkc(t_1)$ and $lkc(t_2)$ are *synchronisation equivalent cuts*, denoted $sqc(t_1, t_2)$, if $Seg(t_1) \cong Seg(t_2)$. $sqc(t_1, t_2)$ is called an *s-loop*, denoted sqc_{loop}, if $t_1 < t_2$.

Now, we show that a winning strategy can be repeated after synchronisation equivalent cuts.

Lemma 3.3 (Synchronisation equivalent cuts are imitable)**.** *For transitions $t_1, t_2 \in \mathscr{T}_{\sigma^{pr}}$, $sqc(t_1, t_2)$ implies that $lkc(t_2)$ is imitable by $lkc(t_1)$ and vice versa.*

Proof. We show that $\sigma^{pr}_{lkc(t_1) \to lkc(t_2)} = \sigma^{pr} - Fut(\sigma, lkc(t_2)) + Fut(\sigma, lkc(t_1))'$ is a winning strategy. Since $Seg(t_1) \cong Seg(t_2)$ the construction is only a renaming for the nodes in the synchronisation segment and for those that get repeated due to the partial repetition cuts. Thus, only nodes that are causal successors of t_2 may be removed or added. Now, the proof works in the same way as the proof of Lemma 3.2, that partial repetition cuts are imitable. □

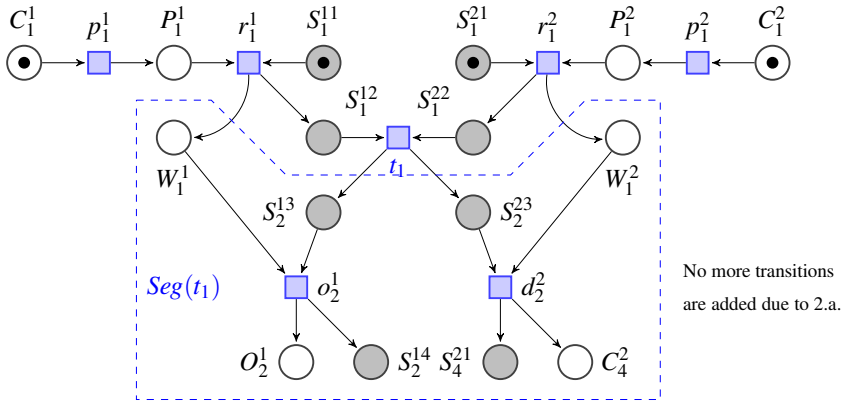

Figure 6: A subprocess of σ_G of Fig. 3 is shown. The nodes inside the dashed, blue box are the nodes of the synchronisation segment $Seg(t_1)$. The initial marking is $In_{Seg(t_1)} = \{W_1^1, S_2^{13}, S_2^{23}, W_1^2\}$. This segment ends after the transitions o_2^1 and d_2^2 since $t_1 < O_2^1, S_2^{14}, S_4^{21}, C_4^2$. So, if σ_G repeats to open the first door and keeping the second door closed after taking the transition t the winning strategy could repeat itself due to synchronisation equivalent cuts.

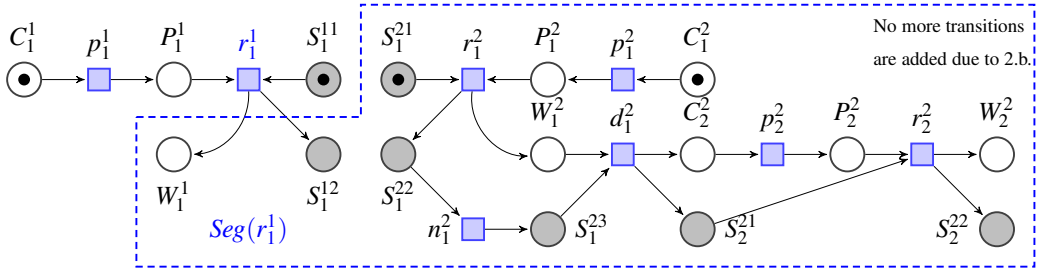

Figure 7: A branching process of the Petri game G of Fig. 2 is shown. The nodes inside the dashed, blue box are the nodes of the synchronisation segment $Seg(r_1^1)$.

We show that 4-player Petri games can be solved in NEXP with the help of Lemma 3.2 and Lemma 3.3. A winning prefix of sufficient size is guessed from which a winning strategy can be constructed. A branching process B is a *winning prefix* if there exists a winning strategy σ with $B \leq \sigma$.

Theorem 3.4. *The synthesis problem of 2-player Petri games is NP-complete and in NEXP for 3- and 4-player Petri games.*

Proof. Structure of this proof: from the 2-player case over the 3-player case to the 4-player case the size of a winning prefix is determined to guarantee that a winning strategy can be constructed by repeating the futures of imitable cuts. A prefix of appropriate size is guessed and it is checked if it is winning. The prefix does not contain any causal successors of transitions t_2 if $\exists t_1 : prc_{loop}(t_1,t_2) \vee sqc_{loop}(t_1,t_2)$. Also, it does not contain causal successors of transitions t_4, if $\exists t_1,t_2,t_3 : t_3 \leq t_2 \wedge ((prc(t_3,t_4) \wedge prc_{loop}(t_1,t_2)) \vee (sqc(t_2,t_3) \wedge sqc_{loop}(t_1,t_2)))$. Let T be the number of transitions of the Petri game.

If a player does not take any joint transition she eventually repeats a place she lays on, i.e. she reaches a partial repetition cut. This occurs after at most T transitions. Otherwise, the two players take a joint transition after at most T transitions. Since there are at most T different joint transitions, a winning prefix from which a winning strategy can be constructed is at most of size $\mathcal{O}(T^2)$. To check if the guessed prefix

is winning, we have to check all reachable markings for the winning properties. In a safe Petri net with T^2 transitions and at most two tokens there are at most T^4 reachable markings. Thus, guessing a winning prefix and checking if it is winning takes time in $\mathcal{O}(|\mathcal{T}|^4)$, which shows together with Lemma 3.1 that the synthesis problem for two-player Petri games is NP-complete.

In the 3-player case, it holds for the same reasons as in the 2-player case that each pair of two players reaches partial repetition cuts after at most T^2 transitions without taking a joint transition with the remaining third player. The remaining third player can take up to T transitions herself before reaching a partial repetition cut or taking a joint transition. Therefore, a synchronisation segment has at most T^2+T transitions resulting in at most $2^{(T^2+T)^2}$ non-isomorphic synchronisation segments. Therefore, a synchronisation equivalent cut is always reached after at most $2^{(T^2+T)^2}$ transitions such that the winning prefix can be extended step by step by reaching synchronisation equivalent cuts again and again. Thus, the size of a winning prefix is in $\mathcal{O}(2^{T^4})$. To check if the prefix is winning takes polynomial time in its size and it follows that the 3-player case is decidable in NEXP.

In a 4-player Petri game, it also holds that each pair of two players reaches partial repetition cuts after at most T^2 transitions without taking a joint transition with one of the two remaining players. So, there are two pairs of players that can take T^2 transitions until they reach partial repetition cuts or one player of each pair communicate with each other. This leaves the other two players until they reach partial repetition cuts or take a joint transition again after T^2 transitions. Thus, there are at most 2^{3T^2} non-isomorphic synchronisation segments. Therefore, the size of a winning prefix of a 4-player Petri game that needs to be checked if it is winning is in $\mathcal{O}(2^{6T^2})$. So the bound is in NEXP. Note that if more than 2 players take a joint transition the size of the synchronisation segments decreases. □

This construction does not work for the 5-player case. The problem that occurs is tricky to see. Assume there is a group of 3 players and a group of 2 players that take joint transitions repeatedly within their group, only two players participating at a time. The group of 2 players get to partial repetition cuts or one of them takes a joint transition with one player of the other group, while the group of 3 players might not get to partial repetition cuts. After two players, one from each group, have taken a joint transition the problem occurs: the remaining 3 players that did not participate in that joint transition might form a new group of 3 players that do not reach partial repetition cuts. This cannot occur in the 4-player case since there is only a pair of players left.

4 Undecidability Result

In this section, a tiling problem is reduced to the synthesis problem of a 6-player Petri game with a global safety condition. The tiling problem used here is the ω bipartite colouring problem (ω-BCP). The undecidable ω Post correspondence problem (ω-PCP) is reducible to the ω-BCP. Later, the undecidability result can be simplified to local safety as a winning condition. The reduction is similar to the work in [12], where the (normal) Post correspondence problem is reduced to a colouring problem followed by a reduction to the synthesis problem of 6-process control games with local termination as a winning condition.

Employing the translation of control games into Petri games in [1], the 6-process control games yield 6-player Petri games (with exponentially many additional nodes) for local termination as a winning condition. In the obtained Petri game, all players alternate in their roles as environment and system players. Instead, we present a direct construction of 6-player Petri games with at most 3 simultaneous system players. Later, we show that the number of system players can be even further reduced to 2.

Definition 4.1 (ω bipartite colourings)**.** Let C be a finite set of colours. An *ω bipartite colouring* is a mapping $f : \mathbb{N}^2 \mapsto C$. The initial colour is $f(1,1)$. We define three subsets of C^2 called the patterns induced by f:

- the *diagonal patterns* of f are all pairs $\{(f(x,y), f(x+1,y+1)) \mid (x,y) \in \mathbb{N}^2)\}$
- the *horizontal patterns* of f are all pairs $\{(f(x,y), f(x+1,y)) \mid (x,y) \in \mathbb{N}^2)\}$
- the *vertical patterns* of f are all pairs $\{(f(x,y), f(x,y+1)) \mid (x,y) \in \mathbb{N}^2)\}$

We define a *colouring constraint* as a 4-tuple (C_i, DP, HP, VP), where C_i is the set of *initial* colours, DP a set of diagonal patterns, VP a set of vertical patterns and HP a set of horizontal patterns. DP, VP and HP are called *forbidden patterns*. A colouring f *satisfies* a colouring constraint if its initial colour is in C_i and no diagonal pattern of f is in DP, no vertical pattern of f is in VP and no horizontal pattern of f is in HP.

ω-BCP. Given a finite set of colours C and colouring constraints (C_i, DP, VP, HP), decide whether there exists an ω bipartite colouring that satisfies the colouring constraints. This problem is a variation of the standard tiling problem and it is also undecidable. In the following we define a Petri game for which a winning strategy exists if and only if a given ω-BCP has a solution. In this Petri game shown in Fig. 8, there are two identical parts, a top part and a bottom part, each consisting of 3 players. The idea is that the number of *rounds* played in the lower and upper parts refer to the x and y coordinates of a tile, respectively, so that the system players choose a colour for each tile. Each colour choice requires one player from each part of the Petri game, so there can be a maximum of 3 colour choices in one run of the Petri game. If the colours chosen refer to a forbidden pattern a bad marking is reached. As the bad markings have to be defined on the Petri net and not on the unfolding, we define when two system players are in the same round or when a player is one round ahead or one round behind to be able to check for forbidden patterns.

Compared rounds of two players. In the Petri net N_{CP} of Fig. 8, we compare the rounds of two players a and b, both from the same part. We say $a, b \in \{0,1,2\}$ are in the same round if and only if for their places e^X_{aj} and e^X_{bk}, $X \in \{T,B\}$, $j,k \in \{0,1,2\}$ or $j,k \in \{3,4\}$ or $j,k \in \{5,2\}$ holds. We say that a is one round ahead of b (or b is one round behind of a) if and only if ($j \in \{3\}$ and $k \in \{2\}$) or ($j \in \{5\}$ and $k \in \{4\}$) . Other combinations of indices are not possible.

Definition 4.2 (Petri game of an ω Bipartite colouring Problem)**.** Let CP be an ω-BCP with colours C and constraints C_i, DP, VP and HP. The components of the Petri net of the Petri game G_{CP} are the components of N_{CP} from Fig. 8. The bad markings of the Petri game G_{CP} are defined as follows:

$\mathscr{B} = \mathscr{B}_{Same} \cup \mathscr{B}_{DP} \cup \mathscr{B}_{HP} \cup \mathscr{B}_{VP} \cup \mathscr{B}_{init}$, where

$\mathscr{B}_{Same} = \{\{e_{a^T j - a^B j'}, e_{b^T k - b^B k'}, (c_u, a), (c_v, b)\} \mid$
a^T and b^T are in the same round and a^B and b^B are in same the round $, c_u \neq c_v\}$,

$\mathscr{B}_{DP} = \{\{e_{a^T j - a^B j'}, e_{b^T k - b^B k'}, (c_u, a), (c_v, b) \mid$
a^T and a^B are one round ahead of b^T and b^B respectively $, (c_v, c_u) \in DP\}$,

$\mathscr{B}_{VP} = \{\{\{e_{a^T j - a^B j'}, e_{b^T k - b^B k'}, (c_u, a), (c_v, b)\} \mid$
a^T is a round ahead of b^T and a^B and b^B are in the same round $, (c_v, c_u) \in VP\}\}$,

$\mathscr{B}_{HP} = \{\{\{e_{a^T j - a^B j'}, e_{b^T k - b^B k'}, (c_u, a), (c_v, b)\} \mid$
a^B is a round ahead of b^B and a^T and b^B are in the same round $, (c_v, c_u) \in HP\}\}$,

$\mathscr{B}_{init} = \{\{e_{a^T j - a^B j'}, (c_u, a)\} \mid j, j' \in \{0,1\}, c_u \notin C_i\}$.

In addition to the colouring constraints, the bad markings contain the set $\mathscr{B}_{Same}$ to ensure that system players must choose the same colour for two checks whenever the checks occur in the same round for

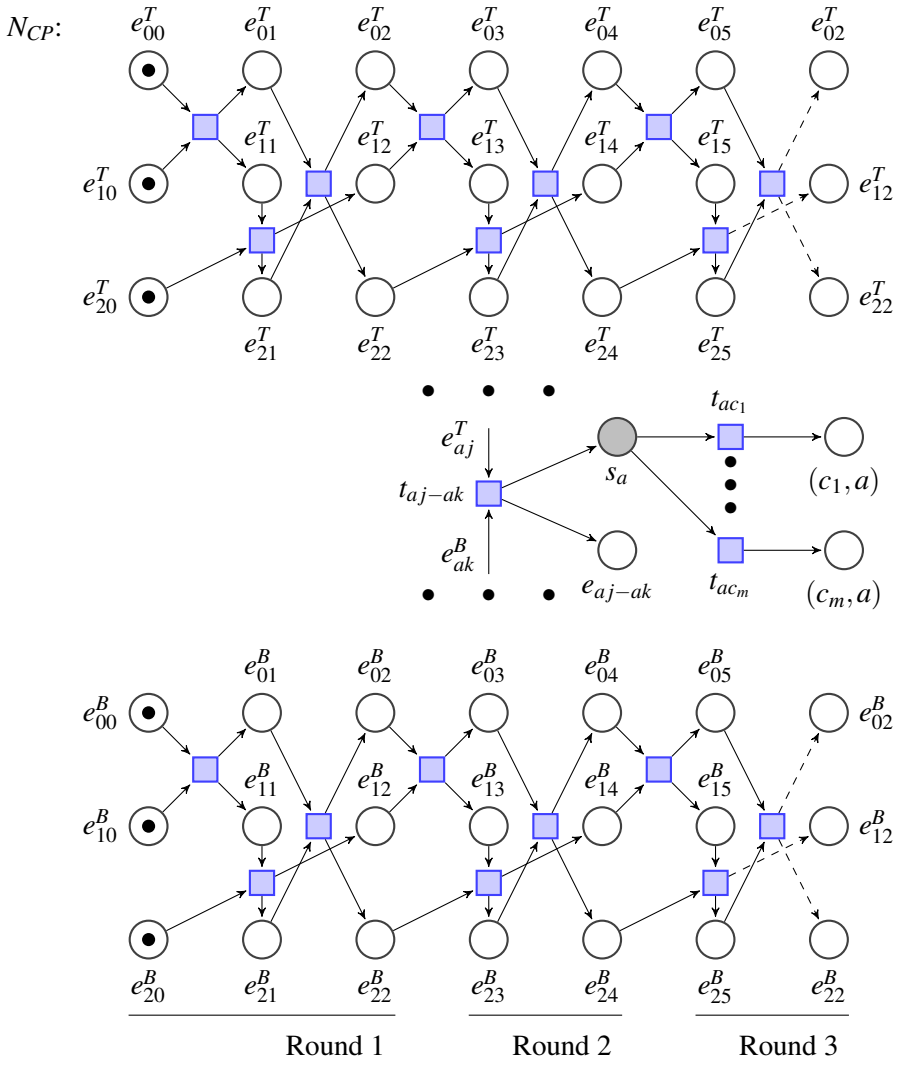

Figure 8: Petri net N_{CP} of the Petri game G_{CP} of an ω-BCP. We have two structurally identical parts consisting of 3 environment players, a top part and a bottom part. Two of the players synchronise at each transition in these parts. After firing 3 such transitions, each player has synchronised with the other two players in its part. We define such a block of 3 transitions as a *round*. Each part has 3 rounds. After the third round the players return to the places they were in before the second round. This means that we have an initial round and then the players alternate between the second and third rounds. As the game progresses, the nodes have exact information in the unfolding about how many rounds they have played. In the middle part, the environment players can take a *check* transition t_{aj-ak}, where $a \in \{0,1,2\}$ and $j,k \in \{0,\ldots,5\}$ and $pre(t_{aj-ak}) = \{e^T_{aj}, e^B_{ak}\}$. After that, the system player on place s_a has to choose a colour. Based on the number of rounds x and y played in the lower part and upper part, respectively, the system player has to determine the colour $f(x,y)$ of the given ω-BCP.

both players. To define a colouring from a strategy of the Petri game, we define the number of rounds played in the unfolding.

Number of rounds. Let $unf(G_{CP})$ be the unfolding of G_{CP} of Fig. 8. The *number of rounds* played in part X, $X \in \{T,B\}$ in a place $p \in \mathscr{P}_{unf(G_{CP})}$ is defined as $r^X(p) = max(\lceil |\{t \in \mathscr{T}_{unf_{CP}} \mid t \leq p \text{ and } t \text{ is a transition in part } X\}|/3 \rceil, 1)$

This means that the number of rounds is incremented after a player has taken two transitions, starting from round 1. The divisor is 3 because the players also get the knowledge of the synchronisation of the other two players in their part. The number of rounds is not increased after a check transition.

Theorem 4.1 (Characterization of ω-BCP as a Petri game)**.** *The ω-BCP is reducible to the synthesis Problem of 6-player Petri games.*

Proof. Let CP be a colouring problem and G_{CP} be the Petri game from Def. 4.2. We show that there exists a solution for CP if and only if there exists a winning strategy for G_{CP}.

We start by assuming that there is a solution $f : \mathbb{N} \times \mathbb{N} \mapsto C$ for the colouring problem CP. After the environment player has chosen a check transition the system player must choose a colour in place s_a, $a \in \{0,1,2\}$. Since the system player knows the entire causal history, she knows the number of rounds played in the top and the bottom part of the Petri game. Let $r^T(s_a) = y$ denote the number of rounds in the top part and $r^B(s_a) = x$ the number of rounds in the bottom part. Then, the winning strategy for the system players is to choose the colour $f(x,y)$. This way no bad marking is reachable. The bad markings in $\mathscr{B}_{Same}$ are avoided because $f(x,y)$ is unique. The bad markings in $\mathscr{B}_{DP}, \mathscr{B}_{VP}$ and $\mathscr{B}_{HP}$ are avoided because f avoids all forbidden patterns. Finally, the bad markings $\mathscr{B}_{init}$ are avoided because $f(1,1)$ is an initial colour.

Now, we assume that there is a winning strategy for the Petri game G_{CP}. We define the colouring $f : \mathbb{N} \times \mathbb{N} \mapsto C$ derived from the winning strategy as follows: $f(x,y) = c$ if there exists a system place $s_a \in \mathscr{P}_{unf(N_{CP})}$ with $r^B(s_a) = x$ and $r^T(s_a) = y$ and $t_{ac} \in post(s_a)$.

We show that for every colour that a system player has to choose, there are sequences of checks such that all colouring constraints are met. In these sequences of checks, any two consecutive checks can be carried out concurrently while the system players do not know whether the other check is the previous check or the next check, so the winning strategy must play correctly for both checks. Informally speaking, this leads to infinite sequences of dependencies as a check depends on the choice of the colour of the previous check and again that check depends on its previous check and so on.

In Fig. 9, it is crucial to see that there are concurrent places between different rounds such that we can always find at least one suitable check for all diagonal, vertical and horizontal patterns. To check the horizontal pattern of $(f(1,1), f(2,1))$ for example, the checks t_{00-03} and t_{20-22} are fired. The places in the upper part are e^T_{00} and e^T_{20} respectively, which are both in the first round. The places in the lower part are e^B_{03} and e^B_{22} respectively, where e^B_{03} is in the second round and e^B_{22} in the first round. These two checks can be performed concurrently since e^T_{00} and e^T_{20} are concurrent, as are e^B_{22} and e^B_{03}. For the initial colour we have the extra initial round which does not get repeated.

The colouring f is well-defined as the set of bad markings $\mathscr{B}_{Same}$ ensures that every two colours chosen when both top rounds and both bottom rounds are the same the colours chosen also must be the same in a winning strategy. This can be seen with the help of Fig. 9, too. There, we can find a sequence of checks throughout one round such that the colour chosen at the beginning of that round (e.g. e^B_{03}) is still the same colour as chosen at the end of that round (e.g e^B_{04}) (assuming that the top round does not change too). □

It is easy to see that the Petri game presented here performs at most 3 checks in every possible

	e^X_{00}	e^X_{10}	e^X_{20}	e^X_{01}	e^X_{11}	e^X_{21}	e^X_{02}	e^X_{12}	e^X_{22}	e^X_{03}	e^X_{13}	e^X_{23}	e^X_{04}	e^X_{14}	e^X_{24}
e^X_{00}		||	||												
e^X_{10}	||		||												
e^X_{20}	||	||		||	||										
e^X_{01}			||		||	||		||							
e^X_{11}			||	||												
e^X_{21}				||				||								
e^X_{02}								||	||							
e^X_{12}				||		||	||		||						
e^X_{22}							||	||		||	||					
e^X_{03}									||		||	||		||	
e^X_{13}									||	||						
e^X_{23}										||				||	
e^X_{04}														||	||
e^X_{14}										||		||	||		
e^X_{24}													||	||	

Figure 9: Table of concurrent places of the first two rounds in the unfolding of the Petri game G_{CP}. Empty cells denote that the places are causally related. Filled cells denote that the places are concurrent. The pattern continues identically. With the help of this table we can see the sequences of checks to ensure that no bad markings are reached in the Petri game. For example, to check that the initial colour chosen after the transition t_{00-00} and after t_{22-00} is the same, we can perform the following sequence of checks of the indices of the places in the upper part: $00-20-01-12-22$. The indices of the place in the lower part remain 00 here. To carry out two consecutive checks in the same play they need to be concurrent. We can check this in the table by going in a row from 00 to 20, then in the column from 20 to 01, then again checking in the row and so on. Note that the place of the bottom player can also change in these sequences.

sequence of transitions. Therefore this Petri game has at most 3 system players. For the undecidability result it is sufficient to have 2 concurrent checks. Therefore, the number of system players can be further reduced to 2 by adapting the Petri game. This could be done by adding two environment players that are consumed performing a check. Furthermore, by adding a transition for each bad marking to a bad place, the set of bad markings can be simplified to a set of bad places, the local safety condition.

5 Conclusions

Our contribution to the synthesis problem of distributed systems is that this problem is undecidable for 6-player Petri games under a local safety condition and that two system processes are sufficient to obtain undecidability. This adds neatly to the result in [7], where Petri games with one system player are solved in exponential time. Furthermore, the synthesis problem for 2-player Petri games belongs to the class of NP-complete problems, and we provide a non-deterministic exponential upper bound for this problem for Petri games with a variable number of players up to 4 under a global safety condition. This is a new class of Petri games for which the synthesis problem is decidable. The case of 5 players remains open.

References

[1] Raven Beutner, Bernd Finkbeiner & Jesko Hecking-Harbusch (2019): *Translating Asynchronous Games for Distributed Synthesis*. In Wan J. Fokkink & Rob van Glabbeek, editors: *30th International Conference on*

Concurrency Theory, CONCUR 2019, *LIPIcs* 140, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 26:1–26:16, doi:10.4230/LIPIcs.CONCUR.2019.26.

[2] Roderick Bloem, Sven Schewe & Ayrat Khalimov (2017): *CTL* synthesis via LTL synthesis*. In Dana Fisman & Swen Jacobs, editors: *Proceedings Sixth Workshop on Synthesis, SYNT@CAV 2017*, *EPTCS* 260, pp. 4–22, doi:10.4204/EPTCS.260.4.

[3] Alonzo Church (1957): *Applications of recursive arithmetic to the problem of circuit synthesis*. *Summaries of the Summer Institute of Symbolic Logic* 1, pp. 3–50.

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

[5] Javier Esparza, Stefan Römer & Walter Vogler (2002): *An Improvement of McMillan's Unfolding Algorithm*. *Formal Methods Syst. Des.* 20(3), pp. 285–310, doi:10.1023/A:1014746130920.

[6] Bernd Finkbeiner (2015): *Bounded Synthesis for Petri Games*. In Roland Meyer, André Platzer & Heike Wehrheim, editors: *Correct System Design - Symposium in Honor of Ernst-Rüdiger Olderog on the Occasion of His 60th Birthday, Proceedings*, *Lecture Notes in Computer Science* 9360, Springer, pp. 223–237, doi:10.1007/978-3-319-23506-6_15.

[7] Bernd Finkbeiner & Paul Gölz (2018): *Synthesis in Distributed Environments*. In Satya Lokam & R. Ramanujam, editors: *37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2017)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 93, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, pp. 28:1–28:14, doi:10.4230/LIPIcs.FSTTCS.2017.28.

[8] Bernd Finkbeiner & Ernst-Rüdiger Olderog (2017): *Petri games: Synthesis of distributed systems with causal memory*. *Information and Computation* 253, pp. 181–203, doi:10.1016/j.ic.2016.07.006. GandALF 2014.

[9] Bernd Finkbeiner & Sven Schewe (2005): *Uniform Distributed Synthesis*. In: *20th IEEE Symposium on Logic in Computer Science (LICS 2005), Proceedings*, pp. 321–330, doi:10.1109/LICS.2005.53.

[10] Blaise Genest, Hugo Gimbert, Anca Muscholl & Igor Walukiewicz (2013): *Asynchronous Games over Tree Architectures*. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska & David Peleg, editors: *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Proceedings, Part II*, *Lecture Notes in Computer Science* 7966, Springer, pp. 275–286, doi:10.1007/978-3-642-39212-2_26.

[11] Hugo Gimbert (2017): *On the Control of Asynchronous Automata*. In Satya V. Lokam & R. Ramanujam, editors: *37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2017, India*, *LIPIcs* 93, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 30:1–30:15, doi:10.4230/LIPIcs.FSTTCS.2017.30.

[12] Hugo Gimbert (2022): *Distributed Asynchronous Games With Causal Memory are Undecidable*. *Log. Methods Comput. Sci.* 18(3), doi:10.46298/lmcs-18(3:30)2022.

[13] Paul Hannibal & Ernst-Rüdiger Olderog (2022): *The Synthesis Problem for Repeatedly Communicating Petri Games*. In Luca Bernardinello & Laure Petrucci, editors: *Application and Theory of Petri Nets and Concurrency - 43rd International Conference, PETRI NETS 2022, Proceedings*, *Lecture Notes in Computer Science* 13288, Springer, pp. 236–257, doi:10.1007/978-3-031-06653-5_13.

[14] O. Kupferman & M.Y. Vardi (2001): *Synthesizing distributed systems*. *Proceedings - Symposium on Logic in Computer Science*, pp. 389–398, doi:10.1109/LICS.2001.932514.

[15] P. Madhusudan & P. S. Thiagarajan (2002): *A Decidable Class of Asynchronous Distributed Controllers*. In Lubos Brim, Petr Jancar, Mojmír Kretínský & Antonín Kucera, editors: *CONCUR 2002 - Concurrency Theory, 13th International Conference, Proceedings*, *Lecture Notes in Computer Science* 2421, Springer, pp. 145–160, doi:10.1007/3-540-45694-5_11.

[16] P. Madhusudan, P. S. Thiagarajan & Shaofa Yang (2005): *The MSO Theory of Connectedly Communicating Processes*. In Ramaswamy Ramanujam & Sandeep Sen, editors: *FSTTCS 2005: Foundations of Software*

Technology and Theoretical Computer Science, 25th International Conference, Proceedings, *Lecture Notes in Computer Science* 3821, Springer, pp. 201–212, doi:10.1007/11590156_16.

[17] Anca Muscholl (2015): *Automated Synthesis of Distributed Controllers*. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi & Bettina Speckmann, editors: *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Proceedings, Part II*, *Lecture Notes in Computer Science* 9135, Springer, pp. 11–27, doi:10.1007/978-3-662-47666-6_2.

[18] Amir Pnueli & Roni Rosner (1989): *On the Synthesis of an Asynchronous Reactive Module*. In Giorgio Ausiello, Mariangiola Dezani-Ciancaglini & Simona Ronchi Della Rocca, editors: *Automata, Languages and Programming, 16th International Colloquium, ICALP89, Proceedings*, *Lecture Notes in Computer Science* 372, Springer, pp. 652–671, doi:10.1007/BFb0035790.

[19] Amir Pnueli & Roni Rosner (1990): *Distributed Reactive Systems Are Hard to Synthesize*. In: *31st Annual Symposium on Foundations of Computer Science, Volume II*, pp. 746–757, doi:10.1109/FSCS.1990.89597.

[20] R. Rosner (1992): *Modular synthesis of reactive systems*. Ph.D. thesis, Weizmann Institute of Science, Rehovot, Israel.