Partial Order Reduction for Reachability Games

Frederik Meyer Bønneland
Department of Computer Science, Aalborg University, Denmark
frederikb@cs.aau.dk

Peter Gjøl Jensen
Department of Computer Science, Aalborg University, Denmark
pgj@cs.aau.dk

Kim Guldstrand Larsen
Department of Computer Science, Aalborg University, Denmark
kgl@cs.aau.dk

Marco Muñiz
Department of Computer Science, Aalborg University, Denmark
muniz@cs.aau.dk

Jiří Srba
Department of Computer Science, Aalborg University, Denmark
srba@cs.aau.dk

Abstract

Partial order reductions have been successfully applied to model checking of concurrent systems and practical applications of the technique show nontrivial reduction in the size of the explored state space. We present a theory of partial order reduction based on stubborn sets in the game-theoretical setting of 2-player games with reachability/safety objectives. Our stubborn reduction allows us to prune the interleaving behaviour of both players in the game, and we formally prove its correctness on the class of games played on general labelled transition systems. We then instantiate the framework to the class of weighted Petri net games with inhibitor arcs and provide its efficient implementation in the model checker TAPAAL. Finally, we evaluate our stubborn reduction on several case studies and demonstrate its efficiency.

2012 ACM Subject Classification Theory of computation → Algorithmic game theory

Keywords and phrases Petri nets, games, synthesis, partial order reduction, stubborn sets

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2019.19

1 Introduction

The state space explosion problem is the main obstacle for model checking of concurrent systems as even very simple processes running in parallel can produce an exponentially large number of possible interleavings and hence make the state space search practically intractable. One of the ways to tame this problem is by employing a variety of partial order reduction techniques, including the seminal work on stubborn set reductions by Valmari et al. [24, 23, 25].

As our main contribution, we generalize the theory of partial order reductions into the framework of 2-player games. The idea is that whenever one of the players can perform a series of moves in different sub-components of the system in parallel (without being disturbed by the other player), we may apply the classical stubborn set reductions in order to reduce the number of interleavings of independent actions. There is a number of subtle points that one has to satisfy so that the reduced game preserves the winning strategies of both players. We formulate a number of sufficient conditions that define the notion of a *stable* stubborn set reduction that guarantees the preservation of winning strategies for both players in the game. In the setting of general game labelled transition systems, we formally prove the correctness of stable reductions and we demonstrate the applicability of the framework on weighted Petri net games with inhibitor arcs. We show how to approximate in a syntax-driven manner the conditions of a stable Petri net game reduction and provide an efficient, open source implementation in the model checker TAPAAL [6]. We also implement a game engine based on dependency graphs, following the approach from [14, 5], and on several case studies evaluate the game engine both with and without the use of the stable stubborn set reduction. The experiments demonstrate that the computation of the stubborn sets has only a minor overhead and has the potential of achieving exponential reduction both in the running time as well as in the number of searched configurations. We believe that this is the first implementation of 2-player game partial order reduction technique for Petri nets working in practice.

Related Work. Partial order reductions in the non-game setting for linear time properties have previously been studied [19, 23, 18, 17] which lends itself towards the safeness or liveness properties we want to preserve for winning states. In [19] and [23] Peled and Valmari present partial order reductions for general LTL. In [18] Lehmann et al. study stubborn sets applied to a subset of LTL properties called simple linear time properties which does not require all the requirements for general LTL preservation.

The extension of partial order reductions to game-oriented formalisms and verification tasks has not yet received much attention in the literature. In [10] partial order reductions for LTL without the next operator are adapted to a subset of alternating-time temporal logic and applied to multi-agent systems. The authors considers games with imperfect information, however they also show that their technique does not work for strategies with perfect information. We assume an antagonistic environment and focus on preserving the existence of winning strategies with perfect information, reducing the state space and improving existing controller synthesis algorithms. Partial order reduction for the problem of checking bisimulation equivalance between two labelled transition systems is presented in [9]. Our partial order reduction is applied directly to a labelled transition system while theirs are applied to the bisimulation game graph. While the setting is distinctly different, our approach is more general as we allow for mixed states, provide less information to the controller, and allow for reduction in both controllable as well as environmental states. Moreover, we provide an implementation of the on-the-fly strategy synthesis algorithm and argue by a number of case studies for its practical applicability.

The work on partial order reductions for modal mu-calculus and CTL (see e.g. [21, 26]) allows us in principle to encode the game semantics as a part of the mu-calculus formula, however, there is to the best of our knowledge no literature documenting the practical applicability of this approach.

Complexity and decidability results for control synthesis in Petri nets games are not encouraging. The control synthesis problem is for many instances of Petri net formalisms undecidable [1, 2], including those that allow for inhibition [2] which we utilise to model our case studies. If the problem is decidable for a given instance of a Petri net formalism (like e.g. for bounded nets) then it is usually of exponential complexity. In fact, most questions about the behaviour of bounded Petri nets are at least PSPACE-hard [11]. Among these questions is the existence of an infinite run [7] that we need to test as one of the sufficient conditions for applying stubborn set reductions to games. Instead of using exact infinite run detection approaches like in [7], we opt for efficient overapproximation algorithms to detect cycles using both syntactic and local state information.

2 Preliminaries

▶ **Definition 1** (Game Labelled Transition System). *A (deterministic) Game Labelled Transition System (GLTS) is a tuple $G = (\mathcal{S}, A_1, A_2, \rightarrow, Goal)$ where $\mathcal{S}$ is a set of states, A_1 is a finite set of actions for player 1 (the controller), A_2 is a finite set of actions for player 2 (the environment) where $A_1 \cap A_2 = \emptyset$ and $A = A_1 \cup A_2$, $\rightarrow \subseteq \mathcal{S} \times A \times \mathcal{S}$ is a transition relation s.t. if $(s, a, s') \in \rightarrow$ and $(s, a, s'') \in \rightarrow$ then $s' = s''$, and $Goal \subseteq \mathcal{S}$ is a set of goal states.*

Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, Goal)$ be a fixed GLTS for the remainder of the section. Whenever $(s, a, s') \in \rightarrow$ we write $s \xrightarrow{a} s'$ and say that a is enabled in s and can be *fired* in s yielding s'. Otherwise we say that a is *disabled* in s. The set of *enabled* player i actions where $i \in \{1, 2\}$ in a state $s \in \mathcal{S}$ is given by $en_i(s) = \{a \in A_i \mid \exists s' \in \mathcal{S}.\ s \xrightarrow{a} s'\}$. The set of all enabled actions is given by $en(s) = en_1(s) \cup en_2(s)$. For a state $s \in \mathcal{S}$ where $en(s) \neq \emptyset$ if $en_2(s) = \emptyset$ then we call s a player 1 state, if $en_1(s) = \emptyset$ then we call s a player 2 state, and otherwise we call it a mixed state. The GLTS G is called non-mixed if all states are either player 1 or player 2 states. For a sequence of actions $w = a_1 a_2 \cdots a_n \in A^*$ we write $s \xrightarrow{w} s'$ if $s \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} s'$. If $w \in A^\omega$, i.e. it is infinite, then we write $s \xrightarrow{w}$. Actions that are a part of w are said to occur in w. A sequence of states induced by $w \in A^* \cup A^\omega$ is called a *run* and is written as $\pi = s_0 s_1 \cdots$. We use $\Pi_G(s)$ to denote the set of all runs starting from a state $s \in \mathcal{S}$ in GLTS G, s.t. for all $s_0 s_1 \cdots \in \Pi_G(s)$ we have $s_0 = s$, and $\Pi_G = \bigcup_{s \in \mathcal{S}} \Pi_G(s)$ as the set of all runs. The length of a run π (number of states in the run) is given by the function $\ell : \Pi_G \rightarrow \mathbb{N}^0 \cup \{\infty\}$. A position in a run $\pi = s_0 s_1 \ldots \in \Pi_G(s)$ is a natural number $i \in \mathbb{N}^0$ that refers to the state s_i and is written as π_i. A position i can range from 0 to $\ell(\pi)$ s.t. if π is infinite then $i \in \mathbb{N}^0$ and otherwise $0 \leq i \leq \ell(\pi)$. Let $\Pi_G^{max}(s)$ be the set of all maximal runs starting from s, defined as $\Pi_G^{max}(s) = \{\pi \in \Pi_G(s) \mid \ell(\pi) = \infty \vee en(\pi_{\ell(\pi)}) = \emptyset)\}$. We omit the GLTS G from the subscript of run sets if it is clear from the context.

A reduced game is defined by a function called a reduction.

▶ **Definition 2** (Reduction). *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, Goal)$ be a GLTS. A* reduction *is a function $St : \mathcal{S} \rightarrow 2^A$.*

▶ **Definition 3** (Reduced Game). *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, Goal)$ be a GLTS and St be a reduction. The* reduced game *of G by the reduction St is given by $G_{St} = (\mathcal{S}, A_1, A_2, \xrightarrow[St]{}, Goal)$ where $s \xrightarrow[St]{a} s'$ iff $s \xrightarrow{a} s'$ and $a \in St(s)$.*

The set of actions $St(s)$ is the *stubborn set* of s with the reduction St. The set of non-stubborn actions for s is defined as $\overline{St(s)} = A \setminus St(s)$.

A (memoryless) strategy is a function which proposes the next action player 1 wants to be fired.

▶ **Definition 4** (Strategy). *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, Goal)$ be a GLTS. A* strategy *is a function $\sigma : \mathcal{S} \rightarrow A_1 \cup \{\bot\}$ where for all $s \in \mathcal{S}$ we have if $en_1(s) \neq \emptyset$ then $\sigma(s) \in en_1(s)$ else $\sigma(s) = \bot$.*

The intuition is that in order to ensure progress, player 1 always has to propose an action if she has an enabled action. Let σ be a fixed strategy for the remainder of the section. We define a function $next_\sigma(s)$ that returns the set of actions considered at $s \in \mathcal{S}$ under σ as:

$$next_\sigma(s) = \begin{cases} en_2(s) \cup \sigma(s) & \text{if } \sigma(s) \neq \bot \\ en_2(s) & \text{otherwise.} \end{cases}$$

Let $\Pi_\sigma^{max}(s) \subseteq \Pi^{max}(s)$ be the set of maximal runs subject to σ starting at $s \in \mathcal{S}$, defined as:

$$\Pi_\sigma^{max}(s) = \{\pi \in \Pi^{max}(s) \mid \forall i \in \{1, ..., \ell(\pi)\}.\ \exists a \in next_\sigma(\pi_{i-1}).\ \pi_{i-1} \xrightarrow{a} \pi_i)\} \ .$$

▶ **Definition 5** (Winning Strategy). *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, Goal)$ be a GLTS and $s \in \mathcal{S}$ be a state. A strategy σ is a* winning strategy *for player 1 at s in G iff for all $\pi \in \Pi^{max}_{\sigma}(s)$ there exists a position i s.t. $\pi_i \in Goal$.*

If a state is winning for player 1 in G then no matter what action sequence the environment chooses, eventually a goal state is reached. Furthermore, for a given winning strategy σ at s in G there is a finite number $n \in \mathbb{N}$ such that a goal state is always reached with at most n action firings. We call the minimum such number the *strategy depth* of σ.

▶ **Definition 6** (Strategy Depth). *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, Goal)$ be a GLTS, $s \in \mathcal{S}$ a winning state for player 1 in G where $s \notin Goal$, and σ a winning strategy at s in G. Then $n \in \mathbb{N}^0$ is the* depth *of σ at s in G if:*

- *for all $\pi \in \Pi^{max}_{G,\sigma}(s)$ there exists $0 \leq i \leq n$ s.t. $\pi_i \in Goal$, and*
- *there exists $\pi' \in \Pi^{max}_{G,\sigma}(s)$ s.t. $\pi'_n \in Goal$ and for all $0 \leq j < n$ we have $\pi'_j \notin Goal$.*

▶ **Lemma 7.** *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, Goal)$ be a GLTS, $s \in \mathcal{S}$ a winning state for player 1 in G, and σ a winning strategy at s in G.*

1. *There exists $n \in \mathbb{N}^0$ that is the depth of σ at s in G.*
2. *For all $a \in next_{\sigma}(s)$ where $s \xrightarrow{a} s'$, the depth of σ at s' in G is m such that $0 \leq m < n$.*

A set of actions for a given state and a given set of goal states is called an interesting set if for any path leading to any goal state at least one action from the set of interesting actions has to be fired.

▶ **Definition 8** (Interesting Actions). *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, Goal)$ be a GLTS and $s \in \mathcal{S}$ a state. A set of actions $A_s(Goal) \subseteq A$ is called an* interesting set *of actions for s and $Goal$ if whenever $s \notin Goal$, $w = a_1 \cdots a_n \in A^*$, $s \xrightarrow{w} s'$, and $s' \in Goal$ then there exists $1 \leq i \leq n$ s.t. $a_i \in A_s(Goal)$.*

▶ **Example 9.** In Figure 1 we see an example of a GLTS $G = (\mathcal{S}, A_1, A_2, \rightarrow, Goal)$ where $\mathcal{S} = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$ is the states denoted by a circle, $A_1 = \{a, b, c\}$ is the player 1 actions, $A_2 = \{d\}$ is the player 2 actions, and $\rightarrow$ is denoted by the solid and dashed lines between states and labelled with a corresponding action for player 1 and 2, respectively. Let $Goal = \{s_6\}$.

We consider different proposals for a set of interesting actions for the state s_1. The set $\{b\}$ is an interesting set of actions in s_1 since the goal state s_6 cannot be reached without firing b at least once. Furthermore, the sets $\{a\}$ and $\{c\}$ are also sets of interesting actions for the state s_1.

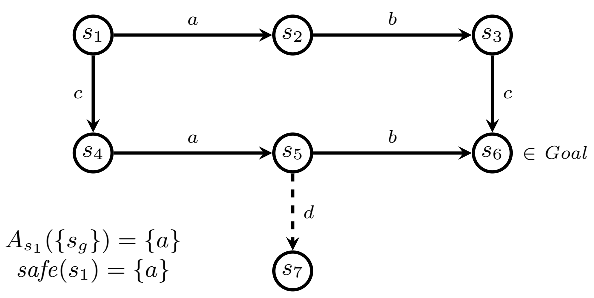

Figure 1 Example of safe and interesting sets of actions for a state s_1.

Player 1 has to consider his safe actions. A player 1 action is *safe* in a given player 1 state if by firing it before any sequence of player 1 actions excluding the safe action then it will never reach a player 2 state.

▶ **Definition 10** (Safe Action). *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, Goal)$ be a GLTS and $s \in \mathcal{S}$ a state s.t. $en_2(s) = \emptyset$. An action $a \in A_1 \cap en_1(s)$ is* safe *in s if whenever $w \in (A_1 \setminus \{a\})^*$ and $s \xrightarrow{w} s'$ and $en_2(s') = \emptyset$ and $s \xrightarrow{aw} s''$ then $en_2(s'') = \emptyset$. The set of all safe actions for s is written as $safe(s)$.*

▶ **Example 11.** Consider again the GLTS in Figure 1. We reasoned in Example 9 that the set $\{b\}$ is an interesting set of actions in the state s_1. However, b is not a safe player 1 action in s_1 since by definition b has to be enabled at s_1 to be safe. The enabled actions at s_1 is $en(s_1) = \{a, c\}$, and between these two actions only a is safe. The action c is not safe since we have $s_1 \xrightarrow{a} s_2$ and $en_2(s_2) = \emptyset$ but $s_1 \xrightarrow{ca} s_5$ and $en_2(s_5) \neq \emptyset$. It is clear from the figure that s_1 is a winning state for player 1 with a as the action player 1 should choose in order to win.

3 Stable Reduction

A reduction St provides at each state a set of actions which are sufficient to fire such that a certain property is preserved in the reduced game. In the game setting, we have to guarantee the preservation of winning strategies for both players in the game. In what follows, we shall introduce a number of conditions that preserve winning strategies and we call such a reduction a *stable* one.

For the remainder of the section let $s \in S$ be a state and $Goal \subseteq \mathcal{S}$ be a set of goal states, and let $A_s(Goal)$ be a fixed set of interesting actions for s and $Goal$.

▶ **Definition 12** (Stable Strategy Conditions). *A reduction St is called* stable *if St satisfies for every $s \in S$ Conditions **I**, **W**, **R**, **G1**, **G2**, **S**, **C**, and **D**.*

I *If $en_1(s) \neq \emptyset$ and $en_2(s) \neq \emptyset$ then $en(s) \subseteq St(s)$.*

W *For all $w \in \overline{St(s)}^*$ and all $a \in St(s)$ if $s \xrightarrow{wa} s'$ then $s \xrightarrow{aw} s'$.*

R *$A_s(Goal) \subseteq St(s)$*

G1 *For all $w \in \overline{St(s)}^*$ if $en_2(s) = \emptyset$ and $s \xrightarrow{w} s'$ then $en_2(s') = \emptyset$.*

G2 *For all $w \in \overline{St(s)}^*$ if $en_1(s) = \emptyset$ and $s \xrightarrow{w} s'$ then $en_1(s') = \emptyset$.*

S *$en_1(s) \cap St(s) \subseteq safe(s)$ or $en_1(s) \subseteq St(s)$*

C *For all $a \in A_2$ if there exists $w \in A_2^\omega$ s.t. $s \xrightarrow{w}$ and a occurs infinitely often in w then $a \in St(s)$.*

D *If $en_2(s) \neq \emptyset$ then there exists $a \in en_2(s) \cap St(s)$ s.t. for all $w \in \overline{St(s)}^*$ where $s \xrightarrow{w} s'$ we have $a \in en_2(s')$.*

If s is a mixed state then Condition **I** ensures all enabled actions are included in the reduction. That is, we do not attempt to reduce the state space from this state. Condition **W** ensures that we can swap the ordering of firing actions, such that firing the actions included in the reduction first still ensures we can reach a given state, i.e. they are independent. Condition **R** ensures that a goal state cannot be reached solely by exploring actions not in the reduction, i.e. reachability of paths to goal states are preserved in the reduction. Conditions **G1** and **G2** ensure that if a state is a player 1 (or player 2) state then a player 2 (or player 1) state cannot be reached solely by exploring actions not in the reduction, i.e. reachability of paths to mixed states and opposing player states are preserved in the reduction. Condition **S** ensures either that all stubborn player 1 actions are also safe and if this is not the case then all player 1 actions are included in the reduction. Condition **C** preserves infinite paths on which only player 2 actions are fired. Condition **D** ensures that at least one player 2 action cannot be disabled solely by exploring actions not in the reduction.

▶ **Example 13.** In Figure 2 we see an example of a GLTS $G = (\mathcal{S}, A_1, A_2, \rightarrow, \mathit{Goal})$ where $\mathcal{S} = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}\}$ are the states denoted by a circle, $A_1 = \{a, b, c, d\}$ is the player 1 actions, $A_2 = \{e\}$ is the player 2 action, and $\rightarrow$ is denoted by the solid and dashed lines between states and labelled with a corresponding action for player 1 and 2, respectively. Let $\mathit{Goal} = \{s_8\}$ be a fixed set of goal states for this GLTS and $A_{s_1}(\mathit{Goal}) = \{a\}$ be a set of interesting actions. Thick lines indicate transitions and states that are preserved by a stable reduction St, while thin lines indicates transitions and states that are removed by the same reduction. For state s_1 we have $\mathit{St}(s_1) = \{a, c\}$ as it is sufficient to satisfy the stable reduction conditions. We satisfy **G1** since c has to be fired once to reach s_7. For $s_1 \xrightarrow{ba} s_5$ and $s_1 \xrightarrow{bc} s_7$ we also have $s_1 \xrightarrow{ab} s_5$ and $s_1 \xrightarrow{cb} s_7$, so **W** is satisfied. Clearly $\mathit{St}(s_1)$ is an interesting set since $A_{s_1}(\mathit{Goal}) \subseteq \mathit{St}(s_1)$, so **R** is satisfied. Condition **S** is satisfied since $\mathit{St}(s_1) \cap \mathit{en}(s_1) \subseteq \mathit{safe}(s_1)$. We have that **I**, **G2**, **C**, and **D** are satisfied as well since their antecedents are not true.

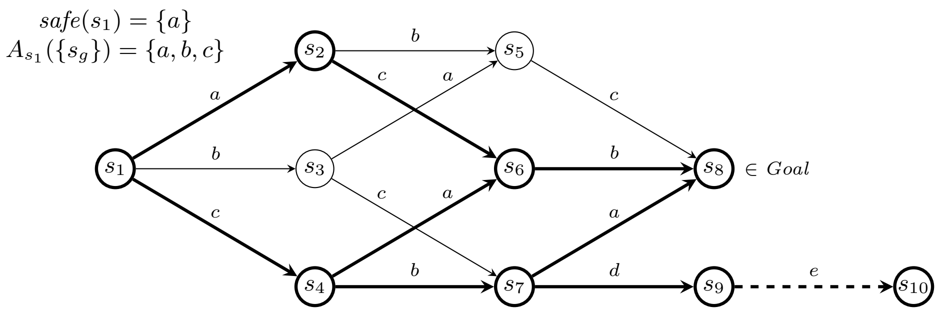

Figure 2 Example of a stable reduction for a state s_1.

We shall first notice the fact that if a goal state is reachable from some state, then the state has at least one enabled action that is also in the stubborn set.

▶ **Lemma 14.** *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, \mathit{Goal})$ be a GLTS, St a reduction that satisfies Conditions **W** and **R**, and $s \in \mathcal{S}$ a state. If there exists $w \in A^*$ s.t. $s \xrightarrow{w} s'$ and $s' \in \mathit{Goal}$ then $\mathit{St}(s) \cap \mathit{en}(s) \neq \emptyset$.*

The correctness of stable stubborn reductions is proved by the next two lemmas. Both lemmas are proved by induction on the depth of a winning strategy for player 1 in the game.

▶ **Lemma 15.** *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, \mathit{Goal})$ be a GLTS and St a stable reduction. For all $s \in \mathcal{S}$ if state s is winning for player 1 in G then state s is winning for player 1 in G_{St}.*

▶ **Lemma 16.** *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, \mathit{Goal})$ be a GLTS and St a stable reduction. For all $s \in \mathcal{S}$ if state s is winning for player 1 in G_{St} then state s is winning for player 1 in G.*

We can now present the main theorem showing that stable reductions preserve the winning strategies of both players in the game.

▶ **Theorem 17** (Strategy Preservation for GLTS). *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow)$ be a GLTS and St a stable reduction. For all $s \in \mathcal{S}$ state s is winning for player 1 in G iff state s is winning for player 1 in G_{St}.*

Moreover, for non-mixed games we can simplify the conditions of stable reductions by removing the requirement on safe actions.

▶ **Theorem 18** (Strategy Preservation for Non-Mixed GLTS). *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow)$ be a non-mixed GLTS and St a stable reduction with Condition* ***S*** *excluded. For all $s \in \mathcal{S}$ state s is winning for player 1 in G iff state s is winning for player 1 in G_{St}.*

4 Stable Reductions on Petri Net Games

We now introduce the formalism of Petri net games and show how to algorithmically construct stable reductions in a syntax-driven manner.

▶ **Definition 19** (Petri Net Game). *A Petri net game is a tuple $N = (P, T_1, T_2, W, I)$ where P and $T = T_1 \uplus T_2$ are finite sets of places and transitions, respectively, such that $P \cap T = \emptyset$ and where transitions are partitioned into player 1 and player 2 transitions, $W : (P \times T) \cup (T \times P) \rightarrow \mathbb{N}^0$ is a weight function for regular arcs, and $I : (P \times T) \rightarrow \mathbb{N}^\infty$ is a weight function for inhibitor arcs. A* marking *M on N is a function $M : P \rightarrow \mathbb{N}^0$. The set $\mathcal{M}(N)$ is the set of all markings for N.*

For the rest of this section, let $N = (P, T_1, T_2, W, I)$ be a fixed Petri net game such that $T = T_1 \cup T_2$. Let us first fix some useful notation. For a place or transition x, we denote the *preset* of x as ${}^\bullet x = \{y \in P \cup T \mid W((y,x)) > 0\}$, and the *postset* of x as $x^\bullet = \{y \in P \cup T \mid W((x,y)) > 0\}$. For a transition t, we denote the *inhibitor preset* of t as ${}^\circ t = \{p \in P \mid I((p,t)) \neq \infty\}$, and the *inhibitor postset* of a place p as $p^\circ = \{t \in T \mid I((p,t)) \neq \infty\}$. For a place p we define the *increasing preset* of p, containing all transitions that increase the number of tokens in p, as ${}^+p = \{t \in {}^\bullet p \mid W((t,p)) > W((p,t))\}$, and similarly the *decreasing postset* of p as $p^- = \{t \in p^\bullet \mid W((t,p)) < W((p,t))\}$. For a transition t we define the *decreasing preset* of t, containing all places that have their number of tokens decreased by t, as ${}^-t = \{p \in {}^\bullet t \mid W((p,t)) > W((t,p))\}$, and similarly the *increasing postset* of t as $t^+ = \{p \in t^\bullet \mid W((p,t)) < W((t,p))\}$. For a set X of either places or transitions, we extend the notation as ${}^\bullet X = \bigcup_{x \in X} {}^\bullet x$ and $X^\bullet = \bigcup_{x \in X} x^\bullet$, and similarly for the other operators.

A Petri net $N = (P, T_1, T_2, W, I)$ defines a GLTS $G(N) = (\mathcal{S}, A_1, A_2, \rightarrow, \mathit{Goal})$ where $\mathcal{S} = \mathcal{M}(N)$ is the set of all markings, $A_1 = T_1$ is the set of player 1 actions, $A_2 = T_2$ is the set of player 2 actions, $M \xrightarrow{t} M'$ whenever for all $p \in P$ we have $M(p) \geq W((p,t))$, $M(p) < I((p,t))$ and $M'(p) = M(p) - W((p,t)) + W((t,p))$, and $\mathit{Goal} \in \mathcal{M}(N)$ is the set of goal markings, described by a simple reachability logic formula defined below.

By analysing the increasing presets and postsets, we can identify a sufficient condition for a transition to be safe.

▶ **Lemma 20** (Safe Transition). *Let $N = (P, T_1, T_2, W, I)$ be a Petri net game and $t \in T$ a transition. If $t^+ \cap {}^\bullet T_2 = \emptyset$ and ${}^-t \cap {}^\circ T_2 = \emptyset$ then t is safe in any marking of N.*

Let E_N be the set of marking expressions in N given by the abstract syntax (here e ranges over E_N):

$$e ::= c \mid p \mid e_1 \oplus e_2$$

where $c \in \mathbb{N}^0$, $p \in P$, and $\oplus \in \{+, -, *\}$. An expression $e \in E_N$ is evaluated relatively to a marking $M \in \mathcal{M}(N)$ by the function $\mathit{eval}_M : E_N \rightarrow \mathbb{Z}$ where $\mathit{eval}_M(c) = c$, $\mathit{eval}_M(p) = M(p)$ and $\mathit{eval}_M(e_1 \oplus e_2) = \mathit{eval}_M(e_1) \oplus \mathit{eval}_M(e_2)$.

Expression e	$incr_M(e)$	$decr_M(e)$
c	$\emptyset$	$\emptyset$
p	${}^+p$	p^-
$e_1 + e_2$	$incr_M(e_1) \cup incr_M(e_2)$	$decr_M(e_1) \cup decr_M(e_2)$
$e_1 - e_2$	$incr_M(e_1) \cup decr_M(e_2)$	$decr_M(e_1) \cup incr_M(e_2)$
$e_1 \cdot e_2$	$incr_M(e_1) \cup decr_M(e_1) \cup incr_M(e_2) \cup decr_M(e_2)$	$incr_M(e_1) \cup decr_M(e_1) \cup incr_M(e_2) \cup decr_M(e_2)$

Table 1 Increasing and decreasing transitions for expression $e \in E_N$.

In Table 1 we define the functions $incr_M : E_N \to 2^T$ and $decr_M : E_N \to 2^T$ that, given an expression $e \in E_N$, return the set of transitions that can (when fired) increase resp. decrease the evaluation of e.

▶ **Lemma 21** ([4]). *Let $N = (P, T_1, T_2, W, I)$ be a Petri net and $M \in \mathcal{M}(N)$ a marking. Let $e \in E_N$ and let $M \xrightarrow{w} M'$ where $w = t_1 t_2 \ldots t_n \in T^*$.*

- *If $eval_M(e) < eval_{M'}(e)$ then there is i, $1 \le i \le n$, such that $t_i \in incr_M(e)$.*
- *If $eval_M(e) > eval_{M'}(e)$ then there is i, $1 \le i \le n$, such that $t_i \in decr_M(e)$.*

We can now define the set of reachability formulae Φ_N that evaluate over the markings in N as follows:

$$\varphi ::= \ true \mid false \mid t \mid e_1 \bowtie e_2 \mid deadlock \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi$$

where $e_1, e_2 \in E_N$, $t \in T$ and $\bowtie \in \{<, \le, =, \ne, >, \ge\}$.

The satisfaction relation for a formula $\varphi \in \Phi_N$ in a marking M is defined as expected:

$$\begin{array}{ll}
M \models true & \\
M \models t & \text{iff } t \in en(M) \\
M \models e_1 \bowtie e_2 & \text{iff } eval_M(e_1) \bowtie eval_M(e_2) \\
M \models deadlock & \text{iff } en(M) = \emptyset \\
M \models \varphi_1 \wedge \varphi_2 & \text{iff } M \models \varphi_1 \text{ and } M \models \varphi_2 \\
M \models \varphi_1 \vee \varphi_2 & \text{iff } M \models \varphi_1 \text{ or } M \models \varphi_2 \\
M \models \neg\varphi & \text{iff } M \not\models \varphi
\end{array}$$

Our aim is to be able to preserve at least one execution to the set $Goal = \{M \in \mathcal{M}(N) \mid M \models \varphi\}$ for a given formula φ describing the set of goal markings. In order to achieve this, we define the set of interesting transitions $A_M(\varphi)$ for a formulae φ so that any firing sequence of transitions from a marking that does not satisfy φ leading to a marking that satisfies φ must contain at least one interesting transition. Table 2 provides the definition of $A_M(\varphi)$ that is similar to the one presented in [4] for the non-game setting, except for the conjunction where we in our setting use Equation (1) that provides an optimisation for Condition **S** and possibly ends with a smaller set of interesting transitions.

$$A_M(\varphi_1 \wedge \varphi_2) = \begin{cases} A_M(\varphi_1) & \text{if } M \models \varphi_2 \\ A_M(\varphi_2) & \text{if } M \models \varphi_1 \\ A_M(\varphi_1) & \text{if } M \not\models \varphi_1 \text{ and } A_M(\varphi_1) \subseteq safe(M) \\ A_M(\varphi_2) & \text{if } M \not\models \varphi_2 \text{ and } A_M(\varphi_2) \subseteq safe(M) \\ A_M(\varphi_1) \cup A_M(\varphi_2) & \text{otherwise} \end{cases} \qquad (1)$$

φ	$A_M(\varphi)$	$A_M(\neg\varphi)$
deadlock	$({}^\bullet t)^- \cup {}^+({}^\circ t)$ for some $t \in en(M)$	$\emptyset$
t	${}^+p$ for some $p \in {}^\bullet t$ where $M(p) < W((p,t))$ or p^- for some $p \in {}^\circ t$ where $M(p) \geq I((p,t))$	$({}^\bullet t)^- \cup {}^+({}^\circ t)$
$e_1 < e_2$	$decr_M(e_1) \cup incr_M(e_2)$	$A_M(e_1 \geq e_2)$
$e_1 \leq e_2$	$decr_M(e_1) \cup incr_M(e_2)$	$A_M(e_1 > e_2)$
$e_1 > e_2$	$incr_M(e_1) \cup decr_M(e_2)$	$A_M(e_1 \leq e_2)$
$e_1 \geq e_2$	$incr_M(e_1) \cup decr_M(e_2)$	$A_M(e_1 < e_2)$
$e_1 = e_2$	$decr_M(e_1) \cup incr_M(e_2)$ if $eval_M(e_1) > eval_M(e_2)$ $incr_M(e_1) \cup decr_M(e_2)$ if $eval_M(e_1) < eval_M(e_2)$	$A_M(e_1 \neq e_2)$
$e_1 \neq e_2$	$incr_M(e_1) \cup decr_M(e_1) \cup incr_M(e_2) \cup decr_M(e_2)$	$A_M(e_1 = e_2)$
$\varphi_1 \wedge \varphi_2$	Defined in Equation (1)	$A_M(\neg\varphi_1 \vee \neg\varphi_2)$
$\varphi_1 \vee \varphi_2$	$A_M(\varphi_1) \cup A_M(\varphi_2)$	$A_M(\neg\varphi_1 \wedge \neg\varphi_2)$

Table 2 Interesting transitions of φ (assuming $M \not\models \varphi$, otherwise $A_M(\varphi) = \emptyset$).

The desired property of the set of interesting transitions is formulated below.

▶ **Lemma 22.** *Let $N = (P, T_1, T_2, W, I)$ be a Petri net, $M \in \mathcal{M}(N)$ a marking, and $\varphi \in \Phi_N$ a formula. If $M \not\models \varphi$ and $M \xrightarrow{w} M'$ where $w \in \overline{A_M(\varphi)}^*$ then $M' \not\models \varphi$.*

We shall now discuss a method for detecting the impossibility of infinite firing sequences consisting of purely player 2 transitions. Let $Fin \subseteq P \cup T_2$ be the smallest set that for every $p \in P$ and every $t \in T_2$ satisfies:

1. $p \in Fin$ whenever $W((p,t)) > W((t,p))$ for every $t \in {}^\bullet p \cap T_2$,
2. $t \in Fin$ whenever ${}^-t \cap Fin \neq \emptyset$, and
3. $p \in Fin$ whenever ${}^\bullet p \cap T_2 \subseteq Fin$.

It is easy to observe that by performing any infinite firing sequence of T_2 transitions, only finitely many tokens can be added to any place from Fin and the infinite firing sequence contains only finitely many occurrences of any transition from Fin. We let $Inf = (P \cup T_2) \setminus Fin$ denote the complement of the set Fin. Note that Fin and Inf only has to be computed once as it is independent of any specific marking.

In Algorithm 1 we present an overapproximation algorithm for detecting cycles of player 2 transitions. The algorithm first checks for orphan transitions (with empty preset) and includes them to the cycle transitions. Then, for a given marking M, it overapproximates the set *MarkedPlaces* of possible places that can be marked by firing T_2 transitions from M. Finally, every transition from the set Inf that has its preset marked is added to the set of possible infinite cycle transitions.

▶ **Lemma 23.** *Let $N = (P, T_1, T_2, W, I)$ be a Petri net game and $M \in \mathcal{M}(N)$. Algorithm 1 terminates and if there exists $w \in T_2^\omega$ s.t. $M \xrightarrow{w}$ where $t \in T_2$ occurs infinitely often in w then $t \in cycle(N, M)$.*

We can now provide a list of syntactic conditions that guarantee the stability of a given reduction.

▶ **Theorem 24** (Stable Reduction Preserving Closure). *Let $N = (P, T_1, T_2, W, I)$ be a Petri net game, φ a formula, and St a reduction of $G(N)$ such that for all $M \in \mathcal{M}(N)$ the following conditions hold.*

1. *If $en_1(M) \neq \emptyset$ and $en_2(M) \neq \emptyset$ then $en(M) \subseteq St(M)$.*
2. *If $en_1(M) \cap St(M) \not\subseteq safe(M)$ then $en_1(M) \subseteq St(M)$.*

Algorithm 1: *cycle*(N, M): Overapproximation algorithm for computing the set of transitions that may appear in infinite player 2 computations.

```
input  : $N = (P, T_1, T_2, W, I)$ and $M \in \mathcal{M}(N)$
output : If a transition $t \in T_2$ appears infinitely often on some infinite firing
         sequence of $T_2$ transitions from the marking $M$ then $t \in cycle(N, M)$.
$CycleTransitions := \emptyset$;
foreach $t \in T_2$ do
    if $^\bullet t = \emptyset$ then
        $CycleTransitions := CycleTransitions \cup \{t\}$;
$FreshPlaces := \{p \in P \mid M(p) > 0\}$; $MarkedPlaces := \emptyset$;
while $MarkedPlaces \neq FreshPlaces$ do
    $MarkedPlaces := FreshPlaces$;
    foreach $p \in P \setminus MarkedPlaces$ do
        if $\exists t \in T_2.\ p \in t^\bullet \wedge {}^\bullet t \subseteq MarkedPlaces$ then
            $FreshPlaces := FreshPlaces \cup \{p\}$;
foreach $t \in Inf \cap T_2$ do
    if $^\bullet t \subseteq MarkedPlaces$ then
        $CycleTransitions := CycleTransitions \cup \{t\}$;
return $CycleTransitions$;
```

3. $A_M(\varphi) \subseteq St(M)$
4. *If* $en_1(M) = \emptyset$ *then* $T_1 \subseteq St(M)$.
5. *If* $en_2(M) = \emptyset$ *then* $T_2 \subseteq St(M)$.
6. $cycle(N, M) \subseteq St(M)$
7. *For all* $t \in St(M)$ *if* $t \notin en(M)$ *then either*
   a. *there exists* $p \in {}^\bullet t$ *s.t.* $M(p) < W((p, t))$ *and* $^+p \subseteq St(s)$, *or*
   b. *there exists* $p \in {}^\circ t$ *s.t.* $M(p) \geq I((p, t))$ *and* $p^- \subseteq St(s)$.
8. *For all* $t \in St(M)$ *if* $t \in en(M)$ *then*
   a. *for all* $p \in {}^-t$ *we have* $p^\bullet \subseteq St(M)$, *and*
   b. *for all* $p \in t^+$ *we have* $p^\circ \subseteq St(M)$.
9. *If* $en_2(M) \neq \emptyset$ *then there exists* $t \in en_2(M) \cup St(M)$ *s.t.* $({}^\bullet t)^- \cup {}^+({}^\circ t) \subseteq St(M)$.

Then St satisfies ***I****,* ***W****,* ***R****,* ***G1****,* ***G2****,* ***S****,* ***C****, and* ***D****.*

In Algorithm 2 we provide a pseudocode for calculating stubborn sets for a given marking. The algorithm calls Algorithm 3 that saturates a given set to satisfy Conditions 7 and 8.

▶ **Theorem 25.** *Algorithm 2 terminates and returns* $St(M)$ *for some stable reduction* St.

▶ Remark 26. In the actual implementation of the algorithm, we first saturate only over the set of interesting transitions and in the case that $Saturate(A_M(\varphi)) \cap en(M) = \emptyset$, we do not explore any of the successors of the marking M as we know that no goal marking can be reached from M (this follows from Lemma 14).

Algorithm 2: Computation of $St(M)$ for some stable reduction St.

```
input  : A Petri net game $N = (P, T_1, T_2, W, I)$ and $M \in \mathcal{M}(N)$ and formula $\varphi$
output : $X \subseteq T$ where $X$ is a stable stubborn set for $M$
1  if $en(M) = \emptyset$ then
2      return $T$;
3  if $en_1(M) \neq \emptyset \wedge en_2(M) \neq \emptyset$ then
4      return $T$;
5  $Y := \emptyset$;
6  if $en_1(M) = \emptyset$ then
7      Pick any $t \in en_2(M)$;
8      $Y := T_1 \cup ({}^{\bullet}t)^{-} \cup {}^{+}({}^{\circ}t)$;
9      $Y := Y \cup cycle(N, M)$;
10 else
11     $Y := T_2$;
12 $Y := Y \cup A_M(\varphi)$;
13 $X := Saturate(Y)$;
14 if $X \cap en_1(M) \nsubseteq safe(M)$ then
15     return $T$;
16 return $X$;
```

5 Implementation and Experiments

We extend the Petri net verification engine `verifypn` [12], a part of TAPAAL tool suite [6], to experimentally demonstrate the viability of our approach. The synthesis algorithm for solving Petri net games is an adaptation of the dependency graph fixed-point computation from [15, 14] that we reimplement in `C++` while utilising PTries [13] for efficient state storage. The source code is available under GPLv3 [3]. We conduct a series of experiments using the following scalable case studies.

- In *Autonomous Intersection Management* (AIM) vehicles move at different speeds towards an intersection and we want to ensure the absence of collisions. We model the problem as a Petri net game and refer to each instance as AIM-W-X-Y-Z where W is the number of intersections with lanes of length X, Z is the number of cars, and Y is the number of different speeds for each car. The controller assign speeds to cars while the environment aims to cause a collision. The goal marking is where all cars reach their destinations while there are no collisions.
- We reformulate the classical *Producer Consumer System* (PCS) as a Petri net game. In each instance PCS-N-K the total of N consumers (controlled by the environment) and N producers (controlled by the controller) share N buffers. Each consumer and producer has a fixed buffer to consume/produce from/to, and each consumer/producer has K different randomly chosen consumption/production rates. The game alternates in rounds where the player choose for each consumer/producer appropriate buffers and rates. The goal of the game is to ensure that the consumers have always enough products in the selected buffers while at the same time the buffers have limited capacity and may not overflow.
- The *Railway Scheduling Problem* contains four instances modelling the Danish train station Lyngby and three of its smaller variants. The scheduling problem, including the

Algorithm 3: *Saturate*(Y)

```
1  $X := \emptyset$;
2  while $Y \neq \emptyset$ do
3      Pick any $t \in Y$;
4      if $t \notin en(M)$ then
5          if $\exists p \in {}^\bullet t.\ M(p) < W((p,t))$ then
6              Pick any $p \in {}^\bullet t$ s.t. $M(p) < W((p,t))$;
7              $Y := Y \cup ({}^+p \setminus X)$;
8          else
9              Pick any $p \in {}^\circ t$ s.t. $M(p) \geq I((p,t))$;
10             $Y := Y \cup (p^- \setminus X)$;
11     else
12         $Y := Y \cup ((({}^-t)^\bullet \cup (t^+)^\circ) \setminus X)$;
13     $X := X \cup \{t\}$;
14     $Y := Y \setminus \{t\}$;
15 return $X$;
```

station layout, was originally described as a game in [16] and each instance is annotated by a number N representing the number of trains that migrate through the railway network. The controller controls the lights and switches, while the environment moves the trains. The goal of the controller is to make sure that all trains reach (without any collisions) their final destinations.

- The *Nim* (NIM-K-S) Petri net game was described in [22] as a two player game where the players in rounds repeatedly remove between 1 and K pebbles from an initial stack containing S pebbles. The player that has a turn and empty stack of pebbles loses. In our (equivalent) model, we are instead adding pebbles and the player that first adds to or above the given number S loses.
- The *Manufacturing Workflow* (MW) contains instances of a software product line Petri net model presented in [20]. The net describes a series of possible ways of configuring a product (performed by the environment) while the controller aims to construct a requested product. The model instance MW-N contains N possible choices of product features.
- The *Order Workflow* (OW) Petri net game model is taken from [8] and the goal of the game is to synthesise a strategy that guarantees a workflow soundness, irrelevant of the choices made by the environment. We scale the workflow by repeatedly re-initialising the workflow N times (denoted by OW-N).

All experimental evaluation is run on AMD Opteron 6376 Processors with 120 GB memory limitation and 12 hours timeout (we measure only the execution time without the parsing time of the models). We use for all experiments the depth first search strategy and we only report the examples where the algorithms both with and without partial order reduction returned a result within the time and memory limits. We provide a reproducibility package with all models and experimental data [3].

5.1 Results

Table 3 shows the experimental evaluation, displaying the relative gain in computation time (in seconds) without and with partial order reduction as well in the number of explored

	Time (seconds)		Markings ×1000		Reduction	
Model	NORMAL	POR	NORMAL	POR	%Time	%Markings
AIM-13-100-6-11	117.9	46.6	1702	514	60	70
AIM-13-100-6-16	173.9	63.2	2464	746	64	70
AIM-13-150-9-16	337.0	219.9	3696	2454	35	34
AIM-13-150-9-21	408.0	294.1	4853	3331	28	31
AIM-14-150-9-16	449.9	278.4	4259	2865	38	33
AIM-15-150-9-16	534.5	384.9	4861	3204	28	34
PCS-2-2	24.0	19.9	9629	6554	17	32
PCS-2-3	116.1	90.9	61990	39114	22	37
PCS-2-4	399.1	283.3	240510	145109	29	40
LyngbySmall-2	3.4	1.0	1803	444	71	75
LyngbySmall-3	23.1	26.1	11473	10701	-13	7
LyngbySmall-4	144.3	193.3	65371	70008	-34	-7
Lyngby-2	3292.0	215.5	1511749	87214	93	94
NIM-5-49500	9.2	3.4	5054	892	63	82
NIM-7-49500	32.7	3.9	24039	1159	88	95
NIM-9-49500	165.1	4.7	114235	1522	97	99
NIM-11-49500	710.7	8.2	533516	1877	99	100
MW-40	735.3	0.2	69439	9	100	100
MW-50	1952.0	0.2	135697	11	100	100
MW-60	4417.0	0.3	234570	13	100	100
OW-10000	0.9	0.7	320	240	22	25
OW-100000	11.1	7.8	3200	2400	30	25
OW-1000000	137.7	109.8	32000	24000	20	25

Table 3 Experiments with and without partial order reduction (POR and NORMAL).

markings (in thousands). The results demonstrate significant reductions across all models, in some cases like in NIM and MW even of several degrees of magnitude. Other models like PCS, AIM and OW show a moderate but significant reduction. We observe that the time reduction is generally only few percent smaller than the reduction in the number of explored markings, showing typically between 2% to 20% overhead for computing (on-the-fly) the stubborn sets. For two instances of the LyngbySmall models we notice an actual slowdown in the running time where for LyngbySmall-3 the number of reduced markings is less significant and it results in 13% slowdown, partially due to the overhead for computing stubborn sets but also because the search strategy changed and this resulted in the fact that we have to search a larger portion of the generated dependency graph before we obtain a conclusive answer. This effect is, in particular, observed in the LyngbySmall-4 example where the number of markings stored when applying partial order reduction actually exceeds those where no reduction is applied. Nevertheless, in general the experiments confirm the practical applicability of partial order reduction for 2-player games with only minimal overhead for computing the stubborn sets.

6 Conclusion

We generalised the partial order reduction technique based on stubborn sets from plain reachability to a game theoretical setting. This required a nontrivial extension of the classical conditions on stubborn sets so that a state space reduction can be achieved for both players in the game. In particular, the computation of the stubborn sets for player 2 (uncontrollable transitions) needed a new insight on how to handle and efficiently approximate the existence of infinite runs with player 2 transitions only. We proved the correctness of our approach and instantiated it to the case of Petri net games. We provided (to the best of our knowledge) the first implementation of partial order reduction for Petri net games and made it available as a part of the model checker TAPAAL. The experiments show promising results on a number of case studies, achieving in general a substantial state space reduction with only a small overhead for computing the stubborn sets. In the future work, we plan to combine our contribution with a recent insight on how to effectively use partial order reduction in the timed setting [4] in order to extend our framework to general timed games.

Acknowledgments. The research leading to these results has received funding from the project DiCyPS funded by the Innovation Fund Denmark and the ERC Advanced Grant LASSO. The last author is partially affiliated with FI MU, Brno, Czech Republic.

References

1 N. Alechina, N. Bulling, S. Demri, and B. Logan. On the complexity of resource-bounded logics. In *Reachability Problems*, volume 9899 of *LNCS*, pages 36–50. Springer-Verlag, 2016.

2 B. Bérard, S. Haddad, M. Sassolas, and N. Sznajder. Concurrent Games on VASS with Inhibition. In *International Conference on Concurrency Theory*, volume 7454 of *LNCS*, pages 39–52. Springer-Verlag, 2012.

3 F. M. Bønneland, P. G. Jensen, K. G. Larsen, M. Mũniz, and J. Srba. Artifact for "Partial Order Reduction for Reachability Games", June 2019. `doi:10.5281/zenodo.3252104`.

4 F.M. Bønneland, J. Dyhr, P.G. Jensen, M. Johannsen, and J. Srba. Start Pruning When Time Gets Urgent: Partial Order Reduction for Timed Systems. In *Computer Aided Verification*, volume 10981 of *LNCS*, pages 527–546. Springer-Verlag, 2018.

5 A.E. Dalsgaard, S. Enevoldsen, P. Fogh, L.S. Jensen, P.G. Jensen, T.S. Jepsen, I. Kaufmann, K.G. Larsen, S.M. Nielsen, M.Chr. Olesen, S. Pastva, and J. Srba. A Distributed Fixed-Point Algorithm for Extended Dependency Graphs. *Fundamenta Informaticae*, 161(4):351–381, 2018. `doi:10.3233/FI-2018-1707`.

6 A. David, L. Jacobsen, M. Jacobsen, K.Y. Jørgensen, M.H. Møller, and J. Srba. TAPAAL 2.0: Integrated Development Environment for Timed-Arc Petri Nets. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 7214 of *LNCS*, pages 492–497. Springer-Verlag, 2012.

7 R. Davidrajuh. Detecting Existence of Cycles in Petri Nets. In *International Joint Conference SOCO'16-CISIS'16-ICEUTE'16*, volume 527 of *AISC*, pages 376–385. Springer-Verlag, 2016.

8 J. Dehnert and A. Zimmermann. Making Workflow Models Sound Using Petri Net Controller Synthesis. In *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, volume 3290 of *LNCS*, pages 139–154. Springer-Verlag, 2004.

9 M. Huhn, P. Niebert, and H. Wehrheim. Partial Order Reductions for Bisimulation Checking. In *Foundations of Software Technology and Theoretical Computer Science*, volume 1530 of *LNCS*, pages 271–282. Springer-Verlag, 1998.

10 W. Jamroga, W. Penczek, P. Dembiński, and A. Mazurkiewicz. Towards Partial Order Reductions for Strategic Ability. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS 2018, pages 156–165, Richland, SC, 2018. ACM.

11 E. Javier. *Decidability and Complexity of Petri Net Problems — An Introduction*, volume 1491 of *LNCS*, pages 374–428. Springer-Verlag, 1998.

12 J.F Jensen, T. Nielsen, L.K. Oestergaard, and J. Srba. TAPAAL and Reachability Analysis of P/T Nets. In *Transactions on Petri Nets and Other Models of Concurrency XI*, volume 9930 of *LNCS*, pages 307–318. Springer-Verlag, 2016.

13 P. G. Jensen, K. G. Larsen, and J. Srba. PTrie: Data Structure for Compressing and Storing Sets via Prefix Sharing. In *Proceedings of the 14th International Colloquium on Theoretical Aspects of Computing (ICTAC'17)*, volume 10580 of *LNCS*, pages 248–265. Springer-Verlag, 2017. `doi:10.1007/978-3-319-67729-3_15`.

14 P.G. Jensen, K.G. Larsen, and J. Srba. Real-Time Strategy Synthesis for Timed-Arc Petri Net Games via Discretization. In *International Symposium on Model Checking Software*, volume 9641 of *LNCS*, pages 129–146. Springer-Verlag, 2016.

15 P.G. Jensen, K.G. Larsen, and J. Srba. Discrete and continuous strategies for timed-arc Petri net games. *International Journal on Software Tools for Technology Transfer*, 20(5):529–546, October 2018.

16 P. Kasting, M.R. Hansen, and S. Vester. Synthesis of Railway-Signaling Plans using Reachability Games. In *Proceedings of the 28th Symposium on the Implementation and Application of Functional Programming Languages*, IFL 2016, pages 9:1–9:13, New York, NY, USA, 2016. ACM. `doi:10.1145/3064899.3064908`.

17 A. Laarman and A. Wijs. Partial-Order Reduction for Multi-core LTL Model Checking. In *Hardware and Software: Verification and Testing*, volume 8855 of *LNCS*, pages 267–283. Springer-Verlag, 2014.

18 A. Lehmann, N. Lohmann, and K. Wolf. Stubborn Sets for Simple Linear Time Properties. In *Application and Theory of Petri Nets*, volume 7347 of *LNCS*, pages 228–247. Springer-Verlag, 2012.

19 D. Peled. All from One, One for All: on Model Checking Using Representatives. In *Computer Aided Verification*, volume 697 of *LNCS*, pages 409–423. Springer-Verlag, 1993.

20 F.G. Quintanilla, S. Kubler, O. Cardin, and P. Castagna. Product Specification in a Service-Oriented Holonic Manufacturing System using Petri-Nets. *IFAC Proceedings Volumes*, 46(7):342 – 347, May 2013. `doi:10.3182/20130522-3-BR-4036.00094`.

21 Y.S. Ramakrishna and S.A. Smolka. Partial-Order Reduction in the Weak Modal Mu-Calculus. In Antoni Mazurkiewicz and Józef Winkowski, editors, *International Conference on Concurrency Theory*, volume 1243 of *LNCS*, pages 5–24. Springer-Verlag, 1997.

22 R. Tagiew. Multi-Agent Petri-Games. In *International Conference on Computational Intelligence for Modeling Control Automation*, volume 10981 of *LNCS*, pages 130–135. IEEE Computer Society, 2008. `doi:10.1109/CIMCA.2008.15`.

23 A. Valmari. A Stubborn Attack on State Explosion. *Formal Methods in System Design*, 1(4):297–322, December 1992.

24 A. Valmari. On-The-Fly Verification with Stubborn Sets. In *Computer Aided Verification*, volume 697 of *LNCS*, pages 397–408. Springer-Verlag, 1993.

25 A. Valmari and H. Hansen. Stubborn Set Intuition Explained. In *Transactions on Petri Nets and Other Models of Concurrency XII*, volume 10470 of *LNCS*, pages 140–165. Springer-Verlag, 2017.

26 B. Willems and P. Wolper. Partial-Order Methods for Model Checking: From Linear Time to Branching Time. In *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, pages 294–303, July 1996. `doi:10.1109/LICS.1996.561357`.

A Proof of Lemma 7

Proof. (1): Due to A_1 and A_2 being finite as well as any G being deterministic we have every state $s \in \mathcal{S}$ is finitely branching. Since s is a winning state for player 1 in G every path leads to a goal state in a finite number of actions. Therefore the tree induced by all paths starting from s and where the leafs are the first occurring goal state is a finite tree, and n exists.

(2): Let n be the depth of σ at s in G and $\pi \in \Pi^{max}_{G,\sigma}(s)$ be any path subject to σ s.t. $\pi_0 \xrightarrow{a} \pi_1$ and $a \in next_\sigma(s)$. If $\pi_n \in Goal$ and for all $0 \leq j < n$ we have $\pi_j \notin Goal$ then the depth of σ at π_1 in G is $n - 1$. Otherwise there must exist some other run $\pi' \in \Pi^{max}_{G,\sigma}(s)$ s.t. $\pi'_1 = \pi_1$ and there exists $m \in \mathbb{N}^0$ s.t. $\pi'_m \in Goal$, $m < n$, and for all $0 \leq j < m$ we have $\pi_j \notin Goal$. Then $m - 1$ is the depth of σ at π_1 in G. ◀

B Proof of Lemma 14

Proof. Assume that there exists $w = a_1 \cdots a_n \in A^*$ s.t. $s \xrightarrow{w} s'$ and $s' \in Goal$. If $w \in \overline{St(s)}^*$ then by Condition **R** we must have $s' \notin Goal$, however this contradicts our assumption. Therefore there must exist an action that occurs in w that is in the stubborn set of s. Let $a_i \in St(s)$ be the first of such an action s.t. for all $1 \leq j < i$ we have $a_j \notin St(s)$. Clearly we have $a_1 \cdots a_j \in \overline{St(s)}^*$ and by Condition **W** we have $a_i \in St(s) \cap en(s)$. ◀

C Proof of Lemma 15

Proof. Assume that $s \in \mathcal{S}$ is a winning state for player 1 in G. By definition we have that there exists σ s.t. for all $\pi \in \Pi^{max}_{G,\sigma}(s)$ there exists a position i s.t. $\pi_i \in Goal$.

We prove by induction $IH(n)$: If s is a winning state for player 1 in G with a strategy with a depth of n then s is a winning state for player 1 in G_{St}.

Base step. Let $n = 0$. Then since n is the depth at s in G we must have $s \in Goal$, s is trivially a winning state for player 1 also in G_{St}, and the induction hypothesis holds.

Induction step. Let $n > 0$. There are three cases: (1) $en_1(s) \neq \emptyset$ and $en_2(s) \neq \emptyset$, (2) $en_2(s) = \emptyset$, and (3) $en_1(s) = \emptyset$. A deadlock at s, i.e. $en(s) = \emptyset$, cannot be the case otherwise we have $n = 0$.

Case (1): $en_1(s) \neq \emptyset$ and $en_2(s) \neq \emptyset$. Assume s is a winning state for player 1 in G with a strategy σ with a depth of n. We want to show there exists a strategy σ' s.t. s is a winning state for player 1 in G_{St} with σ'. Since s is a winning state for player 1 in G with σ if $s \xrightarrow{a} s'$ where $a \in next_\sigma(s)$ then s' is a winning state for player 1 in G with $m < n$ as the depth of σ at s' in G, following property 2 in Lemma 7. By the induction hypothesis s' is a winning state for player 1 in G_{St}. By Condition **I** we know $en_1(s) \subseteq St(s)$ implying that $\sigma(s) \in St(s)$. Player 1 can therefore choose the same action proposed in the original game s.t. $\sigma'(s) = \sigma(s)$ and s is a winning state for player 1 in G_{St}.

Case (2): $en_2(s) = \emptyset$. Assume s is a winning state for player 1 in G with a strategy σ with a depth of n. We want to show there exists a strategy σ' s.t. s is a winning state for player 1 in G_{St} with σ'. Let $\pi \in \Pi^{max}_{G,\sigma}(s)$ be any run and $\pi_0 = s$. Since s is a winning state for player 1 in G with σ we know there exists a $m \leq n$ s.t. $\pi_0 \xrightarrow{a_1} \pi_1 \xrightarrow{a_2} \cdots \xrightarrow{a_m} \pi_m$ and $\pi_m \in Goal$. Let $w = a_1 \cdots a_m$. We start by showing that there exists $1 \leq i \leq m$ s.t. $a_i \in St(s)$. Assume that $w \in \overline{St(s)}^*$ is true. Then we have $\pi_m \notin Goal$ due to Condition **R**, a contradiction. Therefore there must exist $1 \leq i \leq m$ s.t. $a_i \in St(s)$. Let i be minimal in the sense that for all $1 \leq j < i$ we have $a_j \notin St(s)$. We can then divide w s.t. $w = va_iu$,

$v \in \overline{St(s)}^*$ and we have $s \xrightarrow{a_i} s'_0 \xrightarrow{v} \pi_i \xrightarrow{u} \pi_m$ due to Condition **W** as well as $s \xrightarrow[St]{a_i} s'_0$. There are two subcases: (2.1) $a_i \in \mathit{safe}(s)$ and (2.2) $a_i \notin \mathit{safe}(s)$.

- Case (2.1): $a_i \in \mathit{safe}(s)$: For all $1 \leq j < i$ we have $en_2(\pi_j) = \emptyset$ due to i being minimal and Condition **G1**. From that, if $a_i \in \mathit{safe}(s)$ then for all all intermediate states in $s \xrightarrow{a_i v} \pi_i$ we only have player 1 states otherwise a_i is not a safe action due to the definition of safe actions. We have that s'_0 is a player 1 state and there exists the sequence $v = a'_1 a'_2 \cdots a'_{i-1}$ s.t. $s'_0 \xrightarrow{a'_1} s'_1 \xrightarrow{a'_2} \cdots \xrightarrow{a'_{i-1}} \pi_i$ and for all $1 \leq k < i-1$ we have $en_2(s'_k) = \emptyset$. Let σ' be defined such that for all $0 \leq j < i-1$ we have $\sigma(s'_{j-1}) = a'_j$, and let σ' from π_i be defined as σ. Due to property 2 in Lemma 7 the depth of σ at π_i in G is at most $k \leq n-i$. Since G is deterministic by following the strategy σ'_1 from s'_0 we always reach π_i in $i-1$ actions. From this we can infer that the depth of σ'_1 at s'_0 in G is at most $k+i-1$ which is clearly smaller than n. Therefore s'_0 is a winning state for player 1 in G with at most $k+i-1$ as the depth of σ' at s'_0 in G. By the induction hypothesis s'_0 is a winning state for player 1 in G_{St}. Player 1 can then choose a_i in the reduced game such that $\sigma'(s) = a_i$ and s is a winning state for player 1 in G_{St}.
- Case (2.2): $a_i \notin \mathit{safe}(s)$: Since $a_i \notin \mathit{safe}(s)$ we have $St(s) \cap en_1(s) \nsubseteq \mathit{safe}(s)$ and $en_1(s) \subseteq St(s)$ by Condition **S**. If $s \xrightarrow{\sigma(s)} s'$ then s' is a winning state for player 1 in G with $m < n$ as the depth of σ at s' in G, following property 2 in Lemma 7. By the induction hypothesis s' is a winning state for player 1 in G_{St}. Player 1 can choose the same action proposed in the original game such that $\sigma'(s) = \sigma(s)$ and s is a winning state for player 1 in G_{St}.

Case (3): $en_1(s) = \emptyset$. Assume s is a winning state for player 1 in G with σ as the winning strategy. We want to show there exists a strategy σ' s.t. s is a winning state for player 1 in G_{St} with σ'. Since $en_1(s) = \emptyset$ we have $\sigma(s) = \sigma'(s) = \perp$ by the definition of strategies. We have from the definition of a winning strategy no matter what action player 2 chooses the resulting state is a winning state for player 1. What remains to be shown is that at least one enabled player 2 action is included in $St(s)$. From the case we have $en_2(s) \neq \emptyset$ and due to Condition **D** we have that there exists $a \in en_2(s) \cap St(s)$ s.t. $s \xrightarrow[St]{a} s'$, and we are done. ◀

D Proof of Lemma 16

Proof. Assume that $s \in \mathcal{S}$ is a winning state for player 1 in G_{St}. By definition we have that there exists σ s.t. for all $\pi \in \Pi^{max}_{G_{St},\sigma}(s)$ there exists a position i s.t. $\pi_i \in \mathit{Goal}$. Let σ be fixed for the remainder of the proof. Let n be the depth of σ at s in G_{St}.

We proceed by induction on n for the induction hypothesis $\mathit{IH}(n)$: If s is a winning state for player 1 in G_{St} with a strategy with a depth of n then s is a winning state for player 1 in G.

Base step. If $n = 0$ then since n is the depth at s in G_{St} we must have $s \in \mathit{Goal}$, s is trivially a winning state for player 1 also in G, and the induction hypothesis holds.

Induction step. Let $n > 0$. There are three cases: (1) $en_1(s) \cap St(s) \neq \emptyset$ and $en_2(s) \cap St(s) \neq \emptyset$, (2) $en_2(s) \cap St(s) = \emptyset$, and (3) $en_1(s) \cap St(s) = \emptyset$. A deadlock at s, i.e. $en(s) = \emptyset$, cannot be the case otherwise we have $n = 0$. A deadlock in G_{St} at, i.e. $en(s) \cap St(s) = \emptyset$ s cannot be the case either otherwise we have $n = 0$.

Case (1): $en_1(s) \cap St(s) \neq \emptyset$ and $en_2(s) \cap St(s) \neq \emptyset$. Assume s is a winning state for player 1 in G_{St} with a strategy σ with a depth of n. We want to show there exists a strategy σ' s.t. s is a winning state for player 1 in G with σ'. Since s is a winning state for player 1 in G_{St} with σ if $s \xrightarrow[St]{\sigma(s)} s'$ or $s \xrightarrow[St]{a} s'$ where $a \in en_2(s) \cap St(s)$ then s' is a winning state for player 1 in G_{St} with $m < n$ as the depth of σ at s' in G_{St}, following property 2 in Lemma 7. By the induction hypothesis s' is a winning state for player 1 in G. Since s is a mixed state in G_{St} then s must also be a mixed state in G due to $\xrightarrow[St]{}\subseteq \rightarrow$ from the definition of a reduced game and so $s \xrightarrow{\sigma(s)} s'$. Therefore player 1 can choose the same action proposed in the reduced game such that $\sigma'(s) = \sigma(s)$. Furthermore we have $en_2(s) \cap St(s) = en_2(s)$ fron Condition **I**. From this we can conclude that s is a winning state for player 1 in G with strategy σ'.

Case (2): $en_2(s) \cap St(s) = \emptyset$. Assume s is a winning state for player 1 in G_{St} with a strategy σ with a depth of n. We want to show there exists a strategy σ' s.t. s is a winning state for player 1 in G with σ'. Since s is a winning state for player 1 in G_{St} with σ we have $s \xrightarrow[St]{\sigma(s)} s'$ and s' is a winning state for player 1 in G_{St} with $m < n$ as the depth of σ at s' in G_{St}, following property 2 in Lemma 7. By the induction hypothesis s' is a winning state for player 1 in G. Trivially we have that $s \xrightarrow{\sigma(s)} s'$ since we have $\xrightarrow[St]{}\subseteq \rightarrow$ from the definition of a reduced game. Therefore player 1 can choose the same action proposed in the reduced game $\sigma'(s) = \sigma(s)$. Next we want to show by contradiction that s is a player 1 state in G. Assume $en_2(s) \neq \emptyset$, i.e. that s is a mixed state in G. From this we can infer by Condition **I** that $en(s) \subseteq St(s)$ and $en_2(s) \cap St(s) \neq \emptyset$, which is a contradiction. Therefore we have $en_2(s) = \emptyset$, i.e. s is a player 1 state also in G, and s is a winning state for player 1 in G with strategy σ'.

Case (3): $en_1(s) \cap St(s) = \emptyset$. Assume s is a winning state for player 1 in G_{St} with a strategy σ with a depth of n. We want to show there exists a strategy σ' s.t. s is a winning state for player 1 in G with σ'. Since $en_1(s) \cap St(s) = \emptyset$ then we have $\sigma(s) = \bot$. Furthermore, we have $en_1(s) = \emptyset$ since otherwise with Condition **I** we will be able to infer that $en_1(s) \cap St(s) \neq \emptyset$, which is a contradiction. Therefore, in any case we define $\sigma'(s) = \bot$ due to the definition of a strategy. Since s is a winning state for player 1 in G_{St} with σ if $s \xrightarrow[St]{a} s'$ where $a \in en_2(s) \cap St(s)$ then s' is a winning state for player 1 in G_{St} with $m < n$ as the depth of σ at s' in G_{St}, following property 2 in Lemma 7. By the induction hypothesis s' is a winning state for player 1 in G.

What remains to be shown is that for all $a \in en_2(s) \setminus St(s)$ where $s \xrightarrow{a} s'$ that s' is a winning state for player 1 in G. We show by contradiction that this is the case. Assume there exists $\pi \in \Pi_G^{max}(s)$ s.t. $\pi_0 \xrightarrow{a_1} \pi_1$ and $a_1 \in en_2(s) \setminus St(s)$ and for all $i > 0$ we have $\pi_i \notin X$ and $\pi_1 \xrightarrow{a_2} \pi_2 \xrightarrow{a_3} \cdots$. There are two subcases: (3.1) for all $i > 0$ we have $a_i \in A_2$ and (3.2) there exists a minimal $i > 1$ s.t. $a_i \in A_1$.

- Case (3.1): For all $i > 0$ we have $a_i \in A_2$: In this case we have another two subcases: (3.1.1) $\ell(\pi) = \infty$ and (3.1.2) $en(\pi_{\ell(\pi)}) = \emptyset$.
  - Case (3.1.1): $\ell(\pi) = \infty$: In this case we have that there exists $w \in A_2^\omega$ s.t. $s \xrightarrow{w}$, namely $w = a_1 a_2 \cdots$. Let a_{m+1} be the first action to occur in w that also occurs infinitely often in w s.t. $w = v a_{m+1} u$, $v = a_1 \cdots a_m$, and $s \xrightarrow{v} s_m \xrightarrow{a_{m+1}u}$. Since the set of labels A is finite such an action must exist. We proceed by induction on the length m to show that a path to s_m is preserved with the induction hypothesis $IH(m)$: If there exists $w \in A_2^*$ s.t. $|w| = m$ and $s \xrightarrow{w} s_m$ then there exists $w' \in A_2^*$ s.t. $s \xrightarrow[St]{w'} s_m$. *Base step.* Trivial for $m = 0$. *Induction step.* Let $m > 0$. We have already assumed that s_m can be reached from s in m actions using only player 2 actions in G with the sequence v. We want to show by contradiction that there exists a minimal $1 \leq j \leq m$ s.t. $a_j \in St(s)$. By Condition **C** we have $a_{m+1} \in St(s)$. If $a_{m+1} \notin en(s)$

and $v \in \overline{St(s)}^*$ then this would contradict Condition **W**. Therefore, there must exist $1 \leq j \leq m$ s.t. $a_j \in St(s)$. From this we can divide v s.t. $v = v'a_ju'$ and $v' \in \overline{St(s)}^*$. Then due to Condition **W** we have $s \xrightarrow{a_j} s'' \xrightarrow{v'u'} s_m$, $|v'u'| = m-1$, and $s \xrightarrow[St]{a_j} s''$. By the inner induction hypothesis we have there exists $w' \in A_2^*$ s.t. $s'' \xrightarrow[St]{w'} s_m$ and $|w'| = m-1$. With $s \xrightarrow[St]{a_j} s''$ we have $s \xrightarrow{a_jw'} s_m$ and a path to s_m is preserved. At s_m we have $a_{m+1} \in St(s_m)$ by Condition **C**, and the same argument can be applied for each subsequent state and action in w, and an infinite path of player 2 actions is preserved. Therefore, s is a losing state for player 1 However this is a contradiction since s is a winning state for player 1 in G_{St}. Therefore we have that π does not exist.

- Case (3.1.2) $en(\pi_{\ell(\pi)}) = \emptyset$: Let $w = a_1 \cdots a_{\ell(\pi)}$. To show the contradiction we will proceed by induction on the length $m = |w|$ with the induction hypothesis $IH(m)$: If there exists $w' = a'_1 \cdots a'_m \in A_2^*$ s.t. $s \xrightarrow{a'_1} s_1 \xrightarrow{a'_2} \cdots \xrightarrow{a'_m} s'$, $|w'| = m$, $en(s') = \emptyset$, and for all $0 \leq k \leq m$ we have $\pi_k \notin \mathit{Goal}$ then s is a winning state for player 2 in G_{St}. *Base step.* Trivial for $m = 0$. *Induction step.* Let $m > 0$. We have already assumed that a deadlock can be reached from s in m actions using only player 2 actions in G with the path π. We want to show by contradiction that there exists a minimal $1 \leq j \leq \ell(\pi)$ s.t. $a_j \in St(s)$. Now assume that $w \in \overline{St(s)}^*$. From Condition **D** we have that there exists $a \in en_2(s) \cap St(s)$ s.t. for all $w \in \overline{St(s)}^*$ where $s \xrightarrow{w} s''$ we have $a \in en_2(s'')$. From this we have $a \in en_2(\pi_{\ell(\pi)})$, which is a contradiction. Therefore there must exist a minimal $1 \leq j \leq \ell(\pi)$ s.t. $a_j \in St(s)$. From this we can divide w s.t. $w = va_ju$ and $v \in \overline{St(s)}^*$. Then due to Condition **W** we have $s \xrightarrow{a_j} s'' \xrightarrow{vu} \pi_{\ell(\pi)}$, $|vu| = m-1$, and $s \xrightarrow[St]{a_j} s''$. We have $s'' \notin \mathit{Goal}$ since otherwise we would have $a_1 \in St(s)$ by Condition **R**, which is a contradiction. By the inner induction hypothesis s'' is a winning state for player 2 in G_{St}. We already showed that $a_j \in St(s)$ s.t. $s \xrightarrow[St(s)]{a_j} s''$. We therefore have that s is a winning state for player 2 in G_{St} since the path π is preserved. However this is a contradiction since s is a winning state for player 1 in G_{St}. Therefore we have that π does not exist.

- Case (3.2): There exists a minimal $i > 1$ s.t. $a_i \in A_1$: Let $w = a_1 \cdots a_{i-1}$. To show the contradiction we will proceed by induction on $m = i-1$ with the induction hypothesis $IH(m)$: If there exists $w' \in A_2^*$ s.t. $s \xrightarrow{w'}, \pi_{i-1}$, $|w'| = m$, $en(s) = \emptyset$, and for all $0 \leq k \leq m$ we have $\pi_k \notin \mathit{Goal}$ then s is a winning state for player 2 in G_{St}. *Base step.* Trivial for $m = 0$. *Induction step.* Let $m > 0$. We have already assumed that π_{i-1} can be reached from s in m steps using only player 2 actions in G with the path π. We want to show by contradiction that there exists a minimal $1 \leq j < i$ s.t. $a_j \in St(s)$. Now assume that $w \in \overline{St(s)}^*$. Then due to Condition **G2** we have $en_1(\pi_{i-1}) = \emptyset$, which is a contradiction. Therefore there must exist a minimal $1 \leq j < i$ s.t. $a_j \in St(s)$. From this we can divide w s.t. $w = va_ju$ and $v \in \overline{St(s)}^*$. Then due to Condition **W** we have $s \xrightarrow{a_j} s'' \xrightarrow{vu} \pi_{i-1}$, $|vu| = m-1$, and $s \xrightarrow[St]{a_j} s''$. We have $s'' \notin \mathit{Goal}$ since otherwise we would have $a_1 \in St(s)$ by Condition **R**, which is a contradiction. By the induction hypothesis s'' is a winning state for player 2 in G_{St}. We already showed that $a_j \in St(s)$ s.t. $s \xrightarrow[St(s)]{a_j} s''$. We therefore have that s is a winning state for player 2 in G_{St} since the path π is preserved. However this is a contradiction since s is a winning state for player 1 in G_{St}. Therefore we have that π does not exist.

We have showed in every case that π cannot exist. Therefore for all $a \in en_2(s) \setminus St(s)$ where

$s \xrightarrow{a} s'$ we have that s' is a winning state for player 1 in G.

Finally, since we have showed that for all $a \in en_2(s)$ where $s \xrightarrow{a} s'$ we have s' is a winning state for player 1 in G, then we can conclude that s is a winning state for player 1 in G. ◀

E Proof of Theorem 17

Proof. Follows from Lemma 15 and 16. ◀

F Proof of Theorem 18

Proof. The proof of Theorem 18 follows the same approach as in Lemma 15 and 16, with only Case 2 from Lemma 15 receiving changes that makes the case easier.

When the first action in the reduction $a_i \in St(s)$ is swapped by Condition **W**, s.t. $s \xrightarrow{a_j} s' \xrightarrow{v} a_i$, we know s' is a player 1 state (as well as every intermediate state between s' and s_m) otherwise $s' \xrightarrow{v} s_m$ is not possible since $v \in A_1^*$. Therefore, whether a_i is safe or not is not relevant, and s' is a winning state for player 1 in G. ◀

G Proof of Lemma 20

Proof. Assume $t^+ \cap {}^\bullet T_2 = \emptyset$ and ${}^-t \cap {}^\circ T_2 = \emptyset$. We prove directly that t is safe in M. Let $w \in (T_1 \setminus \{t\})^*$ s.t. $M \xrightarrow{w} M'$, $en_2(M') = \emptyset$, and $M \xrightarrow{tw} M''$. The only difference between M' and M'' is that t is fired first and we have $M''(p') = M'(p') + W((t,p')) - W((p',t))$ for all $p \in P$. Then for all $t' \in T_2$ we have that there either exists $p \in {}^\bullet t'$ s.t. $M'(p) < W((p,t'))$, or there exists $p' \in {}^\circ t'$ s.t. $M'(p') \geq I((p',t'))$. In the first case, since $t^+ \cap {}^\bullet T_2 = \emptyset$, we must have $W((t,p)) \leq W((p,t))$ which implies $M''(p) \leq M'(p)$ and $t' \notin en(M'')$. In the second case, since ${}^-t \cap {}^\circ T_2 = \emptyset$, we must have $W((p',t)) \geq W((t,p'))$, which implies $M''(p') \geq M'(p')$ and $t' \notin en(M'')$. Therefore t is safe in M. ◀

H Proof of Lemma 22

Proof. Assume that $M \not\models \varphi$. The proof proceeds by structural induction on φ. All cases, with the exception of $\varphi_1 \wedge \varphi_2$, are proved in Lemma 2 presented in [4].

$\varphi = \varphi_1 \wedge \varphi_2$: There are five subcases defined by Equation 1: (1) $M \models \varphi_2$, (2) $M \models \varphi_1$, (3) $M \not\models \varphi_1$ and $A_M(\varphi_1) \subseteq \mathit{safe}(M)$, (4) $M \not\models \varphi_2$ and $A_M(\varphi_2) \subseteq \mathit{safe}(M)$, and (5) the default cases.

- Case (1): $M \models \varphi_2$: Since we have $M \not\models \varphi$ and $M \models \varphi_2$ we must therefore have that $M \not\models \varphi_1$ by the semantics of φ. By Equation 1, since $M \models \varphi_2$, we have $A_M(\varphi_1 \wedge \varphi_2) = A_M(\varphi_1)$. By the induction hypothesis this implies $M' \not\models \varphi_1$, and from this and the semantics of φ we have $M' \not\models \varphi$.
- Case (2): $M \models \varphi_1$: This case is symmetric to Case (1) and follows the same approach.
- Case (3): $M \not\models \varphi_1$ and $A_M(\varphi_1) \subseteq \mathit{safe}(M)$: By Equation 1 we have $A_M(\varphi_1 \wedge \varphi_2) = A_M(\varphi_1)$. By the induction hypothesis this implies $M' \not\models \varphi_1$, and from this and the semantics of φ we have $M' \not\models \varphi$.
- Case (4): $M \not\models \varphi_2$ and $A_M(\varphi_2) \subseteq \mathit{safe}(M)$: This case is symmetric to Case (3) and follows the same approach.
- Case (5): Default: Since we have $M \not\models \varphi$ we know there exists $i \in \{1,2\}$ s.t. $M \not\models \varphi_i$ by the semantics of φ. By Equation 1 we have $A_M(\varphi_1 \wedge \varphi_2) = A_M(\varphi_1) \cup A_M(\varphi_2)$. By

the induction hypothesis this implies $M' \not\models \varphi_i$, and from this and the semantics of φ we have $M' \not\models \varphi$.

◀

I Proof of Lemma 23

Proof. *Termination*: Since T is finite the loop in line 2 can iterate a finite number of times and it terminates. It is clear that the sets *MarkedPlaces* and *FreshPlaces* are only increasing in the while-loop in line 6, and terminates if they ever become equal. Since these sets are only increasing, and P is finite, the while-loop can iterate at most finite times and it terminates. Since T is finite the loop in line 11 can iterate a finite number of times and it terminates.

Correctness: The trivial case where there exists $t \in T_2$ s.t. ${}^\bullet t = \emptyset$ is handled in line 2.

Assume that there exists $w \in T_2^\omega$ s.t. $M \xrightarrow{w}$ and there exists $t \in T_2$ that occurs infinitely often in w. If w is fireable in M with inhibitor arcs then it is also fireable in M without inhibitor arcs, i.e. removing inhibitor arcs does not remove paths. As an overapproximation we assume that all inhibitor arcs in the algorithm have a weight of ∞.

It follows from that t occurs infinitely often in w that $v, u \in T^*$ s.t. $M \xrightarrow{v} M' \xrightarrow{tu} M''$, vtu is a prefix of w, and for all $p \in {}^\bullet t$ we have $M'(p) \leq M''(p)$, otherwise p will eventually lack a sufficient number of tokens.

First we want to show that $t \in \mathit{Inf}$. Assume for the sake of contradiction that $t \in \mathit{Fin}$. Then there exists $p \in {}^\bullet t$ s.t. $p \in \mathit{Fin}$ and t cannot occur infinitely often in w since p cannot provide enough tokens, a contradiction. Therefore, we have ${}^\bullet t \subseteq \mathit{Inf}$, $t \notin \mathit{Fin}$, and $t \in \mathit{Inf}$.

Next we want to show that ${}^\bullet t \subseteq \mathit{MarkedPlaces}$. Let $P_0 = \{p \in P \mid M(p) > 0\}$ and for all $i \geq 1$ we have $P_i = \{p \in P \mid p \in P_{i-1} \vee \exists t \in T_2.\ p \in {}^\bullet t \wedge {}^\bullet t \subseteq P_{i-1}\}$. For all $j \geq 1$ we have $P_{j-1} \subseteq P_j$. Because P is finite we have that there exists $i \geq 1$ s.t. for all $i < j$ we have $P_i = P_j$, and $P_i = \mathit{MarkedPlaces}$ after the loop in line 6 terminates. Intuitively, P_i is an overapproximation of all places that can be marked by 1 or more tokens at any intermediate marking by firing player 2 transitions from M. For all $p \in {}^\bullet t$ if $M(p) > 0$ then $p \in P_0$ and also $p \in P_i$. If $M(p) < 1$ then we can infer that there exists $t' \in T_2$ that occurs in v and $p \in t'^\bullet$, otherwise we would not have $t \in en(M')$. Therefore we must have ${}^\bullet t \subseteq P_i$ since P_i is an overapproximation.

We have shown $t \in \mathit{Inf}$ and ${}^\bullet t \subseteq \mathit{MarkedPlaces}$, and from line 13 we have $t \in cycle(N, M)$.

◀

J Proof of Theorem 24

Proof. We shall argue that any reduction St satisfying the conditions of the theorem also satisfies the **I**, **W**, **R**, **G1**, **G2**, **S**, **C**, and **D** conditions.

- (**I**): Follows from Condition 1.
- (**W**): Let $M, M' \in \mathcal{M}(N)$ be markings, $t \in St(M)$, and $w \in \overline{St(M)}^*$. We will show that if $M \xrightarrow{wt} M'$ then $M \xrightarrow{tw} M'$.

  Let $M_w \in \mathcal{M}(N)$ be a marking s.t. $M \xrightarrow{w} M_w$. Assume for the sake of contradiction that $t \notin en(M)$. Then t is disabled in M there is $p \in {}^\bullet t$ such that $M(p) < W((p,t))$ or there is $p \in {}^\circ t$ such that $M(p) \geq I((p,t))$. In the first case, due to Condition 7a all the transitions that can add tokens to p are included in $St(M)$. Since $w \in \overline{St(M)}^*$ this implies that $M_w(p) < W((p,t))$ and $t \notin en(M_w)$ contradicting our assumption that $M_w \xrightarrow{t} M'$. In the first case, due to Condition 7b all the transitions that can remove tokens from p are included in $St(M)$. Since $w \in \overline{St(M)}^*$ this implies that $M_w(p) \geq I((p,t))$ and

$t \notin en(M_w)$ contradicting our assumption that $M_w \xrightarrow{t} M'$. Therefore we must have that $t \in en(M)$.
Since $t \in en(M)$ there is $M_t \in \mathcal{M}(N)$ s.t. $M \xrightarrow{t} M_t$. We have to show that $M_t \xrightarrow{w} M'$ is possible. For the sake of contradiction, assume that this is not the case. Then there must exist a transition t' that occurs in w that became disabled because t was fired. There are two cases: t removed one or more tokens from a shared pre-place $p \in {}^{-}t \cap {}^{\bullet}t'$ or added one or more tokens to a place $p \in t^{+} \cap {}^{\circ}t'$. In the first case, due to Condition 8a all the transitions that can remove tokens from p are included in $St(M)$, implying that $t' \in St(M)$. Since $w \in \overline{St(M)}^{*}$ such a t' cannot exist. In the second case, due to Condition 8b all the transitions that can add tokens to p are included in $St(M)$, implying that $t' \in St(M)$. Since $w \in \overline{St(M)}^{*}$ such a t' cannot exist. Therefore we must have that $M_t \xrightarrow{w} M'$ and we can conclude with $M \xrightarrow{tw} M'$.

- (**R**): Follows from Condition 3 and Lemma 22.
- (**G1**): Let $M \in \mathcal{M}(N)$ be a marking and $w \in \overline{St(M)}^{*}$ s.t. $M \xrightarrow{w} M'$. We will show that if $en_2(M) = \emptyset$ then $en_2(M') = \emptyset$.
  Assume that $en_2(M) = \emptyset$. Then by Condition 5 we have $T_2 \subseteq St(M)$. Let $t \in T_2$ be a player 2 transition. By Condition 7 we know that either: There exists $p \in {}^{\bullet}t$ s.t. $M(p) < W((p,t))$ and ${}^{+}p \subseteq St(s)$, or there exists $p \in {}^{\circ}t$ s.t. $M(p) \geq I((p,t))$ and $p^{-} \subseteq St(s)$. In the first case, in order to enable t at least one transition from ${}^{+}p$ has to be fired. However, we know ${}^{+}p \subseteq St(s)$ is true, and therefore none of the transitions in ${}^{+}p$ can occur in w, which implies $t \notin en_2(M')$. In the second case, in order to enable t at least one transition from p^{-} has to be fired. However, we know $p^{-} \subseteq St(s)$ is true, and therefore none of the transitions in p^{-} can occur in w, which implies $t \notin en_2(M')$. These two cases together implies $en_2(M') = \emptyset$.
- (**G2**): Follows the same approach as **G1**.
- (**S**): Follows from Condition 2.
- (**C**): Follows from Condition 6 and Lemma 23.
- (**D**): Let $M \in \mathcal{M}(N)$ be a marking and $w \in \overline{St(M)}^{*}$ s.t. $M \xrightarrow{w} M'$. We will show that if $en_2(M) \neq \emptyset$ then there exists $t \in en_2(M) \cup St(M)$ s.t. $t \in en_2(M')$.
  Assume that $en_2(M) \neq \emptyset$. From Condition 9 we know that there exists $t \in en_2(M) \cup St(M)$ s.t. $({}^{\bullet}t)^{-} \cup {}^{+}({}^{\circ}t) \subseteq St(M)$. Assume for the sake of contradiction that $t \notin en_2(M')$. In this case there must either exist $p \in {}^{\bullet}t$ s.t. $M'(p) < W((p,t))$, or there exists $p \in {}^{\circ}t$ s.t. $M'(p) \geq I((p,t))$. In the first case, since $t \in en_2(M)$ we have that $M(p) \geq W((p,t))$. Therefore at least one transition from p^{-} has to have been fired. However, we know $({}^{\bullet}t)^{-} \subseteq St(M)$ is true, and therefore none of the transitions in p^{-} can occur in w, which implies $M'(p) \geq W((p,t))$, a contradiction. In the second case, since $t \in en_2(M)$ we have have that $M(p) < I((p,t))$. Therefore at least one transition from ${}^{+}p$ has to have been fired. However, we know ${}^{+}({}^{\bullet}t) \subseteq St(M)$ is true, and therefore none of the transitions in ${}^{+}p$ can occur in w, which implies $M'(p) < I((p,t))$, a contradiction. Therefore $t \notin en_2(M')$ cannot be true, and we must have that $t \in en_2(M')$.

This completes the proof of the theorem. ◀

K Proof of Theorem 25

Proof. *Termination*: If $en_1(M) \neq \emptyset$ and $en_2(M) \neq \emptyset$ then we terminate in line 4. Otherwise $Y \neq \emptyset$ and we enter the while-loop in Algorithm 3. Notice that $X \cap Y = \emptyset$ is always the case in the execution of Algorithm 3. We never remove transitions from X after they have been added. Therefore, since in line 13 of Algorithm 3 a new transition is added to X at the end of each loop iteration, the loop can iterate at most once for each transition. Since T is finite by the Petri Net Game definition, the loop iterates a finite number of times, and Algorithm 3 terminates. If $\mathit{en}_1(M) \cap X \nsubseteq \mathit{safe}(M)$ then we terminate in line 15 of Algorithm 2, and otherwise we return in line 15 and Algorithm 2 terminates.

Correctness: It was shown that the construction in Theorem 24 results in a set that is a stubborn set of a stable reduction. It is therefore sufficient to show that Algorithm 2 replicates the construction. Notice that every transition that is added to Y is eventually added to X in line 13 and returned in line 15 of Algorithm 3. Let $t \in Y$.

Condition 1: If $\mathit{en}_1(M) \neq \emptyset$ and $\mathit{en}_2(M) \neq \emptyset$ then we return T in line 4 of Algorithm 2.

Condition 2: If $\mathit{en}_1(M) \cap \mathit{St}(M) \nsubseteq \mathit{safe}(M)$ then we return T in line 15 of Algorithm 2.

Condition 3: We have $A_M(\varphi) \subseteq Y$ in line 12 of Algorithm 2.

Condition 4: We have $T_1 \subseteq Y$ in line 8 of Algorithm 2.

Condition 5: We have $T_2 \subseteq Y$ in line 11 of Algorithm 2.

Condition 6: We have $\mathit{cycle}(N, M) \subseteq Y$ in line 9 of Algorithm 2.

Condition 7a: In line 6 we pick any $p \in {}^\bullet t$ s.t. $M(p) < W((p,t))$, and in line 7 of Algorithm 3 we add ${}^+p$ to Y.

Condition 7b: In line 9 we pick any $p \in {}^\circ t$ s.t. $M(p) \geq I((p,t))$, and in line 10 of Algorithm 3 we add p^- to Y.

Condition 8a: In line 12 of Algorithm 3 we add $({}^-t)^\bullet$ to Y.

Condition 8b: In line 12 of Algorithm 3 we add $(t^+)^\circ$ to Y.

Condition 9: In line 7 of Algorithm 2 we pick any $t' \in \mathit{en}_2(M)$ and in line 8 we add $({}^\bullet t)^- \cup {}^+({}^\circ t)$ to Y. ◀