Logical Methods in Computer Science
Volume 17, Issue 1, 2021, pp. 21:1–21:26
https://lmcs.episciences.org/

Submitted Dec. 23, 2019
Published Mar. 18, 2021

STUBBORN SET REDUCTION FOR TWO-PLAYER REACHABILITY GAMES

FREDERIK MEYER BØNNELAND, PETER GJØL JENSEN, KIM GULDSTRAND LARSEN, MARCO MUÑIZ, AND JIŘÍ SRBA

Department of Computer Science, Aalborg University, Denmark
e-mail address: {frederikb,pgj,kgl,muniz,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 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.

1. Introduction

The state space explosion problem is the main obstacle for model checking of concurrent systems. Even simple processes running in parallel can produce an exponentially large number of interleavings, making full state space search practically intractable. A family of methods for taming this problem is that of partial order reductions [God96, Pel93, Val91] by exploiting the commutativity of independent concurrent processes. Variants of partial order reductions include persistent sets [God96, God90, GW93], ample sets [Pel93, Pel96, Pel98], and stubborn sets [Val91, Val92, Val93, VH17].

As our main contribution, we generalise the theory of the stubborn set variant of partial order reductions into the setting of 2-player games. We exploit the observation that either of the two players often is left with no actions to propose, leaving the opponent to independently dictate the behavior of the system for a limited, consecutive sequence of actions. In such cases we may apply the classical stubborn set reductions in order to reduce the number of interleavings of independent actions. To preserve the winning strategies of both players, a number of conditions of the reduction has to be satisfied. We define the notion of a *stable* stubborn set reduction by a set of sufficient conditions that guarantee the preservation of winning strategies for both players. Furthermore, we formally prove the correctness of stable reductions in the setting of general game labelled transition systems, and instantiate our

Key words and phrases: Petri nets, games, synthesis, partial order reduction, stubborn sets.

LOGICAL METHODS IN COMPUTER SCIENCE DOI:10.23638/LMCS-17(1:21)2021

framework to weighted Petri net games with inhibitor arcs. We propose approximate syntax-driven conditions of a stable Petri net game reduction satisfying the sufficient conditions for our stable reductions and demonstrate their applicability in an efficient, open source implementation in the model checker TAPAAL [DJJ+12]. Our implementation is based on dependency graphs, following the approach from [DEF+18, JLS16], and we demonstrate on several case studies that the computation of the stubborn sets only has a minor overhead while having the potential of achieving exponential reduction both in the running time as well as in the number of searched configurations. To the best of our knowledge, this is the first efficient implementation of a 2-player game partial order reduction technique for Petri net games.

Related Work. Partial order reductions in the non-game setting for linear time properties have previously been studied [LW14, LLW12, Pel93, Val92] which lends itself towards the safety or liveness properties we wish to preserve for winning states. Originally Peled and Valmari presented partial order reductions for general stuttering-insensitive LTL [Pel93, Val92] and Lehmann et al. subsequently studied stubborn sets applied to a subset of LTL properties, called simple linear time properties, allowing them to utilise a relaxed set of conditions compared to those for general LTL preservation [LLW12].

The extension of partial order reductions to game-oriented formalisms and verification tasks has not yet received much attention in the literature. In [JPDM18] 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 consider games with imperfect information, however, they also show that their technique is inapplicable for perfect information games. In our work, 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 equivalence between two labelled transition systems is presented in [Val97, HNW98, GKPP99]. 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 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 weak modal μ-calculus and CTL (see e.g. [RS97, WW96]) allows us in principle to encode the game semantics as a part of a μ-calculus formula. Recently, partial order reduction techniques for parity games have been proposed by Neele et al. [NWW20], which allows for model checking the full modal μ-calculus. However, the use of more general partial order reduction methods may waste reduction potential, as the more general methods usually generate larger stubborn sets to preserve properties that are not required in the 2-player game setting.

Complexity and decidability results for control synthesis in Petri net games are not encouraging. The control synthesis problem is for many instances of Petri net formalisms undecidable [ABDL16, BHSS12], including those that allow for inhibition [BHSS12] 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 high computational complexity. In fact, most questions about the behaviour of bounded Petri nets are at least PSPACE-hard [Esp98]. We opt to use efficient overapproximation algorithms using both syntactic and local state information to generate stable stubborn sets.

The work presented in this article is an extended version with full proofs of our conference paper [BJL+19]. The stubborn set conditions presented in [BJL+19] were insufficient in order to guarantee the preservation of reachability while condition **C** from the conference paper was found to be redundant. These issues are fixed, and in the present article we add an additional visibility condition on player 2 actions and we elaborate on its syntax-based algorithmic overapproximation for the Petri net games. The implementation is accordingly fixed and the efficiency of the method is still confirmed on an extended set of case studies compared to [BJL+19].

2. Preliminaries

We shall first introduce the basic notation and definitions.

Definition 2.1 (Game Labelled Transition System)**.** A (deterministic) Game Labelled Transition System (GLTS) is a tuple $G = (\mathcal{S}, A_1, A_2, \to, \mathit{Goal})$ 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 $A = A_1 \cup A_2$,
- $\to \subseteq \mathcal{S} \times A \times \mathcal{S}$ is a transition relation such that if $(s, a, s') \in \to$ and $(s, a, s'') \in \to$ then $s' = s''$, and
- $\mathit{Goal} \subseteq \mathcal{S}$ is a set of goal states.

Let $G = (\mathcal{S}, A_1, A_2, \to, \mathit{Goal})$ be a fixed GLTS for the remainder of the section. Whenever $(s, a, s') \in \to$ we write $s \xrightarrow{a} s'$ and say that a is enabled in s and can be *executed* 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. If $en(s) = \emptyset$ then we call s a *deadlock* state. The GLTS G is called *non-mixed* if all states are either player 1, player 2, or deadlock states.

For a sequence of actions $w = a_1 a_2 \cdots a_n \in A^*$ we write $s \xrightarrow{w} s'$ if $s \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} s'$ and say it is *executable*. If $w \in A^\omega$, i.e. if 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 number of actions in a run π is given by the function $\ell : \Pi_G \to \mathbb{N}^0 \cup \{\infty\}$ s.t. for a run $\pi = s_0 \cdots s_n$ we have $\ell(\pi) = n$ if π is finite and otherwise $\ell(\pi) = \infty$. 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 $\Pi_G^{max}(s) = \{\pi \in \Pi_G(s) \mid \ell(\pi) \neq \infty \implies en(\pi_{\ell(\pi)}) = \emptyset\}$. 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 2.2 (Reduction)**.** Let $G = (\mathcal{S}, A_1, A_2, \to, \mathit{Goal})$ be a GLTS. A *reduction* is a function $St : \mathcal{S} \to 2^A$.

Definition 2.3 (Reduced Game)**.** Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, \mathit{Goal})$ be a GLTS and St be a reduction. The *reduced game* of G by the reduction St is given by $G_{St} = (\mathcal{S}, A_1, A_2, \xrightarrow[St]{}, \mathit{Goal})$ 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 that proposes the next action player 1 wants to execute.

Definition 2.4 (Strategy)**.** Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, \mathit{Goal})$ 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 that 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 $\Pi^{max}_\sigma(s) \subseteq \Pi^{max}(s)$ be the set of maximal runs subject to σ starting at $s \in \mathcal{S}$, defined as:

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

Definition 2.5 (Winning Strategy)**.** Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, \mathit{Goal})$ 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. $\pi_i \in \mathit{Goal}$. A state s is called *winning* if there is a winning strategy for player 1 at s.

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 $n \in \mathbb{N}^0$ such that we always reach a goal state with at most n action firings, which we prove in Lemma 2.7. We call this minimum number the *strategy depth* of σ.

Definition 2.6 (Strategy Depth)**.** Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, \mathit{Goal})$ be a GLTS, $s \in \mathcal{S}$ a winning state for player 1 in G and σ a winning strategy at s in G. Then $n \in \mathbb{N}^0$ is the *depth* of σ at s in G if:

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

Lemma 2.7. *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, \mathit{Goal})$ be a GLTS, $s \in \mathcal{S}$ a winning state for player 1 in G, and σ a winning strategy at s in G. Then*

(1) *there exists $n \in \mathbb{N}$ that is the depth of σ at s in G, and*
(2) *if $s \notin \mathit{Goal}$ then for all $a \in next_\sigma(s)$ where $s \xrightarrow{a} s'$, the depth of σ at s' in G is m such that $0 \leq m < n$.*

Proof. (1): Due to A_1 and A_2 being finite and any G being deterministic, we know that every state $s \in \mathcal{S}$ is finitely branching. Since s is a winning state for player 1 in G, we get that every run leads to a goal state in a finite number of actions. Therefore, due to König's

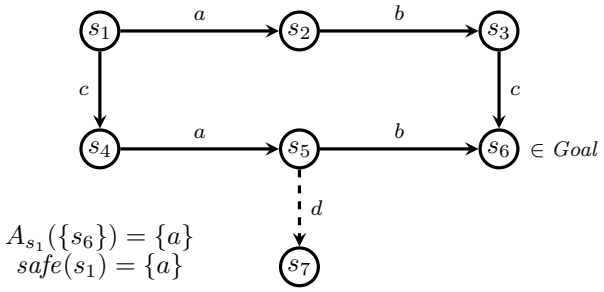

Figure 1: Example of safe and interesting sets of actions for a state s_1

lemma, the tree induced by all runs starting from s, with the leafs being the first occurring goal states, is a finite tree and hence such n exists.

(2): Let n be the depth of σ at s in G and let $s \xrightarrow{a} s'$ such that $a \in next_\sigma(s)$. By contradiction let us assume that the depth of σ at s' is larger than or equal to n. However, this implies the existence of a run π from s' that contains n or more non-goal states before reaching the goal. The run $s\pi$ now contradicts that the depth of s is n. □

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

Definition 2.8 (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 i, $1 \leq i \leq n$, such that $a_i \in A_s(Goal)$.

Example 2.9. In Figure 1 we see an example of a GLTS $G = (\mathcal{S}, A_1, A_2, \rightarrow, Goal)$ where $\mathcal{S} = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$ are the states denoted by circles, $A_1 = \{a, b, c\}$ is the set of player 1 actions, $A_2 = \{d\}$ is the set of player 2 actions, and $\rightarrow$ is denoted by the solid (controllable) and dashed (uncontrollable) transitions between states, labelled by the corresponding actions for player 1 and 2, respectively. Let $Goal = \{s_6\}$. We now 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.

Player 1 has to also consider her safe actions. A player 1 action is *safe* in a given player 1 state if for any player 1 action sequence (excluding the safe action) that does not enable any player 2 action, prefixing this sequence with the safe action will (in case it is executable) also not enable any player 2 action.

Definition 2.10 (Safe Action)**.** Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, Goal)$ be a GLTS and $s \in \mathcal{S}$ a state such that $en_2(s) = \emptyset$. An action $a \in en_1(s)$ is *safe* in s if whenever $w \in (A_1 \setminus \{a\})^*$ with $s \xrightarrow{w} s'$ s.t. $en_2(s') = \emptyset$ and $s \xrightarrow{aw} s''$ then $en_2(s'') = \emptyset$. The set of all safe actions for s is written as $safe(s)$.

Example 2.11. Consider again the GLTS in Figure 1. We reasoned in Example 2.9 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 set of enabled

actions in 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 that s_1 is a winning state for player 1 and player 1 must initially play a as playing c will bring us to the mixed state s_5 from which player 1 does not have winning strategy.

3. Stable Reduction

In this section we introduce the notion of a *stable* reduction St that provides at each state s the set of actions $St(s)$ that are sufficient to be explored so that the given reachability 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 (formulated in general terms of game labelled transition systems) that guarantee that a given reduction preserves winning strategies and we shall call reductions satisfying these conditions *stable*.

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 an arbitrary but fixed set of interesting actions for s and $Goal$.

Definition 3.1 (Stable Reduction Conditions)**.** A reduction St is called *stable* if St satisfies for every $s \in S$ Conditions **I**, **W**, **R**, **G1**, **G2**, **S**, **V** 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 $en_1(s) \subseteq St(s)$
- **V** If there exists $w \in A_2^*$ s.t. $s \xrightarrow{w} s'$ and $s' \in Goal$ then $en_2(s) \subseteq St(s)$.
- **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 that all enabled actions are included in the reduction. That is, we do not attempt to reduce the state space from this state. Condition **W** states that we can swap the ordering of action sequences such that performing stubborn actions first still ensures that we can reach a given state (i.e. a stubborn action commutes with any sequence of nonstubborn actions). Condition **R** ensures that a goal state cannot be reached solely by exploring actions not in the stubborn set (i.e. we preserve the reachability of goal states). Conditions **G1** resp. **G2** ensure that from any state belonging to player 1 (resp. player 2), it is not possible to reach any player 2 (resp. player 1) state or a mixed state, solely by exploring only nonstubborn actions (i.e. reachability of mixed states and opposing player states are preserved in the reduction). Condition **S** ensures that either all enabled stubborn player 1 actions are also safe, or if this is not the case then all enabled player 1 actions are included in the stubborn set. Condition **V** checks if it is possible to reach a goal state by firing exclusively player 2 actions, and includes all enabled player 2 actions into the stubborn set if it is the case. Condition **D** ensures that at least one player 2 action cannot be disabled solely by exploring nonstubborn actions.

Example 3.2. In Figure 2 we see an example of a GLTS using the previously introduced graphical notation. Let $Goal = \{s_8\}$ be the set of goal states and let $A_{s_1}(Goal) = \{a\}$ be a

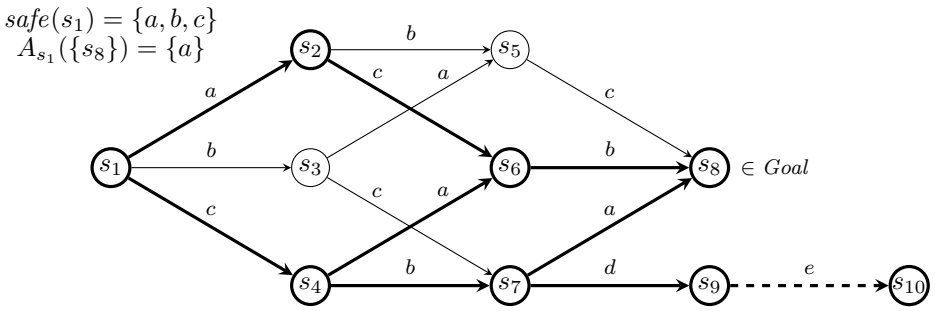

Figure 2: Example of a stable reduction for a state s_1

fixed set of interesting actions. For state s_1 we assume $St(s_1) = \{a, c\}$ as this stubborn set satisfies the stable reduction conditions. We satisfy **G1** since c has to be fired before we can reach the player 2 state s_9. 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 as well. Clearly $St(s_1)$ contains the interesting set $A_{s_1}(Goal)$ that we fixed to $\{a\}$, so **R** is satisfied. Condition **S** is satisfied since $St(s_1) \cap en(s_1) \subseteq safe(s_1)$. We have that **I**, **G2**, **V**, and **D** are satisfied as well since their antecedents are not true. Thick lines in the figure 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.

We shall now prove the correctness of our stubborn set reduction. We 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 3.3. *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, Goal)$ be a GLTS, St a reduction that satisfies Conditions **W** and **R**, and $s \in \mathcal{S} \setminus Goal$ a state. If there exists $w \in A^*$ s.t. $s \xrightarrow{w} s'$ and $s' \in Goal$ then $St(s) \cap en(s) \neq \emptyset$.*

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 j, $1 \leq j < i$, 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 $a_i \in St(s) \cap en(s)$. □

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 3.4. *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, Goal)$ be a GLTS and St a stable reduction. If a state $s \in \mathcal{S}$ is winning for player 1 in G then s is also winning for player 1 in G_{St}.*

Proof. Assume that $s \in \mathcal{S}$ is a winning state for player 1 in G. By definition we have that there exists a player 1 strategy σ such that for all $\pi \in \Pi^{max}_{G,\sigma}(s)$ there exists a position i s.t. $\pi_i \in Goal$. By induction on n we now prove the induction hypothesis $IH(n)$: "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 $s \in Goal$ and so s is trivially a winning state for player 1 also in G_{St}.

Induction step. Let $n > 0$ and let σ be a winning strategy with depth n for s. 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 as we otherwise have $n = 0$.

Case (1): Let $en_1(s) \neq \emptyset$ and $en_2(s) \neq \emptyset$. We assume that s is a winning state for player 1 in G with a strategy σ with a depth of n and we want to show that 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 due to property 2 of Lemma 2.7. By the induction hypothesis s' is a winning state for player 1 in G_{St} and there exists a strategy σ' s.t. σ' is a winning strategy for player 1 at s' 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. $\sigma'(s) = \sigma(s)$. From the definition of a winning strategy we have that no matter what action player 2 chooses, the resulting state is a winning state for player 1, and hence s is a winning state for player 1 in G_{St}.

Case (2): Let $en_2(s) = \emptyset$. Assume that s is a winning state for player 1 in G with a strategy σ with a depth of n. We want to show that 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 an $m \leq n$ s.t. $\pi_0 \xrightarrow{a_1} \pi_1 \xrightarrow{a_2} \cdots \xrightarrow{a_m} \pi_m$ and $\pi_m \in \mathit{Goal}$. Let $w = a_1 \cdots a_m$. We start by showing that there exists i, $1 \leq i \leq m$, such that $a_i \in St(s)$. Assume that $w \in \overline{St(s)}^*$ is true. Then we have $\pi_m \notin \mathit{Goal}$ due to Condition **R**, a contradiction. Therefore there must exist i, $1 \leq i \leq m$, s.t. $a_i \in St(s)$. Let i be minimal in the sense that for all j, $1 \leq j < i$, 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) $a_i \in \mathit{safe}(s)$ or (2.2) $a_i \notin \mathit{safe}(s)$.

- Case (2.1): Let $a_i \in \mathit{safe}(s)$. 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 intermediate states in $s \xrightarrow{a_iv} \pi_i$ 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 let $v = a_1a_2 \cdots a_{i-1}$ s.t. $s'_0 \xrightarrow{a_1} s'_1 \xrightarrow{a_2} \cdots \xrightarrow{a_{i-1}} \pi_i$ and for all k, $1 \leq k < i-1$, we have $en_2(s'_k) = \emptyset$. Let σ'' be defined such that for all j, $0 < j < i-1$, we have $\sigma''(s'_{j-1}) = a_j$, and let σ'' from π_i be defined as σ. Clearly, σ'' is a winning strategy for player 1 at s'_0 in G. Due to property 2 of Lemma 2.7 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'' from s'_0 we always reach π_i in $i-1$ actions. From this we can infer that the depth of σ'' 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 $k + i - 1 < n$ 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} and there exists a strategy σ' s.t. σ' is a winning strategy for player 1 at s'_0 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): Let $a_i \notin \mathit{safe}(s)$. Since $a_i \notin \mathit{safe}(s)$ we have $St(s) \cap en_1(s) \not\subseteq \mathit{safe}(s)$ 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 of Lemma 2.7. By the induction hypothesis s' is a winning state for player 1 in G_{St} and there exists a strategy σ' s.t. σ' is

a winning strategy for player 1 at s' 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): Let $en_1(s) = \emptyset$. Assume that s is a winning state for player 1 in G with σ as the winning strategy. We want to show that 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 that 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 $St(s)$. As $en_2(s) \neq \emptyset$, due to Condition **D** we get that there exists $a \in en_2(s) \cap St(s)$, and this last case is also established. □

Lemma 3.5. *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, Goal)$ be a GLTS and St a stable reduction. If a state $s \in \mathcal{S}$ is winning for player 1 in G_{St} then s is also winning for player 1 in G.*

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

Induction step. Let $n > 0$ and let s be a winning state for player 1 in G_{St} with a strategy with a depth of n. 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 in G_{St} such that $en(s) \cap St(s) = \emptyset$ is not possible as otherwise we have the case where $n = 0$.

Case (1): Let $en_1(s) \cap St(s) \neq \emptyset$ and $en_2(s) \cap St(s) \neq \emptyset$. We assume that s is a winning state for player 1 in G_{St} with a strategy σ with a depth of n. We want to show that 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, whenever $s \xrightarrow[St]{\sigma(s)} s'$ 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 of Lemma 2.7. By the induction hypothesis s' is a winning state for player 1 in G and there exists a strategy σ' s.t. σ' is a winning strategy for player 1 at s' in G. Since s is a mixed state in G_{St} then s must also be a mixed state in G due to $\xrightarrow[St]{} \subseteq \rightarrow$. This implies that $s \xrightarrow{\sigma(s)} s'$. Therefore player 1 can choose the same action proposed in the reduced game such that $\sigma'(s) = \sigma(s)$. Furthermore we have $en_2(s) \cap St(s) = en_2(s)$ from Condition **I**. From this we can conclude that s is a winning state for player 1 in G with strategy σ'.

Case (2): Let $en_2(s) \cap St(s) = \emptyset$. Assume that s is a winning state for player 1 in G_{St} with a strategy σ with a depth of n. We want to show that 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 of Lemma 2.7. By the induction hypothesis s' is a winning state for player 1 in G and there exists a strategy σ' s.t. σ' is a winning strategy for player 1 at s' in G. Trivially we have that $s \xrightarrow{\sigma(s)} s'$ since we have $\xrightarrow[St]{} \subseteq \rightarrow$. Therefore player 1 can choose the same action proposed in the reduced game $\sigma'(s) = \sigma(s)$. Next we show by contradiction that s is a player 1 state also 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): Let $en_1(s) \cap St(s) = \emptyset$. Assume that s is a winning state for player 1 in G_{St} with a strategy σ with a depth of n. We want to show that 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. We define $\sigma' = \sigma$.

What remains to be shown is that s is a winning state for player 1 in G. For the sake of contradiction assume that this is not the case, i.e. that there exists $\pi \in \Pi^{max}_{G,\sigma'}(s)$ such that

$$s = \pi_0 \xrightarrow{a_1} \pi_1 \xrightarrow{a_2} \pi_2 \xrightarrow{a_3} \cdots$$

and $\pi_i \notin \mathit{Goal}$ for all positions i. We shal first argue that $a_1 \notin St(s)$. If this is not the case, then $\pi_0 \xrightarrow[St]{a_1} \pi_1$ also in the reduced game G_{St}. Due to our assumption that $s = \pi_0$ is a winning state for player 1 in G_{St} and $a_1 \in A_2$, we know that also π_1 is a winning state for player 1 in G_{St} with $m < n$ as the depth of σ at π_1 in G_{St}, following property 2 of Lemma 2.7. By the induction hypothesis π_1 is a winning state for player 1 in G, which contradicts the existence of the maximal path π with no goal states.

Let us so assume that $a_1 \notin St(s)$. Let $j > 1$ be the smallest index such that $a_j \in St(s)$ and $a_1 a_2 \cdots a_{j-1} a_j \in A_2^*$. Such index must exist because of the following case analysis.

- Either the sequence $a_1 a_2 \cdots$ contains an action that belongs to A_1 (we note that because of our assumption $a_1 \notin A_1$). Due to Condition **G2** there must exist an action a_j that is stubborn in s and let $j > 1$ be the smallest index such that $a_j \in St(s)$. As a_j is the first action that is stubborn, we get that $a_1 a_2 \cdots a_{j-1} a_j \in A_2^*$ as otherwise the existence of $i \leq j$ where $a_i \in A_1$ contradicts the minimality of j due to Condition **G2**.
- Otherwise the sequence $a_1 a_2 \cdots$ consists solely of actions from A_2. If the sequence contains a stubborn action then we are done and similarly, if the sequence is finite and ends in a deadlock, we get by Condition **D** that there must be an $j > 1$ where $a_j \in St(s)$ as required. The last option is that the sequence $a_1 a_2 \cdots$ is infinite and does not contain any stubborn action. By Condition **D** there exists $a \in en_2(s) \cap St(s)$ such that for all $i > 0$ we have $s \xrightarrow{a_1 \cdots a_i} \pi_i \xrightarrow{a} \pi'_i$ and then by Condition **W** we get $s \xrightarrow{a} \pi'_0 \xrightarrow{a_1 \cdots a_i} \pi'_i$. This implies that from π'_0 we can also execute the infinite sequence of actions $a_1 a_2 \cdots$ while Condition **V** guarantees that none of the states visited during this execution is a goal state. Hence the state π'_0 must be losing for player 1 in G, which however contradicts that by induction hypothesis π'_0 is winning for player 1 in G as $s \xrightarrow{a} \pi'_0$ with $a \in St(s)$ and the depth of player 1 winning strategy at π'_0 in G_{St} is smaller than the depth at s in G_{St}. Hence there cannot be any infinite sequence of nonstubborn actions starting from s.

As we have now established that there is the smallest index $j > 1$ such that $a_j \in St(s)$ and $a_1 a_2 \cdots a_{j-1} a_j \in A_2^*$, the minimality of j implies that $a_1 a_2 \cdots a_{j-1} \in \overline{St(s)}^*$. This means that we can apply Condition **W** and conclude that there exists a maximal run π' given by

$$s \xrightarrow{a_j} s' \xrightarrow{a_1 a_2 \cdots a_{j-1}} \pi_j \xrightarrow{a_{j+1}} \pi_{j+1} \xrightarrow{a_{j+2}} \cdots$$

that is from π_j identical to the run of π. Hence $\pi_i \notin \mathit{Goal}$ for all $i \geq j$. We notice that also the intermediate states in the prefix of the run π' may not be goal states, which is implied by Condition **V** and the fact that $a_1 \in en_2(s)$, $a_1 a_2 \cdots a_{j-1} a_j \in A_2^*$, and $a_1 \notin St(s)$. However,

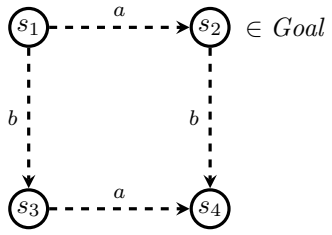

Figure 3: Example showing the importance of Condition **V**

as $a_j \in St(s)$ we get $s \xrightarrow[St]{a_j} s'$ and because $a_j \in A_2$ we know that 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 of Lemma 2.7. By the induction hypothesis s' is a winning state for player 1 in G, which contradicts the existence of a maximal run from s' that contains no goal states. Hence the proof of Case (3) is finished. □

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

Theorem 3.6 (Strategy Preservation for GLTS)**.** *Let $G = (\mathcal{S}, A_1, A_2, \to, Goal)$ be a GLTS and St a stable reduction. A state $s \in \mathcal{S}$ is winning for player 1 in G iff s is winning for player 1 in G_{St}.*

Proof. Follows from Lemma 3.4 and 3.5. □

Remark 3.7. In [BJL$^+$19] we omitted Condition **V** from the definition of stable reduction and this implied that Lemma 3.5 (as it was stated in [BJL$^+$19]) did not hold. We illustrate this in Figure 3 where all actions are player 2 actions and the goal state is s_2. Clearly, player 1 does not have a winning strategy as player 2 can play the action b followed by a and reach the deadlock state s_4 without visiting the goal state. The stubborn set $St(s_1) = \{a\}$ on the other hand satisfies all conditions of the stable reduction, except for **V**, however, it breaks Lemma 3.5 because in the reduced system the action b in s_1 is now exluded and the (only) stubborn action a for the environment brings us to a goal state. It is therefore the case that in the original game s_1 is not a winning state for player 1 but in the reduced game it is. The extra Condition **V** introduced in this article forces us to include all enabled actions in s_1 into the stubborn set, and hence the validity of Lemma 3.5 is recovered.

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

Theorem 3.8 (Strategy Preservation for Non-Mixed GLTS)**.** *Let $G = (\mathcal{S}, A_1, A_2, \to, Goal)$ be a non-mixed GLTS and St a stable reduction with Condition **S** excluded. A state $s \in \mathcal{S}$ is winning for player 1 in G iff s is winning for player 1 in G_{St}.*

Proof. In Lemma 3.5 the condition **S** is not used at all. In Lemma 3.4 the subcase (2.2) is the only one that relies on **S**. Because there are no mixed states, the arguments in subcase (2.1) are valid irrelevant of whether a_i is safe or not. □

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 4.1 (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,
- $W : (P \times T) \cup (T \times P) \to \mathbb{N}^0$ is a weight function for regular arcs, and
- $I : (P \times T) \to \mathbb{N}^\infty$ is a weight function for inhibitor arcs.

A *marking* M is a function $M : P \to \mathbb{N}^0$ and $\mathcal{M}(N)$ denotes 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 $T = T_1 \uplus T_2$. 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 $G(N) = (\mathcal{S}, A_1, A_2, \to, \mathit{Goal})$ 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.

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 relative to a marking $M \in \mathcal{M}(N)$ by the function $\mathit{eval}_M : E_N \to \mathbb{Z}$ where $\mathit{eval}_M(c) = c$, $\mathit{eval}_M(p) = M(p)$ and $\mathit{eval}_M(e_1 \oplus e_2) = \mathit{eval}_M(e_1) \oplus \mathit{eval}_M(e_2)$.

In Table 1 we define the functions $\mathit{incr}_M : E_N \to 2^T$ and $\mathit{decr}_M : E_N \to 2^T$ that, given an expression $e \in E_N$, return the set of transitions that can (when fired) increase resp. decrease the evaluation of e. We note that transitions in $\mathit{incr}_M(e)$ and $\mathit{decr}_M(e)$ are not necessarily enabled in M, however, due to Lemma 4.2, if a transition firing increases the evaluation of e then the transition must be in $\mathit{incr}_M(e)$, and similarly for $\mathit{decr}_M(e)$.

Lemma 4.2 [BJL+18]**.** *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 $w = t_1 t_2 \dots t_n \in T^*$.*

- *If $\mathit{eval}_M(e) < \mathit{eval}_{M'}(e)$ then there is i, $1 \leq i \leq n$, such that $t_i \in \mathit{incr}_M(e)$.*
- *If $\mathit{eval}_M(e) > \mathit{eval}_{M'}(e)$ then there is i, $1 \leq i \leq n$, such that $t_i \in \mathit{decr}_M(e)$.*

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 $e \in E_N$

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

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

where $e_1, e_2 \in E_N$, $t \in T$ and $\bowtie \in \{<, \leq, =, \neq, >, \geq\}$.

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

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

We want to be able to preserve at least one execution to the set $\mathit{Goal} = \{M \in \mathcal{M}(N) \mid M \models \varphi\}$ 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 formula 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 [BJL+18] for the non-game setting, except for the conjunction where we in our setting use Equation (4.1) that provides an optimisation for Condition **S** and possibly ends with a smaller set of interesting transitions.

$$A_M(\varphi_1 \wedge \varphi_2) = \begin{cases} A_M(\varphi_1) & \text{if } M \models \varphi_2 \\ A_M(\varphi_2) & \text{if } M \models \varphi_1 \\ A_M(\varphi_1) & \text{if } M \not\models \varphi_1 \text{ and } A_M(\varphi_1) \subseteq \mathit{safe}(M) \\ A_M(\varphi_2) & \text{if } M \not\models \varphi_2 \text{ and } A_M(\varphi_2) \subseteq \mathit{safe}(M) \\ A_M(\varphi_i) & \text{otherwise where } i \in \{1, 2\} \end{cases} \tag{4.1}$$

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

Lemma 4.3. *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 $M' \not\models \varphi$.*

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 [BJL+18]. Let $\varphi = \varphi_1 \wedge \varphi_2$.

φ	$A_M(\varphi)$	$A_M(\neg\varphi)$
deadlock	$t \cup ({}^\bullet t)^- \cup {}^+({}^\circ t)$ for some selected $t \in en(M)$	$\emptyset$
t	${}^+p$ for some selected $p \in {}^\bullet t$ where $M(p) < W(p,t)$, or p^- for some selected $p \in {}^\circ t$ 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 (4.1)	$A_M(\neg\varphi_1 \vee \neg\varphi_2)$
$\varphi_1 \vee \varphi_2$	$A_M(\varphi_1) \cup A_M(\varphi_2)$	$A_M(\neg\varphi_1 \wedge \neg\varphi_2)$

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

There are five subcases defined by Equation 4.1: (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 case.

- Case (1): Let $M \models \varphi_2$. 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 4.1, 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): Let $M \models \varphi_1$. This case is symmetric to Case (1) and follows the same approach.
- Case (3): Let $M \not\models \varphi_1$ and $A_M(\varphi_1) \subseteq \mathit{safe}(M)$. By Equation 4.1 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): Let $M \not\models \varphi_2$ and $A_M(\varphi_2) \subseteq \mathit{safe}(M)$. This case is symmetric to Case (3) and follows the same approach.
- Case (5): Default case. We have $M \not\models \varphi_1$ and $M \not\models \varphi_2$ due to Equation 4.1 and $A_M(\varphi_1 \wedge \varphi_2) = A_M(\varphi_i)$ for some $i \in \{1,2\}$. By the induction hypothesis this implies $M' \not\models \varphi_i$, and from this and the semantics of φ we have $M' \not\models \varphi$. □

As a next step, we provide an algorithm that returns *true* whenever there is a sequence of player 2 actions that leads to a marking satisfying a given formula φ (and hence overapproximates Condition **V** from the definition of a stable reduction). The pseudocode is given in Algorithm 1. The algorithm uses an extended definition of formula satisfiability that, instead of asking whether a formula holds in a given marking, specifies instead a range of markings by two functions $lb : P \to \mathbb{N}^0$ for fixing a lower bound on the number of tokens in places and $ub : P \to \mathbb{N}^0 \cup \{\infty\}$ for specifying an upper bound. A marking M belongs to the range lb, ub iff for all places $p \in P$ we have $lb(p) \leq M(p) \leq ub(p)$. The extended satisfability predicate $lb, ub \models \varphi$ is given in Table 3 and it must hold whenever there is a marking in the range specified by lb and ub such that the marking satisfies the formula φ. Finally, Algorithm 1 computes a safe overapproximation of the lower and upper bounds such that if $M \xrightarrow{w} M'$ for some $w \in T_2^*$ then $lb(p) \leq M'(p) \leq ub(p)$ for all $p \in P$.

Algorithm 1: *reach*(N, M, φ): Overapproximation for checking if φ can be satisfied by performing only player 2 transitions, assuming that $\min \emptyset = \infty$ and $\sum \emptyset = 0$

input : $N = (P, T_1, T_2, W, I)$ with $M \in \mathcal{M}(N)$ and a formula $\varphi \in \Phi_N$
output : If there is $w \in A_2^*$ s.t. $M \xrightarrow{w} M'$ and $M' \models \varphi$ then the algorithm returns *true*.

We assume that all negations in φ are only in front of atomic propositions (if not, we can use De Morgan's laws in order to guarantee this).
$ub(x) := \infty$ for all $x \in P \cup T_2$;
$ub(p) := M(p)$ for all $p \in P$ such that $W(p,t) \geq W(t,p)$ for every $t \in {}^\bullet p \cap T_2$;
repeat
  foreach $t \in T_2$ **do**
    $ub(t) := \min_{p \in {}^-t} \lfloor \frac{ub(p)}{W(p,t) - W(t,p)} \rfloor$
  foreach $p \in P$ **do**
    $ub(p) := M(p) + \sum_{\substack{t \in {}^\bullet p \cap T_2 \\ W(t,p) > W(p,t)}} ub(t) \cdot \big(W(t,p) - W(p,t)\big)$
until $ub(x)$ *stabilises for all* $x \in P \cup T_2$
foreach $p \in P$ **do**
  $lb(p) := M(p) - \sum_{\substack{t \in T_2 \\ W(p,t) > W(t,p)}} ub(t) \cdot \big(W(p,t) - W(t,p)\big)$
return $lb, ub \models \varphi$; *** See definition in Table 3

Lemma 4.4. *Let $N = (P, T_1, T_2, W, I)$ be a Petri net game, $M \in \mathcal{M}(N)$ a marking on N and $\varphi \in \Phi_N$ a formula. If there is $w \in A_2^*$ s.t. $M \xrightarrow{w} M'$ and $M' \models \varphi$ then reach$(N, M, \varphi) =$ true.*

Proof. Algorithm 1 first computes for each place $p \in P$ the upper bound $ub(p)$ and lower bound $lb(p)$ on the number of tokens that can appear in p by performing any sequence of player 2 transitions, starting from the marking M. The bounds are then used to return the value of the expression $lb, ub \models \varphi$ that is defined in Table 3.

We shall first notice if there is a marking M' such that $lb(p) \leq M'(p) \leq ub(p)$ for all $p \in P$ and $M' \models \varphi$ then $lb, ub \models \varphi$ holds. This can be proved by a straightforward structural induction on φ while following the cases in Table 3 where the functions lb and ub are extended to arithmetical expressions used in the query language such that for every marking M' (as given above) and for every arithmetical expressions e we have $lb(e) \leq eval_{M'}(e) \leq ub(e)$.

What remains to be established is the property that Algorithm 1 correctly computes the lower and upper bounds for all places in the net. We do this by proving the invariant for the repeat-until loop that claims that for every $w \in A_2^*$ such that $M \xrightarrow{w} M'$ we have

(1) $M(p) + \sum_{\substack{t \in w \\ W(t,p) > W(p,t)}} \big(W(t,p) - W(p,t)\big) \leq ub(p)$ for all $p \in P$, and
(2) $\#_t(w) \leq ub(t)$ for all $t \in T_2$ where $\#_t(w)$ denotes the number of occurences of the transition t in the sequence w.

$lb, ub \models \mathit{true}$	
$lb, ub \models t$	iff $ub(p) \geq W(p,t)$ for all $p \in {}^\bullet t$ and $lb(p) < I(p,t)$ for all $p \in t^\circ$
$lb, ub \models \neg t$	iff $lb(p) < W(p,t)$ for some $p \in {}^\bullet t$ or $ub(p) \geq I(p,t)$ for some $p \in t^\circ$
$lb, ub \models e_1 < e_2$	iff $lb(e_1) < ub(e_2)$
$lb, ub \models e_1 \leq e_2$	iff $lb(e_1) \leq ub(e_2)$
$lb, ub \models e_1 = e_2$	iff $\max\{lb(e_1), lb(e_2)\} \leq \min\{ub(e_1), ub(e_2)\}$
$lb, ub \models e_1 \neq e_2$	iff it is not the case that $lb(e_1) = lb(e_2) = ub(e_1) = ub(e_2)$
$lb, ub \models e_1 \geq e_2$	iff $ub(e_1) \geq lb(e_2)$
$lb, ub \models e_1 > e_2$	iff $ub(e_1) > lb(e_2)$
$lb, ub \models \mathit{deadlock}$	iff $lb, ub \not\models t$ for all $t \in T$
$lb, ub \models \neg \mathit{deadlock}$	iff $lb, ub \models t$ for some $t \in T$
$lb, ub \models \varphi_1 \wedge \varphi_2$	iff $lb, ub \models \varphi_1$ and $lb, ub \models \varphi_2$
$lb, ub \models \varphi_1 \vee \varphi_2$	iff $lb, ub \models \varphi_1$ or $lb, ub \models \varphi_2$

$$
\begin{aligned}
lb(c) &= c \quad \text{where } c \text{ is a constant} \\
ub(c) &= c \quad \text{where } c \text{ is a constant} \\
lb(e_1 + e_2) &= lb(e_1) + lb(e_2) \\
ub(e_1 + e_2) &= ub(e_1) + ub(e_2) \\
lb(e_1 - e_2) &= lb(e_1) - ub(e_2) \\
ub(e_1 - e_2) &= ub(e_1) - lb(e_2) \\
lb(e_1 * e_2) &= \min\{lb(e_1) \cdot lb(e_2), lb(e_1) \cdot ub(e_2), ub(e_1) \cdot lb(e_2), ub(e_1) \cdot ub(e_2)\} \\
ub(e_1 * e_2) &= \max\{lb(e_1) \cdot lb(e_2), lb(e_1) \cdot ub(e_2), ub(e_1) \cdot lb(e_2), ub(e_1) \cdot ub(e_2)\}
\end{aligned}
$$

Table 3: Definition of $lb, ub \models \varphi$ assuming that $lb(p)$ and $ub(p)$ are given for all $p \in P$

Here the notation $t \in w$ means that a summand is added for every occurence of t in the sequence w. We note that invariant (1) clearly implies that whenever $M \xrightarrow{w} M'$ for $w \in A_2^*$ then $M'(p) \leq ub(p)$ for all $p \in P$. Notice that the repeat-until loop in Algorithm 1 clearly terminates since during the iteration of the loop ub can only become smaller.

First, we notice that before entering the repeat-until loop, the invariant holds because intitially the upper bound values are all set to ∞ and only at line 3 the upper bound for a place p is set to $M(p)$ provided that the firing of any transition $t \in T_2$ can never increase the number of tokens in p. This clearly satisfies invariant (1).

Let us now assume that both (1) and (2) hold at the beginning of the execution of the repeat-until loop. Suppose that the value $ub(t)$ is decreased for some transition t by the assignment at line 6. This means that there is a place $p \in {}^\bullet t$ such that $W(p,t) > W(t,p)$, meaning that firing of t removes $W(t,p) - W(p,t)$ tokens from p. As there can be at most

$ub(p)$ tokens added to the place p due to invariant (1), this limits the number of times that the transition t can fire to $\lfloor \frac{ub(p)}{W(t,p)-W(p,t)} \rfloor$ and hence it preserves invariant (2). Similarly, suppose that the value of $ub(p)$ is decreased for some place p by the assignment at line 8. Due to invariant (2), we know that every transition $t \in T_2$ can be fired at most $ub(t)$ times and hence adds at most $ub(t) \cdot \big(W(t,p) - W(p,t)\big)$ tokens to p. As we add those contributions for all such transitions together with the number $M(p)$ of tokens in the starting marking M, we satisfy also invariant (1).

Finally, the assignment at line 11 provides a safe lower bound on the number of tokens that can be in the place p, as due to invariant (2) we know that $ub(t)$ is the maximum number of times a transition t can fire, and we subtract the number of tokens that each t removes from p by $ub(t)$. Hence, we can conclude that whenever $M \xrightarrow{w} M'$ for $w \in A_2^*$ then $lb(p) \leq M'(p) \leq ub(p)$ for all $p \in P$ and the correctness of the lemma is established. □

Example 4.5. In Figure 4 we see a Petri net consisting of four places $P = \{p_1, p_2, p_3, p_4\}$ and three player 2 transitions $T_2 = \{t_1, t_2, t_3\}$. The weights are given as seen in the figure (arcs without any annotations have the default weight 1) and the initial marking contains three tokens in the place p_1. Initially, for all $x \in P \cup T$ we have $ub(x) = \infty$ as seen in line 2 of Algorithm 1. In line 3 we can set the upper bound of some places if the number of tokens are non-increasing, i.e. for all $t \in {}^\bullet p \cap T_2$ we have $W(p,t) \geq W(t,p)$. In Figure 4 this is the case only for p_1, we therefore have $ub(p_1) = M(p_1) = 3$. Next, the upper bound for all places and transitions are calculated through a repeat-until loop. The upper bound for transitions are found by checking, given the current upper bound on places, how many times we can fire a transition. In line 6 we get

$$ub(t_1) = \left\lfloor \frac{ub(p_1)}{W(p_1,t_1) - W(t_1,p_1)} \right\rfloor = \left\lfloor \frac{3}{1-0} \right\rfloor = 3$$

and

$$\begin{aligned} ub(t_2) &= \min \left\{ \left\lfloor \frac{ub(p_1)}{W(p_1,t_2) - W(t_2,p_1)} \right\rfloor, \left\lfloor \frac{ub(p_3)}{W(p_3,t_2) - W(t_2,p_3)} \right\rfloor \right\} \\ &= \min \left\{ \left\lfloor \frac{3}{2-0} \right\rfloor, \left\lfloor \frac{\infty}{1-0} \right\rfloor \right\} \\ &= \min\{1, \infty\} = 1. \end{aligned}$$

In the next iteration, at line 8 we get

$$ub(p_2) = M(p_2) + ub(t_1) \cdot (W(t_1,p_2) - W(p_2,t_1)) = 0 + 3 \cdot (1-0) = 3$$

and similarly

$$ub(p_4) = M(p_4) + ub(t_2) \cdot (W(t_2,p_4) - W(p_4,t_2)) = 0 + 1 \cdot (1-0) = 1.$$

Afterwards, there are no further changes to be made to the upper bounds and the repeat-until loop terminates. Finally, the calculated lower bounds for all places are 0 in our example.

Before we can state our main theorem, we need to find an overapproximation method for determining safe transitions. This can be done by analysing the increasing presets and postsets of transitions as demonstrated in the following lemma.

Lemma 4.6 (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.*

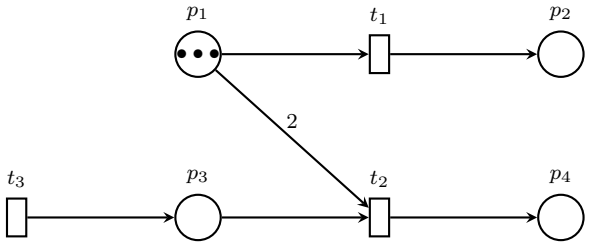

Figure 4: Example Petri Net for Algorithm 1

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 $W(t,p') \geq W(p',t)$, which implies $M''(p') \geq M'(p')$ and $t' \notin en(M'')$. Therefore t is safe in M. □

We can now provide a list of syntactic conditions that guarantee the stability of a given reduction and state the main theorem of this section.

Theorem 4.7 (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 $en_1(M) \subseteq St(M)$.*
(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) *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 $p^- \subseteq St(s)$.*
(7) *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 $p^\circ \subseteq St(M)$.*
(8) *If $en_2(M) \neq \emptyset$ then there exists $t \in en_2(M) \cap St(M)$ s.t. $({}^\bullet t)^- \cup {}^+({}^\circ t) \subseteq St(M)$.*
(9) *If $en_1(M) = \emptyset$ and $reach(N, M, \varphi) = true$ then $en(M) \subseteq St(M)$.*

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

Proof. We shall argue that any reduction St satisfying the conditions of the theorem also satisfies the **I**, **W**, **R**, **G1**, **G2**, **S**, **V**, 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 $t \notin en(M)$. As t is disabled in M, there must be $p \in {}^\bullet t$ 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 6a 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 second case, due to Condition 6b 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 7a 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 7b 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 4.3.

(**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 6 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 imply that $en_2(M') = \emptyset$.

(**G2**) Follows the same approach as **G1**.

(**S**) Follows from Condition 2.

(**V**) Follows from Condition 9 and Lemma 4.4. Notice that if $en_1(M) \neq \emptyset$ then the antecedent of Condition **V** never holds if $en_2(M) = \emptyset$ unless M is already a goal marking, or M is a mixed state and the consequent of Condition **V** always holds due to Condition **I**.

(**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 $t \in en_2(M) \cap St(M)$ s.t. $t \in en_2(M')$. Assume that $en_2(M) \neq \emptyset$. From Condition 8 we know that there exists $t \in en_2(M) \cap St(M)$ 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 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 ${}^+p$

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 ∈ M(N) and formula φ
   output : X ⊆ T where X is a stable stubborn set for M
 1 if en(M) = ∅ then
 2     return T;
 3 if en_1(M) ≠ ∅ ∧ en_2(M) ≠ ∅ then
 4     return T;
 5 Y := ∅;
 6 if en_1(M) = ∅ then
 7     if reach(N, M, φ) then
 8         return T;
 9     Pick any t ∈ en_2(M);
10     Y := T_1 ∪ t ∪ (•t)^− ∪ ^+(°t);
11 else
12     Y := T_2;
13 Y := Y ∪ A_M(φ);
14 X := Saturate(Y);
15 if X ∩ en_1(M) ⊈ safe(M) then
16     return T;
17 return X;
```

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. □

In Algorithm 2 we provide a pseudocode for calculating stubborn sets for a given marking. It essentially rephrases Theorem 4.7 into an executable code. The algorithm calls Algorithm 3 that saturates a given set to satisfy Conditions 6 and 7 of Theorem 4.7.

Theorem 4.8. *Algorithm 2 terminates and returns $St(M)$ for some stable reduction St.*

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 $en_1(M) \cap X \not\subseteq safe(M)$ then we terminate in line 16 of Algorithm 2, and otherwise we return in line 17 and Algorithm 2 terminates.

Correctness. It was shown that the construction in Theorem 4.7 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 $t \in Y$ and we discuss that all conditions of Theorem 4.7 hold upon termination.

- Condition 1: If $en_1(M) \neq \emptyset$ and $en_2(M) \neq \emptyset$ then we return T in line 4 of Algorithm 2.

Algorithm 3: *Saturate*(Y)

```
1  X := ∅;
2  while Y ≠ ∅ do
3      Pick any t ∈ Y;
4      if t ∉ en(M) then
5          if ∃p ∈ •t. M(p) < W(p,t) then
6              Pick any p ∈ •t s.t. M(p) < W(p,t);
7              Y := Y ∪ (+p \ X);
8          else
9              Pick any p ∈ °t s.t. M(p) ≥ I(p,t);
10             Y := Y ∪ (p⁻ \ X);
11     else
12         Y := Y ∪ (((⁻t)• ∪ (t⁺)°) \ X);
13     X := X ∪ {t};
14     Y := Y \ {t};
15 return X;
```

- Condition 2: If $en_1(M) \cap St(M) \not\subseteq safe(M)$ then we return T in line 16 of Algorithm 2.
- Condition 3: We have $A_M(\varphi) \subseteq Y$ in line 13 of Algorithm 2.
- Condition 4: We have $T_1 \subseteq Y$ in line 10 of Algorithm 2.
- Condition 5: We have $T_2 \subseteq Y$ in line 12 of Algorithm 2.
- Condition 6a: 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 6b: 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 7a: In line 12 of Algorithm 3 we add $({}^-t)^\bullet$ to Y.
- Condition 7b: In line 12 of Algorithm 3 we add $(t^+)^\circ$ to Y.
- Condition 8: In line 9 of Algorithm 2 we pick any $t' \in en_2(M)$ and in line 10 we add $({}^\bullet t)^- \cup {}^+({}^\circ t)$ to Y.
- Condition 9: If $en_1(M) = \emptyset$ and $reach(N, M, \varphi) = true$ then we return T at line 8 of Algorithm 2. □

Remark 4.9. 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 3.3).

5. Implementation and Experiments

We extend the Petri net verification engine `verifypn` [JNOS16], a part of the TAPAAL tool suite [DJJ+12], 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 [JLS18, JLS16] that we reimplement in `C++` while utilising PTries [JLS17] for efficient state storage. The source code is available under GPLv3 [BJL+20]. 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 players 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 modeling the Danish train station Lyngby and three of its smaller variants. The scheduling problem, including the station layout, was originally described as a game in [KHV16] 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 [Tag08] 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 an empty stack of pebbles loses. In our (equivalent) model, we are instead adding pebbles to an initially empty stack 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 [QKCC13]. 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 [DZ04] and the goal of the game is to synthesise a strategy that guarantees 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).
- In *Flexible Manufacturing Systems* (FMS) we use the Petri net models from [LZ04, AE98] modeling different production lines with shared resources. The Petri nets FMS-D [AE98] and FMS-C [LZ04] both contain a deadlock and the problem is to control a small subset of transitions so that the deadlock can be avoided. The models are scaled by the number of resources and products in the line. The goal in the FMS-N [LZ04] model is to control a subset of transitions in the net in order to guarantee that a given resource (Petri net place) never becomes empty.

All experimental evaluation is run on AMD Epyc 7551 Processors with 110 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 [BJL+20].

Results. Table 4 shows the experimental evaluation, displaying the relative gain in computation time (in seconds) without (NORMAL) and with (POR) partial order reduction as well in the number of unique markings (in thousands) that were stored during the fixed-point computation on the constructed dependency graph. The results demonstrate significant reductions across all models, in some cases like in NIM and MW even of several degrees of magnitude due to the exponential speed up when using partial order reduction. The case studies FMS-N and FMS-C show a large and consistent reduction in time across all instance sizes. Other models like AIM, PCS, OW and FMS-D show a moderate but significant reduction. We observe that the time reduction is generally only few percent different from the reduction in the number of explored markings, indicating only a few percent overhead for computing (on-the-fly) the stubborn sets. In the FMS-C and in particular the FMS-N model we can see that we achieve significantly larger reduction in running time than in the reduced number of stored markings. This is caused by the fact that the partial order reduction reduces also the number of possible paths in which a certain marking can be discovered.

Our partial order technique noticably speeds up the computation in Lyngby2 model, but there are also two instances of the LyngbySmall models where the reduction both in time and size of the state space is less significant. We conjecture that this is because the search strategy changes when partial order reduction is applied and this results in the fact that we have to search in these two instances a larger portion of the generated dependency graph before we obtain a conclusive answer. Nevertheless, in general the experiments confirm the high practical applicability of partial order reduction for 2-player games with only minimal overhead for computing the stubborn sets. exponential

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 technique for interval approximation on the number of tokens in reachable markings. 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 [BJL+18] in order to extend our framework to general timed games.

Acknowledgments. We are grateful to Thomas Neele from Eindhoven University of Technology for letting us know about the false claim in Lemma 3.5 that was presented in the conference version of this article. The counterexample, presented in Remark 3.7, is attributed to him. We are obliged to Antti Valmari for noticing that condition **C** in our conference paper is redundant and can be substituted by conditions **W** and **D**, as it is done in this article. We also thank the anonymous reviewers for their numerous suggestions that

	Time (seconds)		Markings ×1000		Reduction	
Model	NORMAL	POR	NORMAL	POR	%Time	%Markings
AIM-13-100-6-11	54.15	19.86	1702	510	63	70
AIM-13-100-6-16	76.07	28.71	2464	740	62	70
AIM-13-150-9-16	162.10	115.30	3696	2455	29	34
AIM-13-150-9-21	212.80	153.00	4853	3331	28	31
AIM-14-150-9-16	200.30	142.90	4259	2865	29	33
AIM-15-150-9-16	243.30	172.50	4861	3205	29	34
PCS-2-3	49.71	37.86	13660	9839	24	28
PCS-2-4	181.50	126.60	37580	25625	30	32
PCS-2-5	488.40	331.90	84059	55049	32	35
PCS-2-6	1226.00	756.50	164096	104185	38	37
LyngbySmall2	1.47	0.01	359	2	99	99
LyngbySmall3	9.79	5.59	2118	1165	43	45
LyngbySmall4	60.66	45.32	11605	7407	25	36
Lyngby2	1440.00	116.00	137169	11213	92	92
NIM-5-49500	3.88	1.24	1635	595	68	64
NIM-7-49500	14.40	1.78	5282	753	88	86
NIM-9-49500	65.36	2.42	17326	963	96	94
NIM-11-49500	326.40	3.28	59491	1167	99	98
MW-20	18.03	0.02	4333	4	100	100
MW-30	93.71	0.04	14643	6	100	100
MW-40	311.50	0.06	34733	9	100	100
MW-50	795.30	0.09	67869	11	100	100
MW-60	1749.00	0.13	117313	13	100	100
OW-100000	4.20	3.25	2300	1800	23	22
OW-1000000	52.04	40.37	23000	18000	22	22
OW-10000000	591.30	435.60	230000	180000	26	22
FMS-D-4	40.81	35.38	7145	6682	13	6
FMS-D-5	98.74	90.98	15735	15156	8	4
FMS-D-6	170.20	159.90	25873	25265	6	2
FMS-D-7	246.80	238.20	36569	35918	3	2
FMS-C-300	218.50	105.40	24730	16103	52	35
FMS-C-400	349.50	170.40	32877	21411	51	35
FMS-C-500	471.30	239.90	41026	26718	49	35
FMS-C-600	587.10	285.60	49173	32024	51	35
FMS-N-9000	52.84	16.69	10423	8579	68	18
FMS-N-29000	201.90	64.08	33583	27639	68	18
FMS-N-49000	372.50	119.00	56743	46699	68	18
FMS-N-69000	538.70	179.70	79903	65759	67	18

Table 4: Experiments with and without partial order reduction (POR and NORMAL)

helped us to improve the quality of the presentation. The research leading to these results has received funding from the project DiCyPS funded by the Innovation Fund Denmark, the ERC Advanced Grant LASSO and DFF project QASNET.

References

[ABDL16] 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 Berlin Heidelberg, 2016.

[AE98] I. B. Abdallah and H. A. ElMaraghy. Deadlock Prevention and Avoidance in FMS: A Petri Net Based Approach. *The International Journal of Advanced Manufacturing Technology*, 14(10):704–715, 1998. Springer.

[BHSS12] 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 Berlin Heidelberg, 2012.

[BJL+18] F.M. Bønneland, P.G. Jensen, K.G. Larsen, M. Muñiz, 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.

[BJL+19] F.M. Bønneland, P.G. Jensen, K.G. Larsen, M. Muñiz, and J. Srba. Partial Order Reduction for Reachability Games. In *International Conference on Concurrency Theory*, volume 140 of *Leibniz International Proceedings in Informatics*, pages 23:1–23:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019.

[BJL+20] Frederik Meyer Bønneland, Peter Gjøl Jensen, Kim Guldstrand Larsen, Marco Mũniz, and Jiri Srba. Artifact for "Partial Order Reduction for Reachability Games", September 2020.

[DEF+18] 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. IOS Press.

[DJJ+12] 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 Berlin Heidelberg, 2012.

[DZ04] 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 Berlin Heidelberg, 2004.

[Esp98] J. Esparza. *Decidability and Complexity of Petri Net Problems — An Introduction*, volume 1491 of *LNCS*, pages 374–428. Springer Berlin Heidelberg, 1998.

[GKPP99] R. Gerth, R. Kuiper, D. Peled, and W. Penczek. A Partial Order Approach to Branching Time Logic Model Checking. *Information and Computation*, 150(2):132–152, 1999. Elsevier.

[God90] P. Godefroid. Using Partial Orders to Improve Automatic Verification Methods. In *Computer Aided Verification*, volume 531 of *LNCS*, pages 176–185. Springer Berlin Heidelberg, 1990.

[God96] Patrice Godefroid. *Partial-Order Methods for the Verification of Concurrent Systems: An Approach to the State-Explosion Problem*, volume 1032 of *LNCS*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996.

[GW93] P. Godefroid and P. Wolper. Using Partial Orders for the Efficient Verification of Deadlock Freedom and Safety Properties. *Formal Methods in System Design*, 2(2):149–164, 1993. Springer.

[HNW98] 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 Berlin Heidelberg, 1998.

[JLS16] P.G. Jensen, K.G. Larsen, and J. Srba. Real-Time Strategy Synthesis for Timed-Arc Petri Net Games via Discretization. In *Model Checking Software*, volume 9641 of *10580*, pages 129–146. Springer International Publishing, 2016.

[JLS17] 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 Berlin Heidelberg, 2017.

[JLS18] 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, 2018. Springer Berlin Heidelberg.

[JNOS16] 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 Berlin Heidelberg, 2016.

[JPDM18] 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. ACM, 2018.

[KHV16] 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. ACM, 2016.

[LLW12] 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.

[LW14] 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 Berlin Heidelberg, 2014.

[LZ04] Z.W. Li and M.C. Zhou. Elementary siphons of petri nets and their application to deadlock prevention in flexible manufacturing systems. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 34(1):38–51, 2004.

[NWW20] T. Neele, T.A.C. Willemse, and W. Wesselink. Partial-Order Reduction for Parity Games with an Application on Parameterised Boolean Equation Systems. In *Tools and Algorithms for the Construction and Analysis of Systems"*, volume 12079 of *LNCS*, pages 307–324. Springer Berlin Heidelberg, 2020.

[Pel93] 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 Berlin Heidelberg, 1993.

[Pel96] D. Peled. Combining Partial Order Reductions With On-The-Fly Model-Checking. *Formal Methods in System Design*, 8(1):39–64, 1996. Springer.

[Pel98] D. Peled. Ten Years of Partial Order Reduction. In *Computer Aided Verification*, volume 1427 of *LNCS*, pages 17–28. Springer Berlin Heidelberg, 1998.

[QKCC13] 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, 2013. Elsevier.

[RS97] 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.

[Tag08] 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.

[Val91] A. Valmari. Stubborn Sets for Reduced State Space Generation. In Grzegorz Rozenberg, editor, *Advances in Petri Nets 1990*, volume 483 of *LNCS*, pages 491–515. Springer, 1991.

[Val92] A. Valmari. A Stubborn Attack on State Explosion. *Formal Methods in System Design*, 1(4):297–322, 1992. Springer Berlin Heidelberg.

[Val93] A. Valmari. On-The-Fly Verification with Stubborn Sets. In *Computer Aided Verification*, volume 697 of *LNCS*, pages 397–408. Springer Berlin Heidelberg, 1993.

[Val97] A. Valmari. Stubborn Set Methods for Process Algebras. In *Proceedings of the DIMACS Workshop on Partial Order Methods in Verification*, POMIV'96, page 213–231. Association for Computing Machinery, 1997.

[VH17] 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 Berlin Heidelberg, 2017.

[WW96] 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, 1996.