Stubborn Set Reduction for Timed Reachability and Safety Games

Frederik M. Bønneland, Peter G. Jensen, Kim G. Larsen,
Marco Muñiz, and Jiří Srba

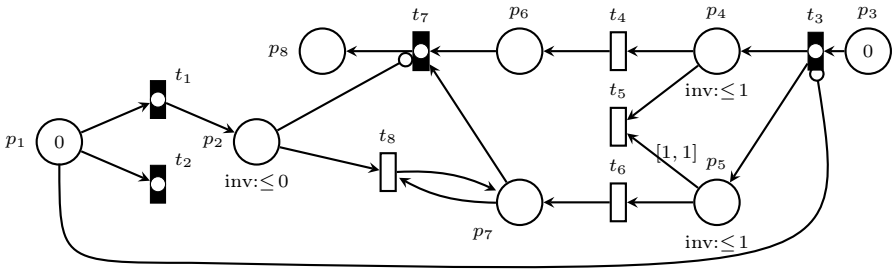
Department of Computer Science, Aalborg University, Denmark
{frederikb,pgj,kgl,muniz,srba}@cs.aau.dk

Abstract. Timed games are an essential formalism for modeling time-sensitive reactive systems that must respond to uncontrollable events triggered by the (hostile) environment. However, the control synthesis problem for these systems is often resource-demanding due to the state space explosion problem. To counter this problem, we present an extension of partial order reduction, based on stubborn sets, into timed games. We introduce the theoretical foundations on the general formalism of timed game labeled transition systems and then instantiate it to the model of timed-arc Petri net games. We provide an efficient implementation of our method as part of the model checker TAPAAL and discuss an experimental evaluation on several case studies that show increasing (sometimes even exponential) savings in time and memory as the case studies scale to larger instances. To the best of our knowledge, this is the first application of partial order reductions to a game formalism that includes time.

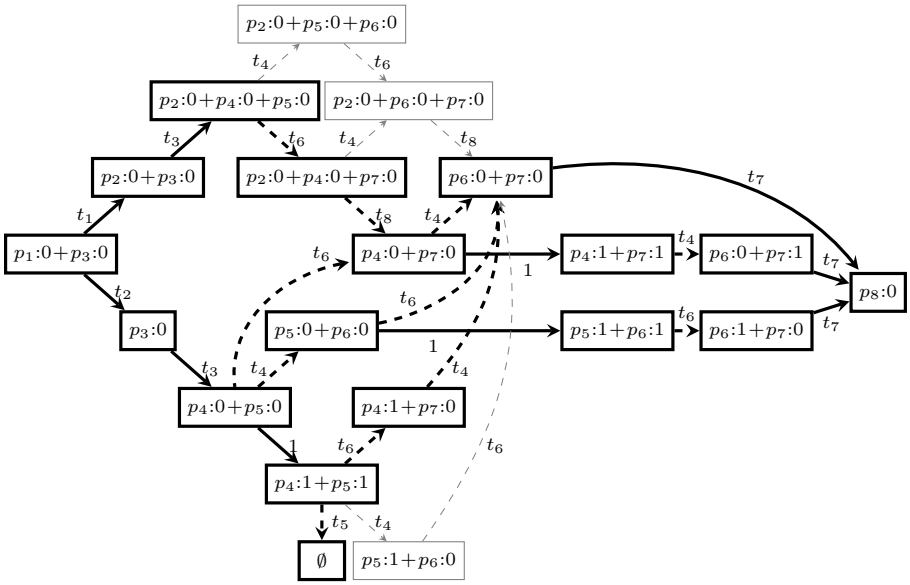
1 Introduction

Even for simple concurrent systems, the reachable state space can become too large and render formal methods such as model checking intractable. This so-called state space explosion problem can be induced by exponentially many, and often redundant (for the validity of the verified property), interleavings of concurrent and independent system actions. Many methods exist for alleviating this problem, such as *symmetry reductions* [12,14] or *parametric verification* [2,1,17]. We focus on a family of methods called *(static) partial order reductions* [19,31,34], which exploit the commutativity of independent system actions to prune redundant interleavings. The most prevalent variants of partial order reductions are *persistent sets* [19,20], *ample sets* [31,32], and *stubborn sets* [34,35].

We combine the approaches of stubborn sets for timed systems and reachability games, initially presented respectively in [8] and [10], into a unified method for general timed games represented as a timed game labelled transition system (TGLTS). As main contributions, we present a new correctness proof of the method for preserving the existence of winning *reachability* strategies to accomodate for the timed setting. In addition, we show that our method also preserves winning *safety* strategies with minimal changes to the game-theoretic framework.



(a) Timed-arc Petri net game



(b) State space generated by the net in Figure 1a

Fig. 1: Example of timed-arc Petri net game and its reachable state space

This provides, to the best of our knowledge, the first partial order reduction approach for general timed games. We instantiate our framework to Timed-Arc Petri net Games (TAPG) [25] with inhibitor arcs, and propose syntax-driven, overapproximation algorithms for generating stable stubborn sets. While these algorithms are similar to the algorithms presented for timed systems and games, we leverage further improvements by a detailed analysis of the guard and invaraint We implement our method in the model checker TAPAAL [13,23] and provide an experimental evaluation to showcase the potential of our method.

Let us first intuitively introduce the model of timed-arc Petri net games [25] that extends timed-arc Petri nets [4,21] into a game setting by splitting transitions into the controllable and uncontrollable ones. In our example in Figure 1a

we consider the discrete time semantics (integer time delays) as in [25,26]. The *controllable* (player 1) transitions are denoted as filled rectangles and the *uncontrollable* (player 2) transitions are shown as transparent rectangles. Places in the net (denoted by circles) can contain tokens which are associated with non-negative integers representing their age. For example, the places p_1 and p_3 each contain a token of age 0 in the initial marking of the net. Delay actions uniformly increase the age of tokens, but delays may be prevented in markings that are *urgent*. A marking is urgent if an urgent transition (denoted with a circle inside the rectangle) is enabled; in our example all controllable transitions are urgent. Furthermore, a marking can be urgent due to place invariants which constrain the allowed age of tokens in a given place, e.g. place p_2 can only contain tokens of age 0 due to the invariant inv: ≤ 0. If a marking is not urgent then a delay action may occur which progresses the age of all tokens in the marking evenly by some numerical value, as long as all place invarints are satisfied. Firing a transition consumes tokens from its input places, assuming that their ages fit into the intervals on the input arcs to the transition; a missing interval stands for $[0, \infty)$. Transition firing then produces fresh 0-age tokens to all output places. The net also contains inhibitor arcs with circle-pointed arrow heads. For example the place p_1 is connected by an inhibitor arc to the transition t_3 and the presence of a token in p_1 in the initial marking inhibits the possibility to fire t_3.

In Figure 1b we can see the reachable state space of the game net. Each state (marking) is annotated with the tokens and their ages, e.g. $p_4 : 1 + p_7 : 0$ is a marking where p_4 contains a token of age 1 and p_7 contains a token of 0. Solid arrows represent player 1 and delay actions, and dashed arrows represent player 2 actions. The objective of the controller (player 1) is to place a token to the place p_8. Indeed, the controller has a *winning strategy* to reach this goal by initially playing the transition t_1 after which, irrespective of the behaviour of the environment (player 2), the target marking is reached. If instead the controller plays from the initial marking the transition t_2, the environment can enforce to reach the empty marking $\emptyset$ with no tokens, which is clearly not a winning strategy for the controller.

The grayed out arrows and markings can be pruned during the state space exploration as they do not influence the existence of a winning stratety for the controller. For example, in the marking $p_2{:}0 + p_4{:}0 + p_5{:}0$, we can observe that only player 2 transitions are enabled and time cannot progress either, hence we can apply the classical stubborn set reduction and consider only one of the possible interleavings of the transitions t_4, t_6 and t_8.

Related work. The literature for partial order reductions for both timed systems and games is scarce and generally does not report favourable experimental evaluation, if any at all. For timed systems, early work by Bengtsson et al. [3] and Minea [29] provides partial order reduction methods but no experimental evaluation to show a practical benefit. For Petri nets, various efforts have been made. Examples include one-safe time Petri nets by Yoneda et al. [38]; however, the method is not suitable for larger models [33]. Boucheneb et al. [5,6,7] present a partial order reduction method for timed Petri nets but only report on exper-

imental results for a small amount of examples on a prototype implementation. Lilius [28] suggests a method that allows for applying partial order reductions for untimed systems; however, no experiments are reported. As an exception, partial order reductions have been used successfully for timed systems that exhibit urgent behaviour [8]. Furthermore, we differentiate from other approaches by preserving both winning reachability and safety strategies as well as extending the approach to games.

Similarly, the extent of applying partial order reductions to games has been limited until recent notable developments. Partial order reductions for bisimulation equivalence were presented in [36,22,18]; however, our approach is more general as we allow for reduction in both controllable and uncontrollable states, and we provide an experimental evaluation. Recently, Neele et al. [30] presented partial order reductions for untimed parity games. This method can model check the full modal μ-calculus, where game semantics are encoded as μ-calculus formulae. This, however, includes conditions that may be redundant and weaken the reduction potential in our less general setting where we preserve the existence of winning two-player reachability and safety strategies [9,10]. Lastly, we preserve winning reachability and safety strategies in the presence of time, which is not yet achieved by other approaches.

2 Preliminaries

Definition 1 (Timed Game Labelled Transition System). *A Timed Game Labelled Transition System (TGLTS) is a tuple* $G = (\mathcal{S}, A_1, A_2, \rightarrow, \mathit{Goal})$ *where* $\mathcal{S}$ *is a set of states,* $A = A_1 \cup A_2$ *is a set of actions divided into a finite set of player 1 actions* A_1 *and a finite set of player 2 actions* A_2 *where* $A_1 \cap A_2 = \emptyset$*,* $\rightarrow \subseteq \mathcal{S} \times (A \cup \mathbb{R}^{\geq 0}) \times \mathcal{S}$ *is a determinisitic transition relation s.t. if* $(s, \alpha, s') \in \rightarrow$ *and* $(s, \alpha, s'') \in \rightarrow$ *then* $s' = s''$*, and* $\mathit{Goal} \subseteq \mathcal{S}$ *is a set of goal states where for all* $s, s' \in \mathcal{S}$ *if* $(s, d, s') \in \rightarrow$ *and* $d \in \mathbb{R}^{\geq 0}$ *then* $s \in \mathit{Goal}$ *iff* $s' \in \mathit{Goal}$*.*

Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, \mathit{Goal})$ be a fixed TGLTS for the remainder of the section. Whenever $(s, \alpha, s') \in \rightarrow$ we write $s \xrightarrow{\alpha} s'$. The set of *enabled* player actions at a state $s \in \mathcal{S}$ is defined as $en(s) = \{a \in A \mid \exists s' \in \mathcal{S}.\ s \xrightarrow{a} s'\}$. The set of *enabled* player 1 actions in a state $s \in \mathcal{S}$ is given by $en_1(s) = en(s) \setminus A_2$, and similarly for player 2. Alternatively, we also refer to player 1 and 2 as the *controller* and the *environment*, respectively. 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. A state s is a *deadlock* state if for all delays $d \in \mathbb{R}^{\geq 0}$ s.t. $s \xrightarrow{d} s'$ we have $en(s') = \emptyset$. The TGLTS G is called non-mixed if all states are either player 1, player 2, or deadlock states.

A state $s \in \mathcal{S}$ is *zero time* if time can not elapse at s. We denote the zero time property of a state s by the predicate $\mathsf{zt}(s)$ and define it as $\mathsf{zt}(s)$ iff for all $s' \in \mathcal{S}$ and all $d \in \mathbb{R}^{\geq 0}$ if $s \xrightarrow{d} s'$ then $d = 0$. We only consider TGLTS that satisfy the following time related axioms for all $s, s', s'' \in \mathcal{S}$ and all $d, d' \in \mathbb{R}^{\geq 0}$:

– If $s \xrightarrow{d} s'$ and $0 \leq d' \leq d$ then there exists $s_{d'} \in \mathcal{S}$ s.t. $s \xrightarrow{d'} s_{d'} \xrightarrow{d-d'} s'$.
– If $s \xrightarrow{d} s'$ and $d = 0$ then $s = s'$.

For a sequence of actions $w = \alpha_1\alpha_2 \cdots \alpha_n \in (A \cup \mathbb{R}^{\geq 0})^*$ we write $s \xrightarrow{w} s'$ if $s \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} s'$ for some $s_1, \ldots, s_{n-1} \in \mathcal{S}$. For an infinite sequence $w \in (A \cup \mathbb{R}^{\geq 0})^\omega$ if for every prefix w' of w there exists $s' \in \mathcal{S}$ s.t. $s \xrightarrow{w'} s'$ then we write $s \xrightarrow{w}$. Actions that are a part of w are said to occur in w.

A (memoryless) strategy is a function which proposes the next move either player 1 or player 2 wants to make, where λ represents the delay action.

Definition 2 (Strategy). *Let $G = (\mathcal{S}, A, \rightarrow, Goal)$ be a TGLTS. A* strategy *for a player i where $i \in \{1, 2\}$ is a function $\sigma_i : \mathcal{S} \rightarrow (\mathbb{R}^{\geq 0} \times A_i) \cup \{\lambda\}$ where for all $s \in \mathcal{S}$ we have:*

– *If $\sigma_i(s) = \lambda$ then for all $d \in \mathbb{R}^{\geq 0}$ where $s \xrightarrow{d} s'$ we have $\sigma_i(s') = \lambda$, and*
  - *either for all $d \in \mathbb{R}^{\geq 0}$ there exists $s'' \in \mathcal{S}$ s.t. $s \xrightarrow{d} s''$, or*
  - *there exists $d \in \mathbb{R}^{\geq 0}$ s.t. $s \xrightarrow{d} s'$ and for all $d' \in \mathbb{R}^{\geq 0}$ where $s' \xrightarrow{d'} s''$ we have $en_i(s'') = \emptyset$.*
– *If $\sigma_i(s) = (d, a)$ then there exists $s' \in \mathcal{S}$ s.t. $s \xrightarrow{da} s'$.*

Let Σ_G^1 (ranged over by σ_1) and Σ_G^2 (ranged over by σ_2) be the set of all possible strategies for player 1 and player 2 for a TGLTS G, respectively.

A run $\pi = s_0 s_1 \cdots \in \mathcal{S}^* \cup \mathcal{S}^\omega$ is a (possibly infinite) sequence of states where for all $i \geq 0$ there exists $d \in \mathbb{R}^{\geq 0}$ and $a \in A$ s.t. $s_i \xrightarrow{da} s_{i+1}$. The length of a run π (number of actions in the run) is given by $\ell(\pi)$ where $\ell(\pi) = \infty$ if $\pi \in \mathcal{S}^\omega$ and otherwise $\ell(\pi) = n$ where $\pi = s_0 \cdots s_n$. Let $\mathbb{N}^0 = \mathbb{N} \cup \{0\}$. A position in a run $\pi = s_0 s_1 \ldots \in \mathcal{S}^* \cup \mathcal{S}^\omega$ 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 $\mathit{\Pi}^{max}_{G,\sigma_1,\sigma_2}(s)$ be the set of maximal runs subject to σ_1 and σ_2 starting at $s \in \mathcal{S}$ defined as follows: $\pi \in \mathit{\Pi}^{max}_{\sigma_1,\sigma_2}(s)$ iff $\pi_0 = s$ and for all $0 \leq i \leq \ell(\pi)$ (or $0 \leq i$ if π is infinite) we either have:

– $\sigma_1(\pi_i) = (d_1, a_1)$ and $(\sigma_2(\pi_i) = \lambda$ or $\sigma_2(\pi_i) = (d_2, a_2)$ and $d_1 \leq d_2)$ and $\pi_i \xrightarrow{d_1 a_1} \pi_{i+1}$.
– $\sigma_2(\pi_i) = (d_2, a_2)$ and $(\sigma_1(\pi_i) = \lambda$ or $\sigma_1(\pi_i) = (d_1, a_1)$ and $d_2 \leq d_1)$ and $\pi_i \xrightarrow{d_2 a_2} \pi_{i+1}$.
– $\sigma_1(\pi_i) = \sigma_2(\pi_i) = \lambda$ and $\pi_i = \pi_{\ell(\pi)}$.

Let $\mathit{\Pi}^{max}_{G,\sigma_1}(s) = \bigcup_{\sigma_2 \in \Sigma_G^2} \mathit{\Pi}^{max}_{G,\sigma_1,\sigma_2}(s)$ be the set of all maximal runs subject to σ_1 and any possible player 2 strategy, and similarly for player 2. We omit the TGLTS G when possible from the subscript of $\mathit{\Pi}^{max}_{G,\sigma_1,\sigma_2}(s)$ for simplicity.

Definition 3 (Winning Strategy). *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, Goal)$ be a TGLTS and $s \in \mathcal{S}$ be a state. A strategy $\sigma_1 \in \Sigma_G^1$ is a* winning (reachability) strategy *for player 1 at s in G iff for all $\sigma_2 \in \Sigma_G^2$ and for all $\pi \in \mathit{\Pi}^{max}_{\sigma_1,\sigma_2}(s)$ there exists a position i s.t. $\pi_i \in Goal$. A state s is* winning *(for player 1) if there exists a winning strategy for player 1 at s in G.*

Remark: We say that a state s is winning for player 2 iff s is not winning for player 1. Similarly, a strategy for player 2 is a winning strategy at s iff there does not exist a winning strategy for player 1 at s.

3 Stable Stubborn Reduction

We now introduce the central concept of *stable reductions* [10] extended to handle timed games. First, we introduce *interesting sets* and *safe actions*.

Definition 4 (Interesting Set of Actions). *Let $G = (\mathcal{S}, A, \rightarrow, Goal)$ be a TGLTS, $s \in \mathcal{S}$ a state, and $X \subseteq \mathcal{S}$ a set of target states. A set of actions $A_s(X) \subseteq A$ is called an* interesting set *of actions for $s \notin X$ if whenever $w = a_1 \cdots a_n \in A^*$, $s \xrightarrow{w} s'$, and $s' \in X$ then there exists $1 \leq i \leq n$ s.t. $a_i \in A_s(X)$. If $s \in X$ then $A_s(X) = \emptyset$.*

Informally, for any run from s to a target state $s' \in X$ (with $s \notin X$) there must be at least one interesting action. For a given set of target states there may be several possible interesting sets of actions and in the rest of this paper we assume that one such set of interesting actions is fixed.

Let us also restate the notion of player 1 safe actions from [9,10]. Intuitively, we require for a safe player 1 action a in a state s that the addition of a to the beginning of an execution sequence does not hand over control to player 2 earlier than otherwise.

Definition 5 (Safe Action). *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, Goal)$ be a TGLTS and $s \in \mathcal{S}$ a player 1 state such that $en_2(s) = \emptyset$. An action $a \in A_1 \cap en_1(s)$ is safe in s if whenever $w \in (A_1 \setminus \{a\})^*$ 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 $safe(s)$.*

A reduced game is defined by a function called a *reduction* that for each state proposes the set of actions to consider at that state.

Definition 6 (Reduction). *Let $G = (\mathcal{S}, A, \rightarrow, Goal)$ be a TGLTS. A* reduction *is a function $St : \mathcal{S} \rightarrow 2^A$.*

Definition 7 (Reduced Game). *Let $G = (\mathcal{S}, A, \rightarrow, Goal)$ be a TGLTS and St be a reduction. The* reduced game *of G by the reduction St is given by $G_{St} = (\mathcal{S}, A, \xrightarrow[St]{}, Goal)$ where $s \xrightarrow[St]{\alpha} s'$ iff $s \xrightarrow{\alpha} s'$ and $\alpha \in St(s) \cup \mathbb{R}^{\geq 0}$.*

The set of states $St(s)$ is the stubborn set of s with the reduction St. The set of non-stubborn actions for s is defined as $\overline{St(s)} = A \setminus St(s)$. We shall now present the key definition of a stable reduction.

Definition 8 (Stable Reduction Conditions). *A reduction St is called* stable *if St satisfies for every $s \in S$ Conditions **I**, **Z**, **W**, **R**, **T**, **G1**, **G2**, **S**, **V**, and **D**. For a stable reduction St we call $St(s)$ a* stable stubborn set *(of s).*

I *If* $en_1(s) \neq \emptyset$ *and* $en_2(s) \neq \emptyset$ *then* $en(s) \subseteq St(s)$.
Z *If* $\neg \mathsf{zt}(s)$ *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 $A_s(Goal) \subseteq St(s)$.
T $A_s(\{s \in \mathcal{S} \mid \neg \mathsf{zt}(s)\}) \subseteq St(s)$.
G1 *If* $en_2(s) = \emptyset$ *then* $A_s(\{s \in \mathcal{S} \mid en_2(s) \neq \emptyset\}) \subseteq St(s)$.
G2 *If* $en_1(s) = \emptyset$ *then* $A_s(\{s \in \mathcal{S} \mid en_1(s) \neq \emptyset\}) \subseteq St(s)$.
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* $a \in en_2(s')$.

Conditions **I** and **Z** ensures that all enabled actions are included in the reduction if a state is either non-urgent or mixed. Condition **W** ensures that a stubborn action can commute with any sequence of nonstubborn actions. Condition **R** prevents goal states from being reachable by exploring only nonstubborn actions. In other words, any sequence of actions leading to a goal state must include at least one stubborn action. Conditions **T**, **G1**, and **G2** are similar to Condition **R**. They ensure that reachability of certain states where either time can elapse (**T**, or the opposing player is allowed to make a move (**G1** and **G2**) are preserved. Condition **S** states that either all enabled stubborn player 1 actions are safe, and otherwise 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. If this is possible, then all enabled player 2 actions are included in the stubborn set. Condition **D** ensures at least one player 2 action cannot be disabled by exploring only nonstubborn actions. This condition preserves cycles and runs of exclusively player 2 actions to deadlocks.

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

Theorem 1 (Reachability Strategy Preservation for TGLTS). *Let* $G = (\mathcal{S}, A_1, A_2, \rightarrow)$ *be a TGLTS 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}.

Furthermore, we can show that slightly modified stable reductions also preserve *winning safety strategies* for both players, which we define as follows.

Definition 9 (Winning Safety Strategy). *Let* $G = (\mathcal{S}, A_1, A_2, \rightarrow, Goal)$ *be a TGLTS and* $s \in \mathcal{S}$ *be a state. A strategy* $\sigma_1 \in \Sigma_G^1$ *is a* winning safety strategy *for player* 1 *at* s *in* G *iff for all* $\sigma_2 \in \Sigma_G^2$, *for all* $\pi \in \Pi_{\sigma_1,\sigma_2}^{max}(s)$, *and for all positions* i *we have* $\pi_i \notin Goal$. *A state* s *is a* winning state *(for player* 1 *and a safety objective) if there exists a winning safety strategy for player* 1 *at* s *in* G.

To accommodate safety, we introduce the following set of modified stable reduction conditions.

V′ If there exists $w \in A_1^*$ s.t. $s \xrightarrow{w} s'$ and $s' \in Goal$ then $en_1(s) \subseteq St(s)$.
D′ If $en_1(s) \neq \emptyset$ then there exists $a \in en_1(s) \cap St(s)$ s.t. for all $w \in \overline{St(s)}^*$ where $s \xrightarrow{w} s'$ we have $a \in en_1(s')$.

We can then state the second main theorem showing that our modified stable reduction preserves the winning safety strategies of both players.

Theorem 2 (Safety Strategy Preservation for TGLTS). *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow)$ be a TGLTS and St a stable reduction with Conditions $\boldsymbol{V}$ and $\boldsymbol{D}$ replaced by $\boldsymbol{V'}$ and $\boldsymbol{D'}$. A state $s \in \mathcal{S}$ is a winning state with a safety objective in G for player 1 iff s is a winning state with a safety objective in G_{St} for player 1.*

4 Stable Reductions on Timed-Arc Petri Net Games

We now introduce the formalism of *Timed-Arc Petri net Games* (TAPG) [25]. Let $\mathbb{N}^\infty = \mathbb{N} \cup \{\infty\}$. We define the set of *well-formed closed time intervals* as $\int \stackrel{\text{def}}{=} \{[a,b] \mid a \in \mathbb{N}^0 \wedge b \in \mathbb{N}^\infty \wedge a \leq b\}$ and its subset $\int^{\text{inv}} \stackrel{\text{def}}{=} \{[0,b] \mid b \in \mathbb{N}^\infty\}$ used in age invariants.

Definition 10 (Timed-Arc Petri Net Game [25]). *A* timed-arc Petri net game *(TAPG) is a 9-tuple $N = (P, T_1, T_2, T_{urg}, IA, OA, g, w, Type, I)$ where*

- *P is a finite set of* places*,*
- *T_1 and T_2 are finite sets of controller and environment* transitions*, respectively, such that $T_1 \cap T_2 = \emptyset$, $T = T_1 \cup T_2$ and $P \cap T = \emptyset$,*
- *$T_{urg} \subseteq T$ is the set of* urgent transitions*,*
- *$IA \subseteq P \times T$ is a finite set of* input arcs*,*
- *$OA \subseteq T \times P$ is a finite set of* output arcs*,*
- *$g : IA \rightarrow \int$ is a* time constraint function *assigning guards (time intervals) to input arcs s.t.*
  - *if $(p,t) \in IA$ and $t \in T_{urg}$ then $g((p,t)) = [0,\infty]$,*
- *$w : IA \cup OA \rightarrow \mathbb{N}$ is a function assigning* weights *to input and output arcs,*
- *$Type : IA \cup OA \rightarrow$ **Types** is a* type function *assigning a type to all arcs where $\textbf{Types} = \{Normal, Inhib\} \cup \{Transport_j \mid j \in \mathbb{N}\}$ such that*
  - *if $Type(z) = Inhib$ then $z \in IA$ and $g(z) = [0,\infty]$,*
  - *if $Type((p,t)) = Transport_j$ for some $(p,t) \in IA$ then there is exactly one $(t,p') \in OA$ such that $Type((t,p')) = Transport_j$,*
  - *if $Type((t,p')) = Transport_j$ for some $(t,p') \in OA$ then there is exactly one $(p,t) \in IA$ such that $Type((p,t)) = Transport_j$,*
  - *if $Type((p,t)) = Transport_j = Type((t,p'))$ then $w((p,t)) = w((t,p'))$,*
- *$I : P \rightarrow \int^{inv}$ is a function assigning* age invariants *to places.*

Note that for transport arcs we assume that they come in pairs (for each type $Transport_j$) and that their weights match. Also for inhibitor arcs and for input arcs to urgent transitions, we require that the guards are $[0,\infty]$.

Before we give the formal semantics of the model, let us fix some notation. Let $N = (P, T_1, T_2, T_{urg}, IA, OA, g, w, Type, I)$ be a TAPG for the rest of the section. We denote by ${}^\bullet x \stackrel{\text{def}}{=} \{y \in P \cup T \mid (y,x) \in IA \cup OA \wedge Type((y,x)) \neq Inhib\}$ the *preset* of a transition or a place x. Similarly, the *postset* is defined

as $x^\bullet \stackrel{\text{def}}{=} \{y \in P \cup T \mid (x,y) \in IA \cup OA \wedge \mathit{Type}((x,y)) \neq I\}$. We denote by ${}^\circ t \stackrel{\text{def}}{=} \{p \in P \mid (p,t) \in IA \wedge \mathit{Type}((p,t)) = \mathit{Inhib}\}$ the *inhibitor preset* of a transition t. The *inhibitor postset* of a place p is defined as $p^\circ \stackrel{\text{def}}{=} \{t \in T \mid (p,t) \in IA \wedge \mathit{Type}((p,t)) = \mathit{Inhib}\}$. For a place $p \in P$ we define the *increasing preset* of p as ${}^+p \stackrel{\text{def}}{=} \{t \in {}^\bullet p \mid w((t,p)) > w((p,t))\}$, and similarly the *decreasing postset* of p as $p^- \stackrel{\text{def}}{=} \{t \in p^\bullet \mid w((t,p)) < w((p,t))\}$. For a transition $t \in T$ we define the *decreasing preset* of t as ${}^-t \stackrel{\text{def}}{=} \{p \in {}^\bullet t \mid w((t,p)) < w((p,t))\}$, and similarly the *increasing postset* of t as $t^+ \stackrel{\text{def}}{=} \{p \in t^\bullet \mid w((t,p)) > w((p,t))\}$. For a set $X \subseteq P \cap T$ we extend the notation as ${}^\bullet X = \bigcup_{x \in X} {}^\bullet x$, and similarly for the other operators. Let $\mathcal{B}(\mathbb{R}^{\geq 0})$ be the set of all finite multisets over $\mathbb{R}^{\geq 0}$. A *marking* M on N is a function $M : P \longrightarrow \mathcal{B}(\mathbb{R}^{\geq 0})$ where for every place $p \in P$ and every token $x \in M(p)$ we have $x \in I(p)$, in other words all tokens have to satisfy the age invariants. The set of all markings in a net N is denoted by $\mathcal{M}(N)$.

We write (p,x) to denote a token at a place p with the age $x \in \mathbb{R}^{\geq 0}$. Then $M = \{(p_1,x_1),(p_2,x_2),\ldots,(p_n,x_n)\}$ is a multiset representing a marking M with n tokens of ages x_i in places p_i. We define the size of a marking as $|M| = \sum_{p \in P} |M(p)|$ where $|M(p)|$ is the number of tokens located in the place p. A marked TAPG (N, M_0) is a TAPG N together with an initial marking M_0 with all tokens of age 0.

Definition 11 (Enabledness). *Let $N = (P, T_1, T_2, T_{urg}, IA, OA, g, w, \mathit{Type}, I)$ be a TAPG. We say that a transition $t \in T$ is* enabled *in a marking M by the multisets of tokens $In = \{(p,x_p^1),(p,x_p^2),\ldots,(p,x_p^{w((p,t))}) \mid p \in {}^\bullet t\} \subseteq M$ and $Out = \{(p',x_{p'}^1),(p',x_{p'}^2),\ldots,(p',x_{p'}^{w((t,p'))}) \mid p' \in t^\bullet\}$ if*

- *for all input arcs except the inhibitor arcs, the tokens from In satisfy the age guards of the arcs, i.e.*

$$\forall p \in {}^\bullet t.\ x_p^i \in g((p,t)) \text{ for } 1 \leq i \leq w((p,t))$$

- *for any inhibitor arc pointing from a place p to the transition t, the number of tokens in p is smaller than the weight of the arc, i.e.*

$$\forall (p,t) \in IA.\, \mathit{Type}((p,t)) = I \Rightarrow |M(p)| < w((p,t))$$

- *for all input arcs and output arcs which constitute a transport arc, the age of the input token must be equal to the age of the output token and satisfy the invariant of the output place, i.e.*

$$\forall (p,t) \in IA. \forall (t,p') \in OA.\, \mathit{Type}((p,t)) = \mathit{Type}((t,p')) = \mathit{Transport}_j \\ \Rightarrow \left(x_p^i = x_{p'}^i \wedge x_{p'}^i \in I(p')\right) \text{ for } 1 \leq i \leq w((p,t))$$

- *for all normal output arcs, the age of the output token is 0, i.e.*

$$\forall (t,p') \in OA.\, \mathit{Type}((t,p')) = \mathit{Normal} \Rightarrow x_{p'}^i = 0 \text{ for } 1 \leq i \leq w((t,p')).$$

A TAPG $N = (P, T_1, T_2, T_{urg}, IA, OA, g, w, Type, I)$ defines a GLTS $G(N) = (\mathcal{S}, A_1, A_2, \rightarrow, Goal)$ 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, $Goal \subseteq \mathcal{M}(N)$ is a subset of markings, and the transition relation is defined as follows:

- If $t \in T$ is enabled in a marking M by the multisets of tokens In and Out then t can *fire* and produce the marking $M' = (M \setminus In) \uplus Out$ where $\uplus$ is the multiset sum operator and $\setminus$ is the multiset difference operator; we write $M \xrightarrow{t} M'$ for this action transition.
- A time *delay* $d \in \mathbb{N}^0$ is allowed in M if
  - $(x + d) \in I(p)$ for all $p \in P$ and all $x \in M(p)$, i.e. by delaying d time units no token violates any of the age invariants, and
  - if $M \xrightarrow{t} M'$ for some $t \in T_{urg}$ then $d = 0$, i.e. enabled urgent transitions disallow time passing.

  By delaying d time units in M we reach the marking M' defined as $M'(p) = \{x + d \mid x \in M(p)\}$ for all $p \in P$; we write $M \xrightarrow{d} M'$ for this delay transition.

For defining the set of goal markings *Goal* we present a Boolean logic over marking expressions. Let e_1 and e_2 be two marking expressions of N and let φ be a formulae with the following syntax:

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

where $t \in T$ and $\bowtie \in \{<, \leq, =, \neq, >, \geq\}$. Let E_N be defined by the following syntax: $e ::= c \mid p \mid e_1 \oplus e_2$, where $c \in \mathbb{N}^0$, $p \in P$, and $\oplus \in \{+, -, *\}$. We evaluate a marking expression e relative to a marking $M \in \mathcal{M}(N)$ by the function $eval_M(e)$ where $eval_M(c) = c$, $eval_M(p) = |M(p)|$ and $eval_M(e_1 \oplus e_2) = eval_M(e_1) \oplus eval_M(e_2)$.

The semantics for the satisfability relation $M \models \varphi$ for a marking $M \in \mathcal{M}(N)$ and a formula φ is given in the standard way for the Boolean connectives and in particular $M \models t$ iff $t \in en(M)$, $M \models deadlock$ iff $en(M) = \emptyset$, and $M \models e_1 \bowtie e_2$ iff $eval_M(e_1) \bowtie eval_M(e_2)$.

Given a formula φ, we aim to preserve at least one run to the set of goal markings $Goal = \{M \in \mathcal{M}(N) \mid M \models \varphi\}$. To achieve this, we use the definition of interesting transitions $A_M(\varphi)$ relative to a formula φ defined in [10]. Lemma 1 shows that $A_M(\varphi)$ is indeed an interesting set of transitions, and is sufficient to preserve Condition **R**.

Lemma 1 ([10]). *Let $N = (P, T_1, T_2, T_{urg}, IA, OA, g, w, Type, I)$ be a TAPG, $M \in \mathcal{M}(N)$ a marking, and φ 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 use the overapproximation algorithm $reach(N, M, \varphi)$ presented in [10] to overapproximate Condition **V**. The algorithm returns *true* whenever there is a sequence of player 2 transtions that leads us from a marking $M \in \mathcal{M}(N)$ to a goal marking M' where $M' \models \varphi$ in the TAPG N, and otherwise it returns *false*. The desired property is stated in Lemma 2. For handling Condition $\mathbf{V}'$ we simply switch all instances of environment transitions T_2 with the controller transitions T_1 in $reach(N, M, \varphi)$ and Lemma 2.

Lemma 2 ([10]). *Let $N = (P, T_1, T_2, T_{urg}, IA, OA, g, w, Type, I)$ be a Petri net game, $M \in \mathcal{M}(N)$ a marking on N and φ a formula. If there is $w \in T_2^*$ s.t. $M \xrightarrow{w} M'$ and $M' \models \varphi$ then $reach(N, M, \varphi) = true$.*

Before we can state our main theorem, we need a method for determining safe transitions. This can be done by analysing the increasing presets and postsets of transitions as demonstrated in the following Lemma 3 from [10] that requires a small adaptation to our setting.

Lemma 3 ([10]). *Let $N = (P, T_1, T_2, T_{urg}, IA, OA, g, w, Type, I)$ be a TAPG 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.*

We can now state our main contribution in Theorem 3. The theorem provides a list of syntactic conditions on TAPG that for a given marking generates a stable stubborn set.

Theorem 3 (Stable Reduction Preserving Closure). *Let $N = (P, T_1, T_2, T_{urg}, IA, OA, g, w, Type, I)$ be a TAPG, 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 $T \subseteq St(M)$.*
2. *If $\neg \mathsf{zt}(M)$ then $T \subseteq St(M)$.*
3. *If $\mathsf{zt}(M)$ then either*
   (a) *there is $t \in T_{urg} \cap en(M) \cap St(M)$ where ${}^\bullet({}^\circ t) \subseteq St(M)$, or*
   (b) *there is $p \in P$ where $I(p) = [a, b]$ and $b \in M(p)$ such that $t \in St(M)$ for every $t \in p^\bullet$ where $b \in g((p, t))$.*
4. *If $en_1(M) \cap St(M) \not\subseteq safe(M)$ then $T \subseteq St(M)$.*
5. *$A_M(\varphi) \subseteq St(M)$*
6. *If $en_1(M) = \emptyset$ then $T_1 \subseteq St(M)$.*
7. *If $en_2(M) = \emptyset$ then $T_2 \subseteq St(M)$.*
8. *For all $t \in St(M) \setminus en(M)$ either*
   (a) *there is $p \in {}^\bullet t$ such that $|\{x \in M(p) \mid x \in g((p, t))\}| < w((p, t))$ and*
      - *$t' \in St(M)$ for all $t' \in {}^+ p$ where there is $p' \in {}^- t'$ with $Type((t', p)) = Type((p', t')) = Transport_j$ and where $g((p', t')) \cap g((p, t)) \neq \emptyset$, and*
      - *if $0 \in g((p, t))$ then also ${}^\bullet p \subseteq St(M)$, or*
   (b) *there is $p \in {}^\circ t$ where $|M(p)| \geq w((p, t))$ such that*
      - *$t' \in St(M)$ for all $t' \in p^-$ where $M(p) \cap g((p, t')) \neq \emptyset$.*
9. *For all $t \in St(M) \cap en(M)$ we have*
   (a) *$t' \in St(M)$ for every $t' \in p^\bullet$ where $p \in {}^\bullet t$ and $g((p, t)) \cap g((p, t')) \neq \emptyset$, and*
   (b) *$(t^\bullet)^\circ \subseteq St(M)$.*
10. *If $en_2(M) \neq \emptyset$ then there exists $t \in en_2(M) \cap St(M)$ s.t. $\{t' \in ({}^\bullet t)^\bullet \mid \exists p \in {}^\bullet t \cup {}^\bullet t' \wedge g((p, t')) \cap g((p, t)) \cap M(p) \neq \emptyset\} \cup {}^+({}^\circ t) \subseteq St(M)$.*
11. *If $en_1(M) = \emptyset$ and $reach(N, M, \varphi) = true$ then $en(M) \subseteq St(M)$.*

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

Algorithm 1: Computation of $St(M)$ for some stable reduction St.

input : A TAPG $N = (P, T_1, T_2, T_{urg}, IA, OA, g, w, Type, I)$ and $M \in \mathcal{M}(N)$ and formula φ
output : $X \subseteq T$ where X is a stable stubborn set for M

1 **if** $en(M) = \emptyset$ **then return** T;
2 **if** $\neg \mathtt{zt}(M)$ **then return** T;
3 **if** $en_1(M) \neq \emptyset \wedge en_2(M) \neq \emptyset$ **then return** T;
4 $Y := \emptyset$;
5 **if** $en_1(M) = \emptyset$ **then**
6 | **if** $reach(N, M, \varphi)$ **then return** T;
7 | Pick any $t \in en_2(M)$; $Y := T_1 \cup t \cup {}^{+}({}^{\circ}t) \cup$
| $\{t' \in ({}^{\bullet}t)^{\bullet} \mid \exists p \in {}^{\bullet}t \cup {}^{\bullet}t' \wedge g((p,t')) \cap g((p,t)) \cap M(p) \neq \emptyset\}$;
8 **else**
9 | $Y := T_2$;
10 **if** $T_{urg} \cap en(M) \neq \emptyset$ **then**
11 | pick any $t \in T_{urg} \cap en(M)$;
12 | $Y := Y \cup \{t\} \cup {}^{\bullet}({}^{\circ}t)$;
13 **else**
14 | pick any $p \in P$ where $I(p) = [a, b]$ and $b \in M(p)$
15 | **forall** $t \in p^{\bullet}$ **do**
16 | | **if** $b \in g((p,t))$ **then** $Y := Y \cup \{t\}$;
17 $Y := Y \cup A_M(\varphi)$; $X := Saturate(Y)$;
18 **if** $X \cap en_1(M) \not\subseteq safe(M)$ **then return** T;
19 **return** X;

In Algorithm 1 we now provide a pseudocode for calculating stable stubborn sets for a given marking. The algorithm calls Algorithm 2 that saturates a given set to satisfy Conditions 8 and 9.

Theorem 4. *Algorithm 1 terminates and returns $St(M)$ for some stable reduction St.*

To preserve safety strategies, we can modify Algorithm 1 slightly as well as $reach(N, M, \varphi)$ presented in [10]. The modified algorithm returns *true* whenever there is a sequence of player 1 transtions (instead of player 2 transitions) that leads us from a marking $M \in \mathcal{M}(N)$ to a goal marking M' where $M' \models \varphi$ in the TAPG N. To satisfy Condition $\mathbf{V}'$ we move the check at Line 6 into the else block at Line 9. For Condition $\mathbf{D}'$, we move the assignment at Line 7 also into the else block at Line 9.

5 Implementation and Experiments

We extend the timed-arc Petri net game synthesis engine `verifydtapn` of TAPAAL [26,25] with the implementation of our stubborn set reduction for timed games and evaluate it on the following case studies.

Algorithm 2: *Saturate*(Y)

```
1  X := ∅;
2  while Y ≠ ∅ do
3      pick any t ∈ Y;
4      if t ∉ en(M) then
5          if ∃p ∈ •t. |{x ∈ M(p) | x ∈ g((p,t))}| < w((p,t)) then
6              pick any such p;
7              if 0 ∈ g((p,t)) then
8                  Y := Y ∪ (•p \ X);
9              else
10                 forall t' ∈ {t'' ∈ +p \ X | Type((t'',p)) = Transport_j} do
11                     forall p' ∈ {p'' ∈ −t' | Type((p'',t')) = Type((t',p))} do
12                         if g((p',t')) ∩ g((p,t)) ≠ ∅ then Y := Y ∪ {t'};
13         else
14             pick any p ∈ °t s.t. |M(p)| ≥ w((p,t));
15             forall t' ∈ p− \ X do
16                 if M(p) ∩ g((p,t')) ≠ ∅ then Y := Y ∪ {t'};
17     else
18         forall p ∈ •t do Y := Y ∪ ({t' ∈ p•|g((p,t)) ∩ g((p,t')) ≠ ∅} \ X);
19         Y := Y ∪ ((t•)° \ X);
20     Y := Y \ {t}; X := X ∪ {t};
21 return X ∩ en(M);
```

- The *Fire Alarm* (FireAlarm-N-game) models a fire alarm system developed by a German company [15,16]. We scale the model by the number of wireless sensors N which report to a central unit. The objective of the game is to ensure the central unit acknowledges the messages of the sensors in the presence of a jammer which can cause message loss.
- The *Blood Transfusion* (BloodTransfusion-N-game) case study models a larger blood transfusion workflow [11], adapted to a timed game. We scale the model by the number of patients N receiving blood transfusions. The goal of the controller is to make sure all patients successfully finish the blood transfusion process on schedule.
- The *Limfjord* (Limfjord-N-K-S) models one direction of the Limfjord lifting bridge that connects the cities of Aalborg and Nørresundby in Northern Jutland, Denmark. We scale the model by the number of lanes N available, the number of cars K crossing the bridge, and the allowed interval of time S for all the cars to cross the bridge. The environment may temporarily either close a lane or raise the bridge to allow for boat traffic. The objective of the controller is to ensure all cars cross the bridge within the time limit.
- The *Railway Scheduling Problem* (LyngbySmall-N) is a model of a smaller variant of the Danish train station Lyngby. The problem and the station layout was initially described in [27]. We scale the model by the number of

	Time (seconds)		Markings ×1000		Improvement	
Model	NORMAL	POR	NORMAL	POR	Time	Markings
FireAlarm-10-game	11.24	3.75	796	498	3.00	1.60
FireAlarm-12-game	52.86	4.55	1727	526	11.62	3.28
FireAlarm-14-game	376.26	7.58	5367	554	49.64	9.69
FireAlarm-16-game	3868.19	8.75	19845	582	442.08	34.10
FireAlarm-100-game	>3 hours	255.80	to	1844	-	-
BloodTransfusion-3-game	3.91	3.45	612	504	1.13	1.21
BloodTransfusion-4-game	96.05	72.70	11864	8118	1.32	1.46
BloodTransfusion-5-game	2323.58	1329.11	196534	111089	1.75	1.77
Limfjord-1-6-12-game	2.48	1.11	212	75	2.23	2.83
Limfjord-1-6-20-game	4.10	1.74	423	156	2.36	2.71
Limfjord-1-10-15-game	5.82	2.16	896	336	2.69	2.67
Limfjord-1-10-20-game	9.18	2.79	1380	468	3.29	2.95
Limfjord-1-14-20-game	18.23	5.51	2550	864	3.31	2.95
Limfjord-1-14-25-game	27.98	8.10	3799	1230	3.45	3.09
Limfjord-2-6-12-game	1119.47	280.71	87718	24099	3.99	3.64
LyngbySmall-2-game	0.49	0.21	41	20	2.33	2.05
LyngbySmall-3-game	2.55	2.38	198	177	1.07	1.12
LyngbySmall-4-game	23.90	85.67	1524	4211	-3.58	-2.76
Covid-9-3-3-900-game	0.42	0.38	112	103	1.11	1.09
Covid-9-4-3-900-game	3.43	3.25	823	697	1.06	1.18
Covid-9-6-3-900-game	164.99	145.46	23609	19296	1.13	1.22

Table 1: Experiments with (POR) and without (NORMAL) partial order reduction

trains (N) entering the station. The controller's goal is to ensure that the trains reach their designated destinations without colliding.

– The *Covid-19 Spreading* (Covid-N-C-I-T-game) models activities of N persons in an indoor area. The goal of the controller is to commute C persons from one room to another via two lobbies and a corridor within T time units while keeping person infections below I. It is not possible to maintain social distancing at the corridor where Covid-19 exposures can occur.

All experiments are run on AMD EPYC 7642 processors with hyper-threading disabled, limited to 30 GB of memory, and a time out of 3 hours. The source code of our implementation is available at [24]. For all the experiments, we use a depth-first search order.

Table 1 shows the experimental evaluation both without (NORMAL) and with (POR) partial order reduction. We report the time in seconds and the number of unique explored markings (in thousands). We also show the relative gain and loss of using partial order reduction for both time and unique markings. The results show a significant potential of our approach. In the FireAlarm-N-game models, there is an exponential speed-up with partial order reductions. We can handle up to 16 sensors without partial order reduction before the 3 hour time out is reached. With partial order reduction, we can verify all our instances of the model up to 100 sensors, and we observe an increasing reduction grow-

ing from being 3 times faster in the 10-sensor instance to more than 422 faster in the 16-sensors instance. In the BloodTransfusion-N-game case study, we see that as the number of patients increases, partial order reduction begins to show increasing savings in both time and number of unique markings explored. We observe that both the number of markings explored and the time used for exploration almost halves in the largest instance. A similar tendency can be observed in the Limfjord-K-N models where the benefit of using POR increases with the problem-size. For the LyngbySmall-N models, while we initially see a speed-up with 2 and 3 trains, partial order reduction with 4 trains becomes, as the only case in our experiments, disadvantageous. Notably, in contrast to the untimed version of the model seen in [10] where a reduction was achieved, virtually no reduction is possible in the timed game variants, leaving only the overhead of calculating stubborn sets. Furthermore, partial order reduction explored more unique markings, which indicates that the reduction changed the search order causing us to explore a larger portion of the state-space. The variants with 2 and 3 trains is the opposite where the search order causes us to explore less of the state-space. We note here that winning strategies exist for the 2 and 3 trains models and not for the 4 trains model. Finally, in the Covid models only moderate reduction in the state-space can be achieved (though improving with the scaling of the model) and the synthesis time improves accordingly, showing that our stubborn set implementation has only a small overhead.

The experiments show that of our approach is generally beneficial and has the potential to achieve exponential speed-up, while having only moderate overhead. The changes in search order may cause large increases in the number of markings explored, as this is usual for on-the-fly verification algorithms.

6 Conclusion

We combined partial order reductions for timed systems and reachability games into a unified framework for timed games on the general formalism timed game labelled transitions systems. This required a new proof of the central theorems to accommodate for the timed setting. Furthermore, we showed that our partial order reduction approach for timed games also preserves winning safety strategies in addition to winning reachability strategies. We instantiated our approach to the formalism of timed-arc Petri net games and suggested specialized overapproximation algorithms. We implemented our approach in the timed-arc Petri net game engine of the TAPAAL model checker suite and evaluated this implementation on a set of scalable case studies. Several of the case studies demonstrate a significant reduction (e.g. the fire alarm case study up to several orders of magnitude in terms of time); the relative time and space reduction is often improving with the increasing scaling of the problems. In the future work, we consider to relax the Conditions **I** and **Z** in order to allow for reductions in mixed and non-urgent states.

References

1. P.A. Abdulla, K. Cerans, B. Jonsson, and Y.K. Tsay. General Decidability Theorems for Infinite-State Systems. In *Symposium on Logic in Computer Science*, LICS'96, page 313–321. IEEE, 1996. doi:10.1109/LICS.1996.561359.
2. R. Alur, T.A. Henzinger, and M.Y. Vardi. Parametric Real-Time Reasoning. In *Symposium on Theory of Computing*, STOC '93, page 592–601. ACM, 1993. doi:10.1145/167088.167242.
3. J. Bengtsson, B. Jonsson, J. Lilius, and W. Yi. Partial Order Reductions for Timed Systems. In *CONCUR*, pages 485–500. Springer Berlin Heidelberg, 1998.
4. T. Bolognesi, F. Lucidi, and S. Trigila. From Timed Petri Nets to Timed LOTOS. In *Proceedings of the IFIP WG 6.1 Tenth International Symposium on Protocol Specification, Testing and Verification X*, page 395–408. North-Holland Publishing Co., 1990. doi:10.5555/645833.670383.
5. H. Boucheneb and K. Barkaoui. Reducing Interleaving Semantics Redundancy in Reachability Analysis of Time Petri Nets. *ACM Trans. Embed. Comput. Syst.*, 12(1):1–24, 2013. ACM. doi:10.1145/2406336.2406343.
6. H. Boucheneb and K. Barkaoui. Stubborn Sets for Time Petri Nets. *ACM Trans. Embed. Comput. Syst.*, 14(1):1–25, 2015. ACM. doi:10.1145/2680541.
7. H. Boucheneb and K. Barkaoui. Delay-Dependent Partial Order Reduction Technique for Real Time Systems. *Real-Time Systems*, 54(2):278–306, 2017. Springer. doi:10.1007/s11241-017-9297-0.
8. 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 Berlin Heidelberg, 2018. doi:10.1007/978-3-319-96145-3_28.
9. F.M. Bønneland, P.G. Jensen, K.G. Larsen, M. Muñiz, and J. Srba. Partial Order Reduction for Reachability Games. In *CONCUR*, volume 140 of *Leibniz International Proceedings in Informatics*, pages 23:1–23:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/LIPIcs.CONCUR.2019.23.
10. F.M. Bønneland, P.G. Jensen, K.G. Larsen, M. Muñiz, and J. Srba. Stubborn Set Reduction for Two-Player Reachability Games. *arXiv preprint arXiv:1912.09875*, 2019.
11. S.C. Christov, G.S. Avrunin, L.A. Clarke, L.J. Osterweil, and E.A. Henneman. A Benchmark for Evaluating Software Engineering Techniques for Improving Medical Processes. In *ICSE Workshop on Software Engineering in Health Care*, SEHC '10, page 50–56. ACM, 2010. doi:10.1145/1809085.1809092.
12. E.M. Clarke, R. Enders, T. Filkorn, and S. Jha. Exploiting Symmetry in Temporal Logic Model Checking. *Formal Methods in System Design*, 9(1):77–104, 1996. Springer Berlin Heidelberg. doi:10.1007/BF00625969.
13. 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 *TACAS*, volume 7214 of *LNCS*, pages 492–497. Springer Berlin Heidelberg, 2012. doi:10.1007/978-3-642-28756-5_36.
14. E.A. Emerson, S. Jha, and D. Peled. Combining Partial Order and Symmetry Reductions. In *Transactions on Petri Nets and Other Models of Concurrency XI*, volume 1217 of *LNCS*, pages 19–34. Springer Berlin Heidelberg, 1997. doi:10.1007/BFb0035378.
15. S. Feo-Arenis, B. Westphal, D. Dietsch, M. Muñiz, and A.S. Andisha. The Wireless Fire Alarm System: Ensuring Conformance to Industrial Standards through Formal

Verification. In *FM 2014: Formal Methods*, volume 8442 of *LNCS*, pages 658–672. Springer International Publishing", 2014. doi:10.1007/978-3-319-06410-9_44.
16. S. Feo-Arenis, B. Westphal, D. Dietsch, M. Muñiz, and A.S. Andisha P. Andreas. The Humble Programmer. *Ready for testing: ensuring conformance to industrial standards through formal verification*, 28(3):499–527, 2016. ACM. doi:10.1007/s00165-016-0365-3.
17. S.M. German and A.P. Sistla. Reasoning about Systems with Many Processes. *Journal of the ACM*, 39(3):675–735, 1992. ACM. doi:10.1145/146637.146681.
18. 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. doi:10.1006/inco.1998.2778.
19. P. Godefroid. *Partial-Order Methods for the Verification of Concurrent Systems: An Approach to the State-Explosion Problem*, volume 1032 of *LNCS*. Springer Berlin Heidelberg, 1996.
20. 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 Berlin Heidelberg. doi:10.1007/BF01383879.
21. H.M. Hanisch. Analysis of Place/Transition Nets With Timed Arcs and Its Application to Batch Process Control. In *Application and Theory of Petri Nets*, volume 691 of *LNCS*, pages 282–299. Springer Berlin Heidelberg, 1993. doi:10.1007/3-540-56863-8_52.
22. 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. doi:10.1007/978-3-540-49382-2_26.
23. J.F. Jensen, T. Nielsen, L.K. Østergaard, 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. doi:10.1007/978-3-662-53401-4_16.
24. Peter G. Jensen. `verifydtapn` source code. https://github.com/TAPAAL/verifydtapn/tree/dual_game_pw, 2021.
25. 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. doi:10.1007/978-3-319-32582-8_9.
26. 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. doi:10.1007/s10009-017-0473-2.
27. P. Kasting, M.R. Hansen, and S. Vester. Synthesis of Railway-Signaling Plans Using Reachability Games. In *Symposium on Theory of Computing*, IFL' 2016, pages 1–13. ACM, 2016. doi:10.1145/3064899.3064908.
28. J. Lilius. Efficient State Space Search for Time Petri Nets. *Electronic Notes in Theoretical Computer Science*, 18(1):113–133, 1998. Elsevier. doi:10.1016/S1571-0661(05)80254-3.
29. M. Minea. Partial Order Reduction for Model Checking of Timed Automata. In *International Conference on Concurrency Theory*, volume 1664 of *LNCS*, pages 431–446. Springer Berlin Heidelberg, 1999. doi:10.1007/3-540-48320-9_30.
30. T. Neele, T.A.C. Willemse, and W. Wesselink. Partial-Order Reduction for Parity Games with an Application on Parameterised Boolean Equation Systems. In *TACAS*, volume 12079 of *LNCS*, pages 307–324. Springer International Publishing, 2020. doi:10.1007/978-3-030-45237-7_19.

31. 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. doi:10.1007/3-540-56922-7_34.
32. D. Peled. Combining Partial Order Reductions With On-The-Fly Model-Checking. *Formal Methods in System Design*, 8(1):39–64, 1996. Springer Berlin Heidelberg. doi:10.1007/BF00121262.
33. R.H. Sloan and U. Buy. Stubborn Sets for Real-Time Petri Nets. *Formal Methods in System Design*, 11(1):23–40, 1997. Springer. doi:10.1023/A:1008629725384.
34. A. Valmari. Stubborn Sets for Reduced State Space Generation. In *Advances in Petri Nets 1990*, volume 483 of *LNCS*, pages 491–515. Springer Berlin Heidelberg, 1991. doi:10.1007/3-540-53863-1_36.
35. A. Valmari. A Stubborn Attack on State Explosion. *Formal Methods in System Design*, 1(4):297–322, 1992. Springer Berlin Heidelberg. doi:10.1007/BF00709154.
36. 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. ACM, 1997. doi:10.5555/266557.266608.
37. A. Valmari. *Stubborn Set Intuition Explained*, volume 10470 of *LNCS*, pages 140–165. Springer Berlin Heidelberg, 2017. doi:10.1007/978-3-662-55862-1_7.
38. T. Yoneda and B.-H. Schlingloff. Efficient Verification of Parallel Real–Time Systems. *Formal Methods in System Design*, 11(2):187–215, 1997. Springer. doi:10.1023/A:1008682131325.

A Proof of Theorem 1

With Lemma 4 and 5 we prove that stable reductions preserve the existence of winning reachability strategies and hence prove Theorem 1.

Lemma 4. *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, Goal)$ be TGLTS and St a stable reduction. For all $s \in \mathcal{S}$ if state s is a winning state for player 1 in G then the state s is a winning state 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 $\sigma_1 \in \Sigma^1_G$ such that for all $\sigma_2 \in \Sigma^2_G$ and for all $\pi \in \Pi^{max}_{G,\sigma_1,\sigma_2}(s)$ there exists a position i s.t. $\pi_i \in Goal$. The goal is to construct a strategy $\sigma'_1 \in \Sigma^1_{G_{St}}$ based on σ_1 s.t. s is a winning state for player 1 in G_{St} with σ'_1. We start by first constructing σ'_1 and then proving it is a winning strategy for player 1 at s in G_{St}.

Constructing σ'_1. Assume that s is not a deadlock. There are three cases: (1) ($en_1(s) \neq \emptyset$ and $en_2(s) \neq \emptyset$) or $\neg\mathsf{zt}(s)$, (2) $en_2(s) = \emptyset$ and $\mathsf{zt}(s)$, and (3) $en_1(s) = \emptyset$ and $\mathsf{zt}(s)$.

Case (1): ($en_1(s) \neq \emptyset$ and $en_2(s) \neq \emptyset$) or $\neg\mathsf{zt}(s)$. Let $\sigma_1(s) = (d, a)$ s.t. $s \xrightarrow{d} s' \xrightarrow{a} s''$. If $a \in St(s')$ then $\sigma'_1(s) = \sigma_1(s)$. Otherwise, a was pruned at s' and $a \notin St(s')$. This implies that the state s' is zero-time $\mathsf{zt}(s')$ since otherwise by Condition **Z** we would have $en(s') \subseteq St(s')$ and $a \in St(s')$. Furthermore, due to Condition **I**, s' is a player 1 state since if $en_2(s') \neq \emptyset$ then we would again have $en(s') \subseteq St(s')$ and $a \in St(s')$. Therefore Case (2) applies at s' where some $a' \in en_1(s')$ is proposed s.t. $\sigma'_1(s') = (0, a')$ and we let $\sigma'_1(s) = (d, a')$.

Case (2): $en_2(s) = \emptyset$ and $\mathsf{zt}(s)$. Let $\pi \in \Pi^{max}_{G,\sigma_1}(s)$ be an arbitrary run. There exists a minimal position i s.t. for all $0 \leq j < i$ we have $\mathsf{zt}(\pi_j)$, π_j is a player 1 state, $\pi_j \notin Goal$, and either:

- π_i is not a player 1 state,
- $\neg\mathsf{zt}(\pi_i)$, or
- $\pi_i \in Goal$.

Let $w = a_1 a_2 \cdots a_i \in A_1^*$ be an action sequence of player 1 actions s.t.

$$s \xrightarrow{a_1} \pi_1 \xrightarrow{a_2} \cdots \xrightarrow{a_i} \pi_i$$

which is possible because all the states are zero-time and 0 delays do not change the state.

We want to show that a run to π_i is preserved in the reduced game and that σ'_1 brings us to that state. We proceed by induction on i with the induction hypothesis $IH(i)$: "If s is a player 1 and zero-time state, $s \xrightarrow{w} \pi_i$, all intermediate states are player 1 states, and $|w| = i$ then there exists $w' \in A_1^*$ s.t. $s \xrightarrow[St]{w'} \pi_i$, all intermediate states are player 1 states, and $|w'| = i$". The base case where $i = 0$ is trivial. Let $i > 0$. Assume for the sake of contradiction that $w \in \overline{St(s)}^*$. Then by the Conditions **G1**, **T**, or **R** we have that $en_2(\pi_i) = \emptyset$, $\mathsf{zt}(\pi_i)$, or $\pi_i \notin Goal$,

respectively. In all cases we have a contradiction. Therefore, there must exist a minimal position $0 \leq j \leq i$ s.t. $a_j \in St(s)$ and for all $0 \leq k < j$ we have $a_k \notin St(s)$. We can then divide w s.t. $w = va_ju$ and we have $s \xrightarrow{a_j} s_0 \xrightarrow{v} \pi_j \xrightarrow{u} \pi_i$ due to Condition **W**. Then there are two subcases: (2.1) $a_j \in \mathit{safe}(s)$ and (2.2) $a_j \notin \mathit{safe}(s)$.

- Case (2.1): $a_j \in \mathit{safe}(s)$. For all $1 \leq k < j$ we have $en_2(\pi_k) = \emptyset$ due to j being minimal and Condition **G1**. From that, if $a_j \in \mathit{safe}(s)$ then for all intermediate states in $s \xrightarrow{a_jv} \pi_j$ we only have player 1 states otherwise a_j 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 $v = a_1a_2 \cdots a_{j-1}$ s.t. $s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots \xrightarrow{a_{j-1}} \pi_j$ and for all $1 \leq k < j-1$ we have $en_2(s_k) = \emptyset$. Clearly, we have $|vu| = i-1$ and all intermediate states are player 1 states. By the induction hypothesis we have that there exists $w' = a'_1a'_2 \cdots a'_{i-1} \in A_1^*$ s.t. $s_0 \xrightarrow[St]{w'} \pi_i$, $|w'| = |vu| = i-1$, and all intermediate states are player 1 states. Let $s_0 \xrightarrow[St]{a'_1} s'_1 \xrightarrow[St]{a'_2} \cdots \xrightarrow[St]{a'_{i-1}} \pi_i$. Since $a_j \in St(s)$ we have $s \xrightarrow[St]{a_jw'} \pi_i$. Let σ'_1 be defined s.t. $\sigma'_1(s) = (0, a_j)$, $\sigma'_1(s_0) = (0, a'_1)$, and for all $1 < k < i-1$ we have $\sigma'_1(s'_k) = (0, a'_{k+1})$.
- Case (2.2): $a_j \notin \mathit{safe}(s)$. In this case we have $en(s) \subseteq St(s)$ due to Condition **S**. Therefore, for $\sigma_1(s) = (0, a_1)$, we also have $a_1 \in en(s) \cap St(s)$ and $s \xrightarrow{a_1} \pi_1$. Clearly, we have $|a_2 \cdots a_i| = i-1$ and all intermediate states are player 1 states. By the induction hypothesis we have that there exists $w' = a'_1a'_2 \cdots a'_{i-1} \in A_1^*$ s.t. $\pi_1 \xrightarrow[St]{w'} \pi_i$, $|w'| = |a_2 \cdots a_i| = i-1$, and all intermediate states are player 1 states. Let $\pi_1 \xrightarrow[St]{a'_1} s'_1 \xrightarrow[St]{a'_2} \cdots \xrightarrow[St]{a'_{i-1}} \pi_i$. Since $a_1 \in St(s)$ we have $s \xrightarrow[St]{a_1w'} \pi_i$. Let σ'_1 be defined s.t. $\sigma'_1(s) = (0, a_1)$, $\sigma'_1(\pi_1) = (0, a'_1)$, and for all $1 < k < i-1$ we have $\sigma'_1(s'_k) = (0, a'_{k+1})$.

Note that any run from s using σ'_1 or σ_1 will end up in the state π_i irrelevant of the opponents strategy as player 2 never has actions enabled before then.

Case (3): $en_1(s) = \emptyset$ and $\mathsf{zt}(s)$. We have $\sigma'_1(s) = \sigma_1(s) = \lambda$ due to the definition of strategies.

From π_i and onwards the constructions continues as described above.

Showing σ'_1 is a winning strategy. Next we show that σ'_1 is a winning strategy at s in G_{St}. In the case that s is a deadlock or $s \in \mathit{Goal}$ then σ'_1 is trivially a winning strategy. Assume for the sake of contradiction that $\sigma'_2 \in \Sigma^2_{G_{St}}$ is a winning player 2 strategy where there exists $\pi \in \Pi^{max}_{G_{St},\sigma'_1,\sigma'_2}(s)$ s.t.

$$s = \pi_0 \xrightarrow[St]{d_1a_1} \pi_1 \xrightarrow[St]{d_2a_2} \pi_2 \xrightarrow[St]{d_3a_3} \cdots$$

and for all positions i we have $\pi_i \notin \mathit{Goal}$.

We want to find a strategy $\sigma_2 \in \Sigma_G^2$ s.t. s is a winning state for player 2 at s in G. Let σ_2 be defined identically to σ_2', i.e let $\sigma_2(s) = \sigma_2'(s)$ for all $s \in \mathcal{S}$. This is a well defined strategy since $\underset{St}{\longrightarrow} \subseteq \rightarrow$ due to the definition of a reduced game.

We show that it is possible to construct a run $\pi' \in \Pi^{max}_{G,\sigma_1,\sigma_2}(s)$ in the original game subject to σ_1 and σ_2 s.t. π' does not visit any goal states. Assume that it is possible to construct π' up to the state π_i' for a position i where $\pi_i' = \pi_i$. Next, we show that we can extend π' from π_i' s.t. there exists a position $j > i$ s.t. $\pi_j' = \pi_j$ where for all $i < k \leq j$ we have $\pi_k' \notin \mathit{Goal}$. We have the same cases used in the construction of σ_1'.

Case (1): If $a_{i+1} \in A_2$ then we have $\sigma_2(\pi_i') = \sigma_2'(\pi_i') = (d_{i+1}, a_{i+1})$ and $\pi_i' \xrightarrow{d_{i+1}a_{i+1}} \pi_{i+1}$. If $a_{i+1} \in A_1$ then let $\pi_i' \xrightarrow{d_{i+1}} s'$. In the case that s' is mixed or not zero-time then we have $\sigma_1(\pi_i') = \sigma_1'(\pi_i') = (d_{i+1}, a_{i+1})$ and $\pi_i' \xrightarrow{d_{i+1}a_{i+1}} \pi_{i+1}$. Clearly, in either of these two cases we have $\pi_{i+1} \notin \mathit{Goal}$ and $i{+}1 > i$. Otherwise, s' is a zero-time and player 1 state and σ_1 and σ_1' may not propose the same action at s'. If this is the case then Case (2) applies at s' where we by the construction of σ_1' have shown that both σ_1 and σ_1' end up in a state π_j for a minimal position $j > i$. For all $i < k \leq j$ we have $en_2(\pi_k') = \emptyset$ so π_j is reached unobstructed by player 2. Furthermore, we have $\pi_k' \notin \mathit{Goal}$ and $\pi_j \notin \mathit{Goal}$.

Case (2): By the construction of σ_1' we have shown that both σ_1 and σ_1' end up in a state π_j for a minimal position $j > i$. For all $i \leq k < j$ we have $en_2(\pi_k') = \emptyset$ so π_j is reached unobstructed by player 2. Furthermore, we have $\pi_k' \notin \mathit{Goal}$ and $\pi_j \notin \mathit{Goal}$.

Case (3): In this case we have $\sigma_2(\pi_i') = \sigma_2'(\pi_i') = (d_{i+1}, a_{i+1})$ and $\pi_i' \xrightarrow{d_{i+1}a_{i+1}} \pi_{i+1}$. Clearly, we have $\pi_{i+1} \notin \mathit{Goal}$ and $i + 1 > i$.

In every case we have shown it is possible to extend the run π' from π_i' to π_j' s.t. $j > i$ and without visiting a goal state. Inductively, we have that π' exists. We can therefore conclude that σ_1 is not a winning strategy for player 1 at s in G. However, this contradicts that s is a winning state for player 1 in G. Hence π does not exist and σ_1' must be a winning strategy for player 1 at s in G_{St}. □

Lemma 5. *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, \mathit{Goal})$ be TGLTS and St a stable reduction. For all $s \in \mathcal{S}$ if state s is a winning state for player 1 in G_{St} then state s is a winning state for player 1 in G.*

Proof. We prove this by contraposition.

Assume that $s \in \mathcal{S}$ is a winning state for player 2 in G. By definition we have that there exists a player 2 strategy $\sigma_2 \in \Sigma_G^2$ such that for all $\sigma_1 \in \Sigma_G^1$, there exists $\pi \in \Pi^{max}_{G,\sigma_1,\sigma_2}(s)$ s.t. for all positions i we have $\pi_i \notin \mathit{Goal}$. The goal is to construct a strategy $\sigma_2' \in \Sigma^2_{G_{St}}$ based on σ_2 s.t. s is a winning state for player 2 in G_{St} with σ_2'. We start by first constructing σ_2' and then proving it is a winning strategy for player 2 at s in G_{St}.

Constructing σ_2'. Assume that s is not a deadlock. There are three cases: (1) ($en_1(s) \neq \emptyset$ and $en_2(s) \neq \emptyset$) or $\neg\mathsf{zt}(s)$, (2) $en_2(s) = \emptyset$ and $\mathsf{zt}(s)$, and (3) $en_1(s) = \emptyset$ and $\mathsf{zt}(s)$.

Case (1): ($en_1(s) \neq \emptyset$ and $en_2(s) \neq \emptyset$) or $\neg\mathsf{zt}(s)$. Let $\sigma_2(s) = (d, a)$ s.t. $s \xrightarrow{d} s' \xrightarrow{a} s''$. If $a \in St(s')$ then $\sigma_2'(s) = \sigma_2(s)$. Otherwise, a was pruned at s' and $a \notin St(s')$. This implies that the state s' is zero-time $\mathsf{zt}(s')$ since otherwise by Condition **Z** we would have $en(s') \subseteq St(s')$ and $a \in St(s')$. Furthermore, due to Condition **I**, s' is a player 2 state since if $en_1(s') \neq \emptyset$ then we would again have $en(s') \subseteq St(s')$ and $a \in St(s')$. Therefore Case (3) applies where some $a' \in en_2(s')$ is proposed s.t. $\sigma_2'(s') = (0, a')$ and we let $\sigma_2'(s) = (d, a')$.

Case (2): $en_2(s) = \emptyset$ and $\mathsf{zt}(s)$. We have $\sigma_2'(s) = \sigma_2(s) = \lambda$ due to the definition of strategies.

Case (3): $en_1(s) = \emptyset$ and $\mathsf{zt}(s)$. If there exists $w \in A_2^*$ s.t. $s \xrightarrow{w} s'$ and $s' \in \mathit{Goal}$ then due to Condition **V** we have $en_2(s) \subseteq St(s)$ and $\sigma_2'(s) = \sigma_2(s)$. Otherwise, let $\pi \in \mathit{\Pi}^{max}_{G,\sigma_2}(s)$ be an arbitrary run where for all positions i we have $\pi_i \notin \mathit{Goal}$. Let $a_1a_2a_3\cdots \in A^* \cup A^\omega$ s.t.

$$s \xrightarrow{d_1a_1} \pi_1 \xrightarrow{d_2a_2} \pi_2 \xrightarrow{d_3a_3} \cdots$$

where $d_i \in \mathbb{R}^{\geq 0}$ for all positions i where $i > 0$. There exists a minimal position i s.t. for all $0 \leq j < i$ we have $\mathsf{zt}(\pi_j)$, π_j is a player 2 state, $\pi_j \notin \mathit{Goal}$, and either:

- π_i is not a player 2 state,
- $\neg\mathsf{zt}(\pi_i)$,
- $en(\pi_i) = \emptyset$, or
- $\ell(\pi) = \infty$ and $a_1a_2\cdots \in A_2^\omega$.

In any case, for all $0 < j \leq i$ we have $d_j = 0$ and $s \xrightarrow{a_1a_2\cdots a_i} \pi_i$ due to 0 delays not changing the state. Let $w = a_1a_2\cdots a_i$. We will handle the fourth case separately.

We want to show that a run to π_i is preserved in the reduced game and that σ_2' brings us to that state. We proceed by induction on i with the induction hypothesis $IH(i)$: "If s is a player 2 and zero-time state, $s \xrightarrow{w} \pi_i$, and $|w| = i$ then there exists $w' \in A_2^*$ s.t. $s \xrightarrow[St]{w'} \pi_i$ and $|w'| = i$".

The base case where $i = 0$ is trivial. Let $i > 0$. Assume for the sake of contradiction that $w \in \overline{St(s)}^*$. Then by the Conditions **G1**, **T**, and **D** we have that $en_1(\pi_i) = \emptyset$, $\mathsf{zt}(\pi_i)$, and $en(\pi_i) \neq \emptyset$, respectively for the first three cases.

In all cases we have a contradiction. Therefore, there must exist a minimal position $0 \leq j < i$ s.t. $a_j \in St(s)$ and for all $0 \leq k < j$ we have $a_k \notin St(s)$. We can then divide w s.t. $w = va_ju$ and we have $s \xrightarrow{a_j} s_0 \xrightarrow{v} \pi_j \xrightarrow{u} \pi_i$ due to Condition **W**. Clearly, we have $|vu| = i-1$. By the induction hypothesis we have that there exists $w' = a_1'a_2'\cdots a_{i-1}' \in A_2^*$ s.t. $s_0 \xrightarrow[St]{w'} \pi_i$ and $|w'| = |vu| = i-1$. Let $s_0 \xrightarrow{a_1'} s_1' \xrightarrow{a_2'} \cdots \xrightarrow{a_{i-1}'} \pi_i$. Since $a_j \in St(s)$ we have $s \xrightarrow[St]{a_jw'} \pi_i$. Let σ_2' be defined s.t. $\sigma_2'(s) = (0, a_j)$, $\sigma_2'(s_0) = (0, a_1')$, and for all $1 < k < i-1$ we have $\sigma_2'(s_k') = (0, a_{k+1}')$. Note that there will always exist a run from s using σ_2' or σ_2 that ends up in the state π_i irrelevant of the opponents strategy.

Lastly, we have the case there π is an infinite run. In this case, due to Theorem 8 presented in [37] and Condition **D**, we have that there exists $w'' = a_1''a_2'' \in A_2^\omega$ s.t. $s \xrightarrow[St]{a_1''} s_1'' \xrightarrow[St]{a_2''} \cdots$. Let σ_2' be defined s.t. $\sigma_2'(s) = (0, a_1'')$, and for all $1 \leq k$ we have $\sigma_2'(s_k'') = (0, a_{k+1}'')$.

From π_i and onwards the constructions continues as described above.

Showing σ_2' is a winning strategy. Next we show that σ_2' is a winning strategy for player 2 at s in G_{St}. In the case that s is a deadlock then σ_2' is trivially a winning strategy. Assume for the sake of contradiction that $\sigma_1' \in \Sigma^1_{G_{St}}$ is a winning strategy for player 1 s.t. for all $\pi \in \Pi^{max}_{G_{St},\sigma_1',\sigma_2'}(s)$ we have

$$s = \pi_0 \xrightarrow{d_1a_1} \pi_1 \xrightarrow{d_2a_2} \pi_2 \xrightarrow{d_3a_3} \cdots$$

and there exists a position i s.t. $\pi_i \in \mathit{Goal}$. Note that if π_0 is a winning state for player 1 in G then π_j where $0 \leq j \leq i$ is also a winning state for player 1 in G due to the definition of a winning strategy.

We want to find a strategy $\sigma_1 \in \Sigma^1_G$ s.t. s is a winning state for player 1 at s in G. Let σ_1 be defined identically to σ_1', i.e let $\sigma_1(s) = \sigma_1'(s)$ for all $s \in \mathcal{S}$. This is a well defined strategy since $\xrightarrow[St]{} \subseteq \rightarrow$ due to the definition of a reduced game.

We show that it is possible to construct a run $\pi' \in \Pi^{max}_{G,\sigma_1,\sigma_2}(s)$ in the original game subject to σ_1 and σ_2 s.t. π' eventually visits a goal state $\pi_p' \in \mathit{Goal}$ for a position p. Assume that it is possible to construct π' up to the state π_i' for a position i where $i < p$ and $\pi_i' = \pi_i$. Next, we show that we can extend π' from π_i' s.t. there exists a position $j > i$ s.t. $\pi_j' = \pi_j$ where π_j' is a winning state for player 1 in G and for all $i < k < j$ we have $\pi_k' \notin \mathit{Goal}$. We have the same cases used in the construction of σ_2'.

Case (1): If $a_{i+1} \in A_1$ then we have $\sigma_1(\pi_i') = \sigma_1'(\pi_i') = (d_{i+1}, a_{i+1})$ and $\pi_i' \xrightarrow{d_{i+1}a_{i+1}} \pi_{i+1}$. If $a_{i+1} \in A_2$ then let $\pi_i' \xrightarrow{d_{i+1}} s'$. In the case that s' is mixed or not zero-time then we have $\sigma_2(\pi_i') = \sigma_2'(\pi_i') = (d_{i+1}, a_{i+1})$ and $\pi_i' \xrightarrow{d_{i+1}a_{i+1}} \pi_{i+1}$. Clearly, in either of these two cases we have $i + 1 > i$ and π_j is a winning state for player 1 in G. Otherwise, s' is a zero-time and player 2 state and σ_2 and σ_2' may not propose the same action at s'. If this is the case then Case (3) applies at s' where we by the construction of σ_2 have shown that both σ_2 and σ_2' end up in a state π_j for a minimal position $j > i$. For all $i < k < j$ we have $\pi_k' \notin \mathit{Goal}$. Therefore, j is the earliest position a goal state may occur in π and π_j is a winning state for player 1 in G.

Case (2): In this case we have $\sigma_1(\pi_i') = \sigma_1'(\pi_i') = (d_{i+1}, a_{i+1})$ and $\pi_i' \xrightarrow{d_{i+1}a_{i+1}} \pi_{i+1}$. Clearly, we have $i + 1 > i$ and π_j is a winning state for player 1 in G.

Case (3): By the construction of σ_2 we have shown that both σ_2 and σ_2' end up in a state π_j for a minimal position $j > i$. For all $i < k < j$ we have $\pi_k' \notin \mathit{Goal}$. Therefore, j is the earliest position a goal state may occur in π and π_j is a winning state for player 1 in G.

In every case we have shown it is possible to extend the run π' from π_i' to π_j' s.t. $j > i$ and π_j is a winning state for player 1 in G. Inductively, we have that π' exists. We can therefore conclude that σ_2 is not a winning strategy for player 2

at s in G. However, this contradicts that s is a winning state for player 2 in G. Hence π does not exist and σ_2' must be a winning strategy for player 2 at s in G_{St}. □

B Proof of Theorem 2

The proof is split into two lemmas for each direction.

Lemma 6. *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, Goal)$ be TGLTS and St a stable reduction with Conditions $\boldsymbol{V}$ and $\boldsymbol{D}$ replaced by Conditions $\boldsymbol{V'}$ and $\boldsymbol{D'}$. For all $s \in \mathcal{S}$ if state s is a winning state for player 1 for a safety objective in G then the state s is a winning state for player 1 for a safety objective in G_{St}.*

Proof. Assume that $s \in \mathcal{S}$ is a winning state for player 1 and for a safety objective in G. By definition we have that there exists a player 1 strategy $\sigma_1 \in \Sigma_G^1$ such that for all $\sigma_2 \in \Sigma_G^2$, for all $\pi \in \mathit{\Pi}^{max}_{G,\sigma_1,\sigma_2}(s)$, and for all positions i we have $\pi_i \notin Goal$. The goal is to construct a strategy $\sigma_1' \in \Sigma^2_{G_{St}}$ based on σ_1 s.t. s is a winning state for player 1 and for a safety objective in G_{St} with σ_1'. We start by first constructing σ_1' and then proving it is a winning safety strategy for player 1 at s in G_{St}.

Constructing σ_1'. Assume that s is not a deadlock. There are three cases: (1) ($en_1(s) \neq \emptyset$ and $en_2(s) \neq \emptyset$) or $\neg\mathsf{zt}(s)$, (2) $en_2(s) = \emptyset$ and $\mathsf{zt}(s)$, and (3) $en_1(s) = \emptyset$ and $\mathsf{zt}(s)$.

Case (1): ($en_1(s) \neq \emptyset$ and $en_2(s) \neq \emptyset$) or $\neg\mathsf{zt}(s)$. Let $\sigma_1(s) = (d, a)$ s.t. $s \xrightarrow{d} s' \xrightarrow{a} s''$. If $a \in St(s')$ then $\sigma_1'(s) = \sigma_2(s)$. Otherwise, a was pruned at s' and $a \notin St(s')$. This implies that the state s' is zero-time $\mathsf{zt}(s')$ since otherwise by Condition **Z** we would have $en(s') \subseteq St(s')$ and $a \in St(s')$. Furthermore, due to Condition **I**, s' is a player 1 state since if $en_2(s') \neq \emptyset$ then we would again have $en(s') \subseteq St(s')$ and $a \in St(s')$. Therefore Case (2) applies where some $a' \in en_1(s')$ is proposed s.t. $\sigma_1'(s') = (0, a')$ and we let $\sigma_1'(s) = (d, a')$.

Case (2): $en_2(s) = \emptyset$ and $\mathsf{zt}(s)$. If there exists $w \in A_1^*$ s.t. $s \xrightarrow{w} s'$ and $s' \in Goal$ then due to Condition $\mathbf{V'}$ we have $en_1(s) \subseteq St(s)$ and $\sigma_1'(s) = \sigma_2(s)$. Otherwise, let $\pi \in \mathit{\Pi}^{max}_{G,\sigma_1}(s)$ be an arbitrary run where for all positions i we have $\pi_i \notin Goal$. Let $a_1a_2a_3\cdots \in A^* \cup A^\omega$ s.t.

$$s \xrightarrow{d_1a_1} \pi_1 \xrightarrow{d_2a_2} \pi_2 \xrightarrow{d_3a_3} \cdots$$

where $d_i \in \mathbb{R}^{\geq 0}$ for all positions i where $i > 0$. There exists a minimal position i s.t. for all $0 \leq j < i$ we have $\mathsf{zt}(\pi_j)$, π_j is a player 1 state, $\pi_j \notin Goal$, and either:

- π_i is not a player 1 state,
- $\neg\mathsf{zt}(\pi_i)$,
- $en(\pi_i) = \emptyset$, or
- $\ell(\pi) = \infty$ and $a_1a_2\cdots \in A_1^\omega$.

In any case, for all $0 < j \leq i$ we have $d_j = 0$ and $s \xrightarrow{a_1 a_2 \cdots a_i} \pi_i$ due to 0 delays not changing the state. Let $w = a_1 a_2 \cdots a_i$. We will handle the fourth case separately.

We want to show that a run to π_i is preserved in the reduced game and that σ_1' brings us to that state. We proceed by induction on i with the induction hypothesis $IH(i)$: "If s is a player 1 and zero-time state, $s \xrightarrow{w} \pi_i$, all intermediate states are player 1 states, and $|w| = i$ then there exists $w' \in A_1^*$ s.t. $s \xrightarrow[St]{w'} \pi_i$, all intermediate states are player 1, and $|w'| = i$".

The base case where $i = 0$ is trivial. Let $i > 0$. Assume for the sake of contradiction that $w \in \overline{St(s)}^*$. Then by the Conditions **G2**, **T**, and **D′** we have that $en_2(\pi_i) = \emptyset$, $\mathtt{zt}(\pi_i)$, and $en(\pi_i) \neq \emptyset$, respectively for the first three cases.

In all cases we have a contradiction. Therefore, there must exist a minimal position $0 \leq j < i$ s.t. $a_j \in St(s)$ and for all $0 \leq k < j$ we have $a_k \notin St(s)$. We can then divide w s.t. $w = va_ju$ and we have $s \xrightarrow{a_j} s_0 \xrightarrow{v} \pi_j \xrightarrow{u} \pi_i$ due to Condition **W**. Clearly, we have $|vu| = i - 1$. If $a_j \in \mathit{safe}(s)$ or $a_j \notin \mathit{safe}(s)$ then we proceed in a similar manner to Case 2 in Lemma 4 where if $a_j \notin \mathit{safe}(s)$ then σ_1' simply replicates σ_1 until π_i is reached. If $a_j \in \mathit{safe}(s)$, we proceed with the induction. By the induction hypothesis we have that there exists $w' = a_1' a_2' \cdots a_{i-1}' \in A_1^*$ s.t. $s_0 \xrightarrow[St]{w'} \pi_i$, all intermediate states are player 1 states, and $|w'| = |vu| = i - 1$. Let $s_0 \xrightarrow{a_1'} s_1' \xrightarrow{a_2'} \cdots \xrightarrow{a_{i-1}'} \pi_i$. Since $a_j \in St(s)$ we have $s \xrightarrow[St]{a_j w'} \pi_i$. Let σ_1' be defined s.t. $\sigma_1'(s) = (0, a_j)$, $\sigma_1'(s_0) = (0, a_1')$, and for all $1 < k < i - 1$ we have $\sigma_1'(s_k') = (0, a_{k+1}')$. Note that any run from s using σ_1' or σ_1 will end up in the state π_i irrelevant of the opponents strategy as player 2 never has actions enabled before then.

Lastly, we have the case there π is an infinite run. In this case, due to Theorem 8 presented in [37] and Condition **D′**, we have that there exists $w'' = a_1'' a_2'' \cdots \in A_1^\omega$ s.t. $s \xrightarrow[St]{a_1''} s_1'' \xrightarrow[St]{a_2''} \cdots$. Let σ_2' be defined s.t. $\sigma_2'(s) = (0, a_1'')$, and for all $1 \leq k$ we have $\sigma_2'(s_k'') = (0, a_{k+1}'')$.

Case (3): $en_1(s) = \emptyset$ and $\mathtt{zt}(s)$. We have $\sigma_1'(s) = \sigma_1(s) = \lambda$ due to the definition of strategies.

From π_i and onwards the constructions continues as described above.

Showing σ_1' is a winning safety strategy. Next we show that σ_1' is a winning safety strategy for player 1 at s in G_{St}. In the case that s is a deadlock then σ_1' is trivially a winning strategy. Assume for the sake of contradiction that $\sigma_2' \in \Sigma^1_{G_{St}}$ is a winning safety strategy for player 2 where there exists $\pi \in \Pi^{max}_{G_{St}, \sigma_1', \sigma_2'}(s)$ s.t.

$$s = \pi_0 \xrightarrow{d_1 a_1} \pi_1 \xrightarrow{d_2 a_2} \pi_2 \xrightarrow{d_3 a_3} \cdots$$

and there exists a position i s.t. $\pi_i \in \mathit{Goal}$. Note that if π_0 is a winning state for player 2 and for a safety objective in G then π_j where $0 \leq j \leq i$ is also a

winning state for player 2 and for a safety objective in G due to the definition of a winning safety strategy.

We want to find a strategy $\sigma_2 \in \Sigma_G^2$ s.t. s is a winning state for player 2 and for a safety objective at s in G. Let σ_2 be defined identically to σ_2', i.e let $\sigma_2(s) = \sigma_2'(s)$ for all $s \in \mathcal{S}$. This is a well defined strategy since $\xrightarrow[St]{} \subseteq \rightarrow$ due to the definition of a reduced game.

We show that it is possible to construct a run $\pi' \in \Pi^{max}_{G,\sigma_1,\sigma_2}(s)$ in the original game subject to σ_1 and σ_2 s.t. π' eventually visits a goal state $\pi'_p \in \mathit{Goal}$ for a position p. Assume that it is possible to construct π' up to the state π'_i for a position i where $i < p$ and $\pi'_i = \pi_i$. Next, we show that we can extend π' from π'_i s.t. there exists a position $j > i$ s.t. $\pi'_j = \pi_j$ where π'_j is a winning state for player 2 and for a safety objective in G and for all $i < k < j$ we have $\pi'_k \notin \mathit{Goal}$. We have the same cases used in the construction of σ_2'.

Case (1): If $a_{i+1} \in A_2$ then we have $\sigma_2(\pi'_i) = \sigma_2'(\pi'_i) = (d_{i+1}, a_{i+1})$ and $\pi'_i \xrightarrow{d_{i+1}a_{i+1}} \pi_{i+1}$. If $a_{i+1} \in A_1$ then let $\pi'_i \xrightarrow{d_{i+1}} s'$. In the case that s' is mixed or not zero-time then we have $\sigma_1(\pi'_i) = \sigma_1'(\pi'_i) = (d_{i+1}, a_{i+1})$ and $\pi'_i \xrightarrow{d_{i+1}a_{i+1}} \pi_{i+1}$. Clearly, in either of these two cases we have $i + 1 > i$ and π_j is a winning state for player 2 and for a safety objective in G. Otherwise, s' is a zero-time and player 1 state and σ_1 and σ_1' may not propose the same action at s'. If this is the case then Case (3) applies at s' where we by the construction of σ_1 have shown that both σ_1 and σ_1' end up in a state π_j for a minimal position $j > i$. For all $i < k < j$ we have $\pi'_k \notin \mathit{Goal}$. Therefore, j is the earliest position a goal state may occur in π and π_j is a winning state for player 2 and for a safety objective in G.

Case (2): By the construction of σ_1 we have shown that both σ_1 and σ_1' end up in a state π_j for a minimal position $j > i$. For all $i < k < j$ we have $\pi'_k \notin \mathit{Goal}$. Therefore, j is the earliest position a goal state may occur in π and π_j is a winning state for player 2 and for a safety objective in G.

Case (3): In this case we have $\sigma_2(\pi'_i) = \sigma_2'(\pi'_i) = (d_{i+1}, a_{i+1})$ and $\pi'_i \xrightarrow{d_{i+1}a_{i+1}} \pi_{i+1}$. Clearly, we have $i + 1 > i$ and π_j is a winning state for player 2 and for a safety objective in G.

In every case we have shown it is possible to extend the run π' from π'_i to π'_j s.t. $j > i$ and π_j is a winning state for player 2 and for a safety objective in G. Inductively, we have that π' exists. We can therefore conclude that σ_1 is not a winning safety strategy for player 1 at s in G. However, this contradicts that s is a winning state for player 1 and for a safety objective in G. Hence π does not exist and σ_1' must be a winning safety strategy for player 1 at s in G_{St}. □

Lemma 7. *Let $G = (\mathcal{S}, A_1, A_2, \rightarrow, \mathit{Goal})$ be TGLTS and St a stable reduction with Conditions **V** and **D** replaced by Conditions **V′** and **D′**. For all $s \in \mathcal{S}$ if state s is a winning state for player 1 for a safety objective in G_{St} then the state s is a winning state for player 1 for a safety objective in G.*

Proof. We prove this by contraposition.

Assume that $s \in \mathcal{S}$ is a winning state for player 2 and for a safety objective in G. By definition we have that there exists a player 2 strategy $\sigma_2 \in \Sigma_G^2$ such that for all $\sigma_1 \in \Sigma_G^1$ there exists $\pi \in \Pi^{max}_{G,\sigma_1,\sigma_2}(s)$ and exists a position i s.t. $\pi_i \in \mathit{Goal}$. The goal is to construct a strategy $\sigma_2' \in \Sigma^2_{G_{St}}$ based on σ_2 s.t. s is a winning state for player 2 and for a safety objective in G_{St} with σ_2'. We start by first constructing σ_2' and then proving it is a winning strategy for player 2 at s in G_{St}.

Constructing σ_2'. Assume that s is not a deadlock. There are three cases: (1) ($en_1(s) \neq \emptyset$ and $en_2(s) \neq \emptyset$) or $\neg\mathsf{zt}(s)$, (2) $en_2(s) = \emptyset$ and $\mathsf{zt}(s)$, and (3) $en_1(s) = \emptyset$ and $\mathsf{zt}(s)$.

Case (1): ($en_1(s) \neq \emptyset$ and $en_2(s) \neq \emptyset$) or $\neg\mathsf{zt}(s)$. Let $\sigma_2(s) = (d, a)$ s.t. $s \xrightarrow{d} s' \xrightarrow{a} s''$. If $a \in St(s')$ then $\sigma_2'(s) = \sigma_2(s)$. Otherwise, a was pruned at s' and $a \notin St(s')$. This implies that the state s' is zero-time $\mathsf{zt}(s')$ since otherwise by Condition **Z** we would have $en(s') \subseteq St(s')$ and $a \in St(s')$. Furthermore, due to Condition **I**, s' is a player 1 state since if $en_2(s') \neq \emptyset$ then we would again have $en(s') \subseteq St(s')$ and $a \in St(s')$. Therefore Case (3) applies at s' where some $a' \in en_2(s')$ is proposed s.t. $\sigma_2'(s') = (0, a')$ and we let $\sigma_2'(s) = (d, a')$.

Case (2): $en_2(s) = \emptyset$ and $\mathsf{zt}(s)$. We have $\sigma_2'(s) = \sigma_2(s) = \lambda$ due to the definition of strategies.

Case (3): $en_2(1) = \emptyset$ and $\mathsf{zt}(s)$. Let $\pi \in \Pi^{max}_{G,\sigma_2}(s)$ be an arbitrary run. There exists a minimal position i s.t. for all $0 \leq j < i$ we have $\mathsf{zt}(\pi_j)$, π_j is a player 2 state, $\pi_j \notin \mathit{Goal}$, and either:

- π_i is not a player 2 state,
- $\neg\mathsf{zt}(\pi_i)$, or
- $\pi_i \in \mathit{Goal}$.

Let $w = a_1 a_2 \cdots a_i \in A_2^*$ be an action sequence of player 2 actions s.t.

$$s \xrightarrow{a_1} \pi_1 \xrightarrow{a_2} \cdots \xrightarrow{a_i} \pi_i$$

which is possible because all the states are zero-time and 0 delays do not change the state.

We want to show that a run to π_i is preserved in the reduced game and that σ_2' brings us to that state. We proceed by induction on i with the induction hypothesis $IH(i)$: "If s is a player 2 and zero-time state, $s \xrightarrow{w} \pi_i$, and $|w| = i$ then there exists $w' \in A_2^*$ s.t. $s \xrightarrow[St]{w'} \pi_i$ and $|w'| = i$". The base case where $i = 0$ is trivial. Let $i > 0$. Assume for the sake of contradiction that $w \in \overline{St(s)}^*$. Then by the Conditions **G2**, **T**, or **R** we have that $en_1(\pi_i) = \emptyset$, $\mathsf{zt}(\pi_i)$, or $\pi_i \notin \mathit{Goal}$, respectively. In all cases we have a contradiction. Therefore, there must exist a minimal position $0 \leq j \leq i$ s.t. $a_j \in St(s)$ and for all $0 \leq k < j$ we have $a_k \notin St(s)$. We can then divide w s.t. $w = v a_j u$ and we have $s \xrightarrow{a_j} s_0 \xrightarrow{v} \pi_j \xrightarrow{u} \pi_i$ due to Condition **W**. For all $1 \leq k < j$ we have $en_2(\pi_k) = \emptyset$ due to j being minimal and Condition **G1**. Clearly, we have $|vu| = i - 1$. By the induction hypothesis we have that there exists $w' = a_1' a_2' \cdots a_{i-1}' \in A_2^*$ s.t. $s_0 \xrightarrow[St]{w'} \pi_i$ and

$|w'| = |vu| = i-1$. Let $s_0 \xrightarrow[St]{a'_1} s'_1 \xrightarrow[St]{a'_2} \cdots \xrightarrow[St]{a'_{i-1}} \pi_i$. Since $a_j \in St(s)$ we have $s \xrightarrow[St]{a_j w'} \pi_i$. Let σ'_2 be defined s.t. $\sigma'_2(s) = (0, a_j)$, $\sigma'_2(s_0) = (0, a'_1)$, and for all $1 < k < i-1$ we have $\sigma'_2(s'_k) = (0, a'_{k+1})$. Note that there will always exist a run from s using σ'_2 or σ_2 that ends up in the state π_i irrelevant of the opponents strategy.

From π_i and onwards the constructions continues as described above.

Showing σ'_2 is a winning safety strategy. Next we show that σ'_2 is a winning safety strategy at s in G_{St}. We want to show what for any $\sigma'_1 \in \Sigma^1_{G_{St}}$ there exists a run $\pi' \in \Pi^{max}_{G_{St},\sigma'_1,\sigma'_2}(s)$ s.t.

$$s = \pi'_0 \xrightarrow[St]{d_1 a_1} \pi'_1 \xrightarrow[St]{d_2 a_2} \pi'_2 \xrightarrow[St]{d_3 a_3} \cdots$$

and there exists a position i s.t. $\pi_i \in \mathit{Goal}$. In the case that s is a deadlock or $s \in \mathit{Goal}$ then σ'_2 is trivially a winning safety strategy.

Let $\sigma_1 \in \Sigma^1_G$ be a strategy for player 1 in the original game and let it be defined identically to σ'_1, i.e. let $\sigma_1(s) = \sigma'_1(s)$ for all $s \in \mathcal{S}$. This is a well defined strategy since $\xrightarrow[St]{} \subseteq \rightarrow$ due to the definition of a reduced game. We know that σ_2 is a winning safety strategy for player 2 at s in G, so there exists $\pi \in \Pi^{max}_{G,\sigma_1,\sigma_2}(s)$ and there exists a position p s.t. $\pi_p \in \mathit{Goal}$. We construct $\pi' \in \Pi^{max}_{G_{St},\sigma'_1,\sigma'_2}(s)$ from π to show that π' visits a goal state. Assume that it is possible to construct π' up to the state π'_i for a position i where $\pi'_i = \pi_i$. Next, we show that we can extend π' from π'_i s.t. there exists a position $j > i$ s.t. $\pi'_j = \pi_j$ where for all $i < k < j$ we have $\pi'_k \notin \mathit{Goal}$. We have the same cases used in the construction of σ'_2.

Case (1): If $a_{i+1} \in A_1$ then we have $\sigma_1(\pi'_i) = \sigma'_1(\pi'_i) = (d_{i+1}, a_{i+1})$, $\pi'_i \xrightarrow{d_{i+1} a_{i+1}} \pi_{i+1}$, and π'_i is extended to π'_{i+1}. If $a_{i+1} \in A_2$ then let $\pi'_i \xrightarrow{d_{i+1}} s'$. In the case that s' is mixed or not zero-time then we have $\sigma_2(\pi'_i) = \sigma'_2(\pi'_i) = (d_{i+1}, a_{i+1})$, $\pi'_i \xrightarrow{d_{i+1} a_{i+1}} \pi_{i+1}$, and π'_i is extended to π'_{i+1}. Clearly, we have $\pi_{i+1} = \pi'_{i+1}$ and $i+1 > i$. Otherwise, s' is a zero-time and player 2 state and σ_2 and σ'_2 may not propose the same action at s'. If this is the case then Case (2) applies at s' where we by the construction of σ'_2 have shown that both σ_2 and σ'_2 end up in a state $\pi_j = \pi'_j$ for a minimal position $j > i$. For all $i < k < j$ we have $\pi'_k \notin \mathit{Goal}$ and j is the earliest position a goal state may occur in π.

Case (2): In this case we have $\sigma_1(\pi'_i) = \sigma'_1(\pi'_i) = (d_{i+1}, a_{i+1})$, $\pi'_i \xrightarrow{d_{i+1} a_{i+1}} \pi_{i+1}$, and π'_i is extended to π'_{i+1}. Clearly, we have $\pi_{i+1} = \pi'_{i+1}$ and $i+1 > i$.

Case (3): By the construction of σ'_2 we have shown that both σ_2 and σ'_2 end up in a state $\pi_j = \pi'_j$ for a minimal position $j > i$. For all $i \leq k < j$ we have $en_1(\pi'_k) = \emptyset$ so π_j is reached. Furthermore, we have $\pi'_k \notin \mathit{Goal}$ and j is the earliest position a goal state may occur in π.

In every case we have shown it is possible to extend the run π' from π'_i to π'_j s.t. $j > i$, $\pi_j = \pi'_j$, and for all $i < k < j$ we have $\pi'_j \notin \mathit{Goal}$. Inductively, we have that π' eventually reaches a goal state since π eventually reaches a goal state. Therefore, σ'_2 must be a winning safety strategy for player 2 at s in G_{St}. □

C Proof of Theorem 3

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

- (**I**): Follows from Condition 1.
- (**Z**): Follows from Condition 2.
- (**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. $M \xrightarrow{w} M_w$. By contradiction assume that $t \notin en(M)$. Then t is disabled in M because there is $p \in {}^\bullet t$ such that $|\{x \in M(p) \mid x \in g((p,t))\}| < w((p,t))$ or there is $p \in t^\circ$ such that $|M(p)| \geq w((p,t))$. In the first case, due to Condition 8a all the transitions that can add tokens that are in the guard $g((p,t))$ to p are included in $St(M)$. Since $w \in \overline{St(M)}^*$ this implies that $|\{x \in M_w(p) \mid x \in g((p,t))\}| < w((p,t))$ and $t \notin en(M_w)$ contradicting our assumption that $M_w \xrightarrow{t} M'$. In the second case, due to Condition 8b all the transitions that can remove at least one token from p are included in $St(M)$. Since $w \in \overline{St(M)}^*$ this implies that $|M_w(p)| \geq w((p,t))$ and $t \notin en(M_w)$, again 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 a possible execution sequence. For the sake of contradition, 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 $p \in {}^\bullet t \cap {}^\bullet t'$ where $g((p,t)) \cap g((p,t')) \neq \emptyset$ or t added one or more tokens to a place $p \in t^\bullet \cap {}^\circ t'$. In the first case, due to Condition 9a all the transitions that can remove tokens that are in the guard $g((p,t))$ 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 9b we know that $(t^\bullet)^\circ \subseteq St(M)$, 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 can conlude with $M \xrightarrow{tw} M'$.
- (**R**): Follows from Condition 5 and Lemma 1.
- (**T**): 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 $\mathsf{zt}(M)$ then $\mathsf{zt}(M')$. Assume that $\mathsf{zt}(M)$. Since $\mathsf{zt}(M)$ there are two cases: $T_{urg} \cap en(M) \neq \emptyset$ or there is $p \in P$ where $I(p) = [a,b]$ and $b \in M(p)$. In the first case, there is $t \in T_{urg} \cap en(M) \cap St(M)$ and ${}^\bullet({}^\circ t) \subseteq St(M)$ due to Condition 3a. For any $p \in {}^\bullet t$ and $p' \in {}^\circ t$ we have that $|\{x \in M(p) \mid x \in g((p,t))\}| \geq w((p,t))$ and $|M(p')| < w((p',t))$. Due to Condition 9a we know for all $t' \in p^\bullet$ that $t' \in St(M)$ if $g((p,t)) \cap g((p,t')) \neq \emptyset$. Therefore we have $|\{x \in M'(p) \mid x \in g((p,t))\}| \geq w((p,t))$ since $w \in \overline{St(M)}^*$. Due to ${}^\bullet({}^\circ t)$ in Condition 3a w cannot add any tokens to p' since $w \in \overline{St(M)}^*$ and we have that $|M'(p')| < w((p',t))$. This implies $\mathsf{zt}(M')$. In the second case, there is $p \in P$ where $I(p) = [a,b]$ and $b \in M(p)$. Due to Condition 9b for

all $t \in p^\bullet$ where $b \in g((p,t))$ we have that $t \in St(M)$. Therefore t can never occur in w since $w \in \overline{St(M)}^*$ and we must have that $b \in M'(p)$, implying that $\mathsf{zt}(M')$.

- (**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 7 we have $T_2 \subseteq St(M)$. Let $t \in T_2$ be a player 2 transition. By Condition 8 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 4.
- (**V**): Follows from Condition 11 and Lemma 2. 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 $t \in en_2(M) \cup St(M)$ s.t. $t \in en_2(M')$. Assume that $en_2(M) \neq \emptyset$. From Condition 10 we know that there exists $t \in en_2(M) \cap St(M)$ s.t. $\{t' \in ({}^\bullet t)^\bullet \mid \exists p \in {}^\bullet t \cup {}^\bullet t' \wedge g((p,t')) \cap g((p,t)) \cap M(p) \neq \emptyset\} \cup {}^+({}^\circ t) \subseteq St(M)$. 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. $|\{x \in M'(p) \mid x \in g((p,t))\}| < w((p,t))$, or there exsits $p \in {}^\circ t$ s.t. $|M'(p)| \geq I((p,t))$. In the first case, since $t \in en_2(M)$ we have that $|\{x \in M(p) \mid x \in g((p,t))\}| < w((p,t))$. Therefore at least one transition $t' \in \{t' \in ({}^\bullet t)^\bullet \mid \exists p \in {}^\bullet t \cup {}^\bullet t' \wedge g((p,t')) \cap g((p,t)) \cap M(p) \neq \emptyset\}$, i.e. transitions that removes appropriately ages tokens from the preset of t, has to have been fired in w. However, due to Condition 10, we know that $t' \in St(M)$ is true, and therefore t' cannot occur in w. This implies $|\{x \in M'(p) \mid x \in g((p,t))\}| \geq w((p,t))$, 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 $t \in en_2(M')$. Condition $\mathbf{D}'$ follows the same approach as Condition **D** with all instances of T_2 switched with T_1.

This completes the proof of the theorem. □

D Proof of Theorem 4

Proof. Let $N = (P, T_1, T_2, T_{urg}, IA, OA, g, w, Type, I)$ be a TAPG, $M \in \mathcal{M}(N)$ a marking, and φ a formula.

Termination. Algorithm 1 contains no loops so it is sufficient to show that the call to *Saturate*(Y) in Line 17 (Algorithm 2) terminates. We observe that the loops at Line 10, 11, 15, and 18 in Algorithm 2 iterates over subsets of places and transitions which are finite by definition. Therefore, we now only need to show that the while-loop in Line 2 terminates.

Assume that $Y \neq \emptyset$, and we enter the body of the while-loop. A transition $t \in T$ is chosen in Line 3 and Lines 4 to 19 is executed. As shown, every loop in the body of the while-loop terminates, and we reach Line 20. We remove t from Y and add t to X in Line 20. Note that during the execution of the while-loop, a transition is added Y only if is not already in X (Lines 8, 10, 15, and 18). Therefore, we choose a unique transition in Line 3 in every iteration of the while-loop, and the number of transitions in X is strictly increasing every iteration. By definition, the set of transitions T is finite the while-loop iterates a finite number of times, and Algorithm 2 terminates.

Correctness. To show correctness we argue that Algorithm 1 and 2 implements the syntactic conditions of Theorem 3.

- Condition 1: Corresponds to the check at Line 3 in Algorithm 1.
- Condition 2: Corresponds to the check at Line 2 in Algorithm 1.
- Condition 3: Corresponds to Lines 10 to 16 in Algorithm 1.
- Condition 4: Corresponds to the check at Line 18 in Algorithm 1.
- Condition 5: Corresponds to at Line 17 in Algorithm 1.
- Condition 6: Corresponds to at Line 7 in Algorithm 1.
- Condition 7: Corresponds to at Line 9 in Algorithm 1.
- Condition 8: Corresponds to Lines 4 to 16 in Algorithm 2. Every disabled stubborn transition t is initially added to Y is eventually picked at Line 3 before it is added to X. Therefore, the code at Lines 4 to 16 are executed for t.
- Condition 9: Corresponds to Lines 18 to 19 in Algorithm 2. Every enabled stubborn transition t is initially added to Y is eventually picked at Line 3 before it is added to X. Therefore, the code at Lines 18 to 19 are executed for t.
- Condition 10: Corresponds to Line 7 in Algorithm 1.
- Condition 11: Corresponds to the check at Line 18 in Algorithm 1.

This concludes the proof. □