Start Pruning When Time Gets Urgent: Partial Order Reduction for Timed Systems

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. Partial order reduction for timed systems is a challenging topic due to the dependencies among events induced by time acting as a global synchronization mechanism. So far, there has only been a limited success in finding practically applicable solutions yielding significant state space reductions. We suggest a working and efficient method to facilitate stubborn set reduction for timed systems with urgent behaviour. We first describe the framework in the general setting of timed labelled transition systems and then instantiate it to the case of timed-arc Petri nets. The basic idea is that we can employ classical untimed partial order reduction techniques as long as urgent behaviour is enforced. Our solution is implemented in the model checker TAPAAL and the feature is now broadly available to the users of the tool. By a series of larger case studies, we document the benefits of our method and its applicability to real-world scenarios.

1 Introduction

Partial order reduction techniques for untimed systems, introduced by Godefroid, Peled, and Valmari in the nineties (see e.g. [6]), have since long proved successful in combating the notorious state space explosion problem. For *timed* systems, the success of partial order reduction has been significantly challenged by the strong dependencies between events caused by time as a global synchronizer. Only recently—and moreover in combination with *approximate* abstraction techniques—stubborn set techniques have demonstrated a true reduction potential for systems modelled by timed automata [23].

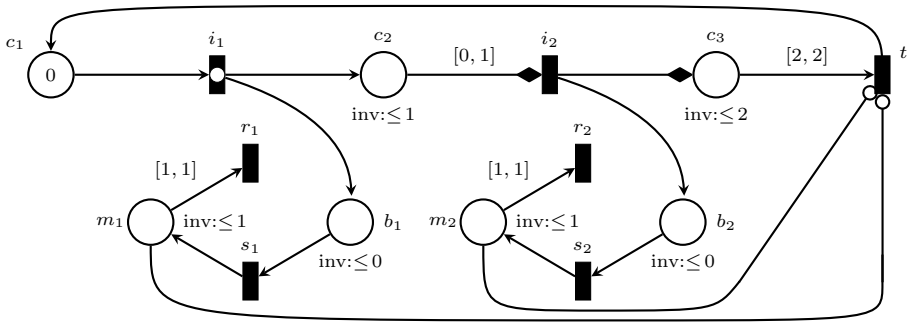
We pursue an orthogonal solution to the current partial order approaches for timed systems and, based on a stubborn set reduction [28, 39], we target a general class of timed systems with *urgent behaviour*. In a modular modelling approach for timed systems, urgency is needed to realistically model behaviour in a component that should be unobservable to other components [36]. Examples of such instantaneously evolving behaviours include, among others, cases like behaviour detection in a part of a sensor (whose duration is assumed to be negligible) or handling of release and completion of periodic tasks in a real-time operating system. We observe that focusing on the urgent part of the behaviour of a timed system allows us to exploit the full range of partial order reduction

techniques already validated for untimed systems. This leads to an exact and broadly applicable reduction technique, which we shall demonstrate on a series of industrial case studies showing significant space and time reduction. In order to highlight the generality of the approach, we first describe our reduction technique in the setting of timed labelled transition systems. We shall then instantiate it to timed-arc Petri nets and implement and experimentally validate it in the model checker TAPAAL [19].

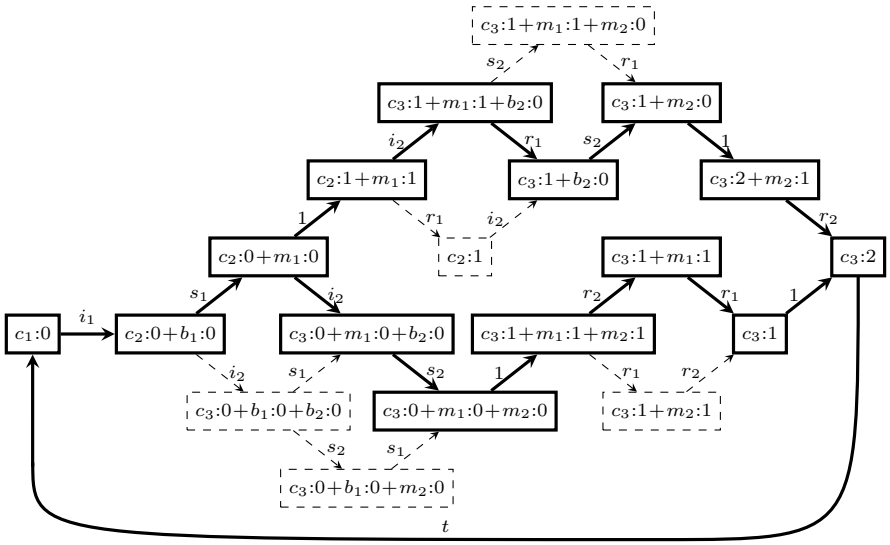
Let us now briefly introduce the model of timed-arc Peri nets and explain our reduction ideas. In timed-arc Petri nets, each token is associated with a nonnegative integer representing its age and input arcs to transitions contain intervals, restricting the ages of tokens available for transition firing (if an interval is missing, we assume the default interval $[0, \infty]$ that accepts all token ages). In Figure 1a we present a simple monitoring system modelled as a timed-arc Petri net. The system consists of two identical sensors where sensor i, $i \in \{1, 2\}$, is represented by the places b_i and m_i, and the transitions s_i and r_i. Once a token of age 0 is placed into the place b_i, the sensor gets started by executing the transition s_i and moving the token from place b_i to m_i where the monitoring process starts. As the place b_i has an associated age invariant ≤ 0, meaning that all tokens in b_i must be of age at most 0, no time delay is allowed and the firing of s_i becomes urgent. In the monitoring place m_i we have to delay one time unit before the transition r_i reporting the reading of the sensor becomes enabled. Due to the age invariant ≤ 1 in the place m_i, we cannot wait longer than one time unit, after which r_i becomes also urgent.

The places c_1, c_2 and c_3 together with the transitions i_1, i_2 and t are used to control the initialization of the sensors. At the execution start, only the transition i_1 is enabled and because it is an urgent transition (denoted by the white circle), no delay is initially possible and i_1 must be fired immediately while removing the token of age 0 from c_1 and placing a new token of age 0 into c_2. At the same time, the first sensor gets started as i_1 also places a fresh token of age 0 into b_1. Now the control part of the net can decide to fire without any delay the transition i_2 and start the second sensor, or it can delay one unit of time after which i_2 becomes urgent due to the age invariant ≤ 1 as the token in c_2 is now of age 1. If i_2 is fired now, it will place a fresh token of age 0 into b_2. However, the token that is moved from c_2 to c_3 by the pair of transport arcs with the diamond-shaped arrow tips preserves its age 1, so now we have to wait precisely one more time unit before t becomes enabled. Moreover, before t can be fired, the places m_1 and m_2 must be empty as otherwise the firing of t is disabled due to inhibitor arcs with circle-shaped arrow tips.

In Figure 1b we represent the reachable state space of the simple monitoring system where markings are represented using the notation like $c_3 : 1 + b_2 : 2$ that stands for one token of age 1 in place c_3 and one token of age 2 in place b_2. The dashed boxes represent the markings that can be avoided during the state space exploration when we apply our partial order reduction method for checking if the termination transition t can become enabled from the initial marking. We can see that the partial order reduction is applied such that it preserves at least



(a) TAPN model of a simple monitoring system



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

Fig. 1: Simple Monitoring System

one path to all configurations where our goal is reached (transition t is enabled) and where time is not urgent anymore (i.e. to the configurations that allow the delay of 1 time unit). The basic idea of our approach is to apply the stubborn set reduction on the commutative diamonds where time is not allowed to elapse.

Related Work. Our stubborn set reduction is based on the work of Valmari et al. [28, 39]. We formulate their stubborn set method in the abstract framework of labelled transition systems with time and add further axioms for time elapsing in order to guarantee preservation of the reachability properties.

For Petri nets, Yoneda and Schlingloff [41] apply a partial order reduction to one-safe time Petri nets, however, as claimed in [38], the method is mainly

suitable for small to medium models due to a computational overhead, confirmed also in [29]. The experimental evaluation in [41] shows only one selected example. Sloan and Buy [38] try to improve on the efficiency of the method, at the expense of considering only a rather limited model of *simple time Petri nets* where each transition has a statically assigned duration. Lilius [29] suggests to instead use alternative semantics of timed Petri nets to remove the issues related to the global nature of time, allowing him to apply directly the untimed partial order approaches. However, the semantics is nonstandard and no experiments are reported. Another approach is by Virbitskaite and Pokozy [40], who apply a partial order method on the *region graph* of bounded time Petri nets. Region graphs are in general not an efficient method for state space representation and the method is demonstrated only on a small buffer example with no further experimental validation. Recently, partial order techniques were suggested by André, Chatain and Rodríguez for parametric time Petri nets [5], however, the approach is working only for safe and acyclic nets. Boucheneb and Barkaoui [14, 13, 12] discuss a partial order reduction technique for timed Petri nets based on *contracted state class graphs* and present a few examples on a prototype implementation (the authors do not refer to any publicly available tool). Their method is different from ours as it aims at adding timing constrains to the independence relation, but it does not exploit urgent behaviour. Moreover, the models of time Petri nets and timed-arc Petri nets are, even on the simplest nets, incomparable due to the different way to modelling time.

The fact that we are still lacking a practically applicable method for the time Petri net model is documented by a missing implementation of the technique in leading tools for time Petri net model checking like TINA [9] and Romeo [22]. We are not aware of any work on partial order reduction technique for the class of timed-arc Petri nets that we consider in this paper. This is likely because this class of nets provides even more complex timing behaviour, as we consider unbounded nets where each token carries its timing information (and needs a separate clock to remember the timing), while in time Petri nets timing is associated only to a priory fixed number of transitions in the net.

In the setting of timed automata [3], early work on partial order reduction includes Bengtsson et al. [8] and Minea [32] where they introduce the notion of local as well as global clocks but provide no experimental evaluation. Dams et al. [18] introduce the notion of *covering* in order to generalize dependencies but also here no empirical evaluation is provided. Lugiez, Niebert et al. [30, 34] study the notion of *event zones* (capturing time-durations between events) and use it to implement Mazurkiewicz-trace reductions. Salah, Bozga and Maler [37] introduce and implement an exact method based on merging zones resulting from different interleavings. The method achieves performance comparable with the approximate convex-hull abstraction which is by now superseded by the exact LU-abstraction [7]. Most recently, Hansen et al. [23] introduce a variant of stubborn sets for reducing an *abstracted zone graph*, thus in general offering overapproximate analysis. Our technique is orthogonal to the other approaches mentioned above; not only is the model different but also the application of our

reduction gives exact results and is based on new reduction ideas. Finally, the idea of applying partial order reduction for independent events that happen at the same time appeared also in [15] where the authors, however, use a static method that declares actions as independent only if they do not communicate, do not emit signals and do not access any shared variables. Our realization of the method to the case of timed-arc Petri nets applies a dynamic (on-the-fly) reduction, while executing a detailed timing analysis that allows us to declare more transitions as independent—sometimes even in the case when they share resources.

2 Partial Order Reduction for Timed Systems

We shall now describe the general idea of our partial order reduction technique (based on stubborn sets [28, 39]) in terms of timed transition systems. We consider real-time delays in the rest of this section, as these results are not specific only to discrete time semantics. Let A be a given set of actions such that $A \cap \mathbb{R}_{\geq 0} = \emptyset$ where $\mathbb{R}_{\geq 0}$ stands for the set of nonnegative real numbers.

Definition 1 (Timed Transition System). *A* timed transition system *is a tuple $(S, s_0, \rightarrow)$ where S is a set of states, $s_0 \in S$ is the initial state, and $\rightarrow \subseteq S \times (A \cup \mathbb{R}_{\geq 0}) \times S$ is the transition relation.*

If $(s, \alpha, s') \in \rightarrow$ we write $s \xrightarrow{\alpha} s'$. We implicitly assume that if $s \xrightarrow{0} s'$ then $s = s'$, i.e. zero time delays do not change the current state. The set of *enabled actions* at a state $s \in S$ is defined as $\mathsf{En}(s) \stackrel{\mathsf{def}}{=} \{a \in A \mid \exists s' \in S.\ s \xrightarrow{a} s'\}$. Given a sequence of actions $w = \alpha_1\alpha_2\alpha_3 \ldots \alpha_n \in (A \cup \mathbb{R}_{\geq 0})^*$ we write $s \xrightarrow{w} s'$ iff $s \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} s'$. If there is a sequence w of length n such that $s \xrightarrow{w} s'$, we also write $s \rightarrow^n s'$. Finally, let $\rightarrow^*$ be the reflexive and transitive closure of the relation $\rightarrow$ such that $s \rightarrow s'$ iff there is $\alpha \in \mathbb{R}_{\geq 0} \cup A$ and $s \xrightarrow{\alpha} s'$.

For the rest of this section, we assume a fixed transition system $(S, s_0, \rightarrow)$ and a set of goal states $G \subseteq S$. The *reachability problem*, given a timed transition system $(S, s_0, \rightarrow)$ and a set of goal states G, is to decide whether there is $s' \in G$ such that $s_0 \rightarrow^* s'$.

We now develop the theoretical foundations of stubborn sets for timed transition systems. A state $s \in S$ 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 $s' \in S$ and all $d \in \mathbb{R}_{\geq 0}$ if $s \xrightarrow{d} s'$ then $d = 0$. A *reduction* of a timed transition system is a function $\mathsf{St} : S \rightarrow 2^A$. A reduction defines a reduced transition relation $\xrightarrow[\mathsf{St}]{} \subseteq \rightarrow$ such that $s \xrightarrow[\mathsf{St}]{\alpha} s'$ iff $s \xrightarrow{\alpha} s'$ and $\alpha \in \mathsf{St}(s) \cup \mathbb{R}_{\geq 0}$. For a given state $s \in S$ we define $\overline{\mathsf{St}(s)} \stackrel{\mathsf{def}}{=} A \setminus \mathsf{St}(s)$ as the set of all actions that are not in $\mathsf{St}(s)$.

Definition 2 (Reachability Conditions). *A reduction* St *on a timed transition system $(S, s_0, \rightarrow)$ is* reachability preserving *if it satisfies the following four conditions.*

$(\mathcal{Z})\ \forall s \in S.\ \neg\mathsf{zt}(s) \implies \mathsf{En}(s) \subseteq \mathsf{St}(s)$
$(\mathcal{D})\ \forall s, s' \in S.\ \forall w \in \overline{\mathsf{St}(s)}^*.\ \mathsf{zt}(s) \wedge s \xrightarrow{w} s' \implies \mathsf{zt}(s')$
$(\mathcal{R})\ \forall s, s' \in S.\ \forall w \in \overline{\mathsf{St}(s)}^*.\ \mathsf{zt}(s) \wedge s \xrightarrow{w} s' \wedge s \notin G \implies s' \notin G$
$(\mathcal{W})\ \forall s, s' \in S.\ \forall w \in \overline{\mathsf{St}(s)}^*.\ \forall a \in \mathsf{St}(s).\ \mathsf{zt}(s) \wedge s \xrightarrow{wa} s' \implies s \xrightarrow{aw} s'$

Condition $\mathcal{Z}$ declares that in a state where a delay is possible, all enabled actions become stubborn actions. Condition $\mathcal{D}$ guarantees that in order to enable a time delay from a state where delaying is not allowed, a stubborn action must be executed. Similarly, Condition $\mathcal{R}$ requires that a stubborn action must be executed before a goal state can be reached from a non-goal state. Finally, Condition $\mathcal{W}$ allows us to commute stubborn actions with non-stubborn actions. The following theorem shows that reachability preserving reductions generate pruned transition systems where the reachability of goal states is preserved.

Theorem 1 (Shortest-Distance Reachability Preservation). *Let* St *be a reachability preserving reduction satisfying* $\mathcal{Z}$*,* $\mathcal{D}$*,* $\mathcal{R}$ *and* $\mathcal{W}$*. Let* $s \in S$*. If* $s \to^n s'$ *for some* $s' \in G$ *then also* $s \xrightarrow[\mathsf{St}]{}^m s''$ *for some* $s'' \in G$ *where* $m \leq n$*.*

Proof. We proceed by induction on n. *Base step.* If $n = 0$, then $s = s'$ and $m = n = 0$. *Inductive step.* Let $s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \dots \xrightarrow{\alpha_n} s_{n+1}$ where $s_0 \notin G$ and $s_{n+1} \in G$. Without loss of generality we assume that for all i, $0 \leq i \leq n$, we have $\alpha_i \neq 0$ (otherwise we can simply skip these 0-delay actions and get a shorter sequence). We have two cases. Case $\neg\mathsf{zt}(s_0)$: by condition $\mathcal{Z}$ we have $\mathsf{En}(s_0) \subseteq \mathsf{St}(s_0)$ and by the definition of $\xrightarrow[\mathsf{St}]{}$ we have $s_0 \xrightarrow[\mathsf{St}]{\alpha_0} s_1$ since $\alpha_0 \in \mathsf{En}(s_0) \cup \mathbb{R}_{\geq 0}$. By the induction hypothesis we have $s_1 \xrightarrow[\mathsf{St}]{}^m s''$ with $s'' \in G$ and $m \leq n$ and $m + 1 \leq n + 1$. Case $\mathsf{zt}(s_0)$: let $w = \alpha_0\alpha_1 \dots \alpha_n$ and α_i be such that $\alpha_i \in \mathsf{St}(s_0)$ and for all $k < i$ holds that $\alpha_k \notin \mathsf{St}(s_0)$, i.e. α_i is the first stubborn action in w. Such an α_i has to exist otherwise $s_{n+1} \notin G$ due to condition $\mathcal{R}$. Because of condition $\mathcal{D}$ we get $\mathsf{zt}(s_k)$ for all k, $0 \leq k < i$, otherwise α_i cannot be the first stubborn action in w. We can split w as $w = u\alpha_i v$ with $u \in \overline{\mathsf{St}(s_0)}^*$. Since all states in the path to s_i are zero time, by $\mathcal{W}$ we can swap α_i as $s_0 \xrightarrow{\alpha_i} s'_1 \xrightarrow{u} s_i \xrightarrow{v} s'$ with $|uv| = n$. Since $\alpha_i \in \mathsf{St}(s_0)$ we get $s_0 \xrightarrow[\mathsf{St}]{\alpha_i} s'_1$ and by the induction hypothesis we have $s'_1 \xrightarrow[\mathsf{St}]{}^m s''$ where $s'' \in G$, $m \leq n$, and $m + 1 \leq n + 1$. □

3 Timed-Arc Petri Nets

We shall now define the model of timed-arc Petri nets (as informally described in the introduction) together with a reachability logic and a few technical lemmas needed later on. Let $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$ and $\mathbb{N}_0^\infty = \mathbb{N}_0 \cup \{\infty\}$. We define the set of *well-formed closed time intervals* as $\mathcal{I} \stackrel{\text{def}}{=} \{[a, b] \mid a \in \mathbb{N}_0, b \in \mathbb{N}_0^\infty, a \leq b\}$ and its subset $\mathcal{I}^{\text{inv}} \stackrel{\text{def}}{=} \{[0, b] \mid b \in \mathbb{N}_0^\infty\}$ used in age invariants.

Definition 3 (Timed-Arc Petri Net). *A* timed-arc Petri net *(TAPN) is a 9-tuple* $N = (P, T, T_{urg}, IA, OA, g, w, Type, I)$ *where*

- P *is a finite set of* places,
- T *is a finite set of* transitions *such that* $P \cap T = \emptyset$,
- $T_{urg} \subseteq T$ *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,
- $g : IA \to \mathcal{I}$ *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* $g((p,t)) = [0,\infty]$,
- $w : IA \cup OA \to \mathbb{N}$ *is a function assigning* weights *to input and output arcs,*
- $Type : IA \cup OA \to \mathbf{Types}$ *is a* type function *assigning a type to all arcs where* $\mathbf{Types} = \{Normal, Inhib\} \cup \{Transport_j \mid j \in \mathbb{N}\}$ *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* $w((p,t)) = w((t,p'))$,
- $I : P \to \mathcal{I}^{inv}$ *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 $N = (P, T, T_{urg}, IA, OA, g, w, Type, I)$ be a TAPN. We denote by ${}^\bullet x \stackrel{\text{def}}{=} \{y \in P \cup T \mid (y,x) \in IA \cup OA,\ Type((y,x)) \neq Inhib\}$ the preset of a transition or a place x. Similarly, the postset is defined as $x^\bullet \stackrel{\text{def}}{=} \{y \in P \cup T \mid (x,y) \in (IA \cup OA)\}$. We denote by ${}^\circ t \stackrel{\text{def}}{=} \{p \in P \mid (p,t) \in IA \wedge Type((p,t) = Inhib\}$ the inhibitor preset of a transition t. The inhibitor postset of a place p is defined as $p^\circ \stackrel{\text{def}}{=} \{t \in T \mid (p,t) \in IA \wedge Type((p,t) = Inhib\}$. 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 TAPN (N, M_0) is a TAPN N together with an initial marking M_0 with all tokens of age 0.

Definition 4 (Enabledness). *Let* $N = (P, T, T_{urg}, IA, OA, g, w, Type, I)$ *be a TAPN. 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.*

$$\forall (p,t) \in IA.\, Type((p,t)) = Inhib \Rightarrow |M(p)| < w((p,t))$$

– *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.*

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

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

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

A given marked TAPN (N, M_0) defines a timed transition system $T(N) \stackrel{\text{def}}{=} (\mathcal{M}(N), M_0, \rightarrow)$ where the states are markings and the transitions are 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.

Note that the semantics above defines the discrete-time semantics as the delays are restricted to nonnegative integers. It is well known that for timed-arc Petri nets with nonstrict intervals, the marking reachability problem on discrete and continuous time nets coincide [31]. This is, however, not the case for more complex properties like liveness that can be expressed in the CTL logic (for counter examples that can be expressed in CTL see e.g. [25]).

3.1 Reachability Logic and Interesting Sets of Transitions

We now describe a logic for expressing the properties of markings based on the number of tokens in places and transition enabledness, inspired by the logic used in the Model Checking Contest (MCC) Property Language [27]. Let $N = (P, T, T_{urg}, IA, OA, g, w, Type, I)$ be a TAPN. The formulae of the logic are given by the abstract syntax:

Formula φ	$A_M(\varphi)$	$A_M(\neg\varphi)$
deadlock	$({}^\bullet t)^\bullet \cup {}^\bullet({}^\circ t)$ for some $t \in \mathsf{En}(M)$	$\emptyset$
t	${}^\bullet p$ for some $p \in {}^\bullet t$ where $M(p) < w((p,t))$ or $p^\bullet$ for some $p \in {}^\circ t$ where $M(p) \geq w((p,t))$	$({}^\bullet t)^\bullet \cup {}^\bullet({}^\circ t)$
$e_1 < e_2$	$decr_M(e_1) \cup incr_M(e_2)$	$A_M(e_1 \geq e_2)$
$e_1 \leq e_2$	$decr_M(e_1) \cup incr_M(e_2)$	$A_M(e_1 > e_2)$
$e_1 > e_2$	$incr_M(e_1) \cup decr_M(e_2)$	$A_M(e_1 \leq e_2)$
$e_1 \geq e_2$	$incr_M(e_1) \cup decr_M(e_2)$	$A_M(e_1 < e_2)$
$e_1 = e_2$	$decr_M(e_1) \cup incr_M(e_2)$ if $eval_M(e_1) > eval_M(e_2)$ $incr_M(e_1) \cup decr_M(e_2)$ if $eval_M(e_1) < eval_M(e_2)$	$A_M(e_1 \neq e_2)$
$e_1 \neq e_2$	$incr_M(e_1) \cup decr_M(e_1) \cup incr_M(e_2) \cup decr_M(e_2)$	$A_M(e_1 = e_2)$
$\varphi_1 \wedge \varphi_2$	$A_M(\varphi_i)$ for some $i \in \{1,2\}$ where $M \not\models \varphi_i$	$A_M(\neg\varphi_1 \vee \neg\varphi_2)$
$\varphi_1 \vee \varphi_2$	$A_M(\varphi_1) \cup A_M(\varphi_2)$	$A_M(\neg\varphi_1 \wedge \neg\varphi_2)$

Table 1: Interesting transitions of φ (assuming $M \not\models \varphi$, otherwise $A_M(\varphi) = \emptyset$)

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

Table 2: Increasing and decreasing transitions of expression e

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

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

where $t \in T$, $\bowtie \in \{<, \leq, =, \neq, >, \geq\}$, $c \in \mathbb{Z}$, $p \in P$, and $\oplus \in \{+, -, *\}$. Let Φ be the set of all such formulae and let E_N be the set of arithmetic expressions over the net N. The semantics of φ in a marking $M \in \mathcal{M}(N)$ is given by

$$\begin{aligned} M &\models \mathit{deadlock} && \text{if } \mathsf{En}(M) = \emptyset \\ M &\models t && \text{if } t \in \mathsf{En}(M) \\ M &\models e_1 \bowtie e_2 && \text{if } eval_M(e_1) \bowtie eval_M(e_2) \end{aligned}$$

assuming a standard semantics for Boolean operators and where the semantics of arithmetic expressions in a marking M is as follows: $eval_M(c) = c$, $eval_M(p) = |M(p)|$, and $eval_M(e_1 \oplus e_2) = eval_M(e_1) \oplus eval_M(e_2)$.

Let φ be a formula. We are interested in the question, whether we can reach from the initial marking some of the goal markings from $G_\varphi = \{M \in \mathcal{M}(N) \mid M \models \varphi\}$. In order to guide the reduction such that transitions that lead to the goal markings are included in the generated stubborn set, we define the notion

of *interesting transitions* for a marking M relative to φ, and we let $A_M(\varphi) \subseteq T$ denote the set of interesting transitions. Formally, we shall require that whenever $M \xrightarrow{w} M'$ via a sequence of transitions $w = t_1 t_2 \dots t_n \in T^*$ where $M \notin G_\varphi$ and $M' \in G_\varphi$, then there must exist i, $1 \leq i \leq n$, such that $t_i \in A_M(\varphi)$.

Table 1 gives a possible definition of $A_M(\varphi)$. Let us remark that the definition is at several places nondeterministic, allowing for a variety of sets of interesting transitions. Table 1 uses the functions $incr_M : E_N \to 2^T$ and $decr_M : E_N \to 2^T$ defined in Table 2. These functions take as input an expression e, and return all transitions that can possibly, when fired, increase resp. decrease the evaluation of e. The following lemma formally states the required property of the functions $incr_M$ and $decr_M$.

Lemma 1. *Let $N = (P, T, T_{urg}, IA, OA, g, w, Type, I)$ be a TAPN and $M \in \mathcal{M}(N)$ a marking. Let $e \in E_N$ and let $M \xrightarrow{w} M'$ where $w = t_1 t_2 \dots t_n \in T^*$.*

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

We finish this section with the main technical lemma, showing that at least one interesting transition must be fired before we can reach a marking satisfying a given reachability formula.

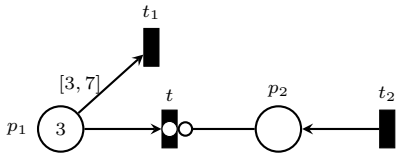
Lemma 2. *Let $N = (P, T, T_{urg}, IA, OA, g, w, Type, I)$ be a TAPN, let $M \in \mathcal{M}(N)$ be its marking and let $\varphi \in \Phi$ be a given formula. If $M \not\models \varphi$ and $M \xrightarrow{w} M'$ where $w \in \overline{A_M(\varphi)}^*$ then $M' \not\models \varphi$.*

4 Partial Order Reductions for TAPN

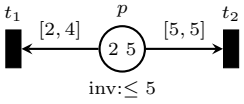
We are now ready to state the main theorem that provides sufficient syntax-driven conditions for a reduction in order to guarantee preservation of reachability. Let $N = (P, T, T_{urg}, IA, OA, g, w, Type, I)$ be a TAPN, let $M \in \mathcal{M}(N)$ be a marking of N, and let $\varphi \in \Phi$ be a formula. We recall that $A_M(\varphi)$ is the set of interesting transitions as defined earlier.

Theorem 2 (Reachability Preserving Closure). *Let St be a reduction such that for all $M \in \mathcal{M}(N)$ it satisfies the following conditions.*

1. *If $\neg \mathsf{zt}(M)$ then $\mathsf{En}(M) \subseteq \mathsf{St}(M)$.*
2. *If $\mathsf{zt}(M)$ then $A_M(\varphi) \subseteq \mathsf{St}(M)$.*
3. *If $\mathsf{zt}(M)$ then either*
   (a) *there is $t \in T_{urg} \cap \mathsf{En}(M) \cap \mathsf{St}(M)$ where ${}^\bullet({}^\circ t) \subseteq \mathsf{St}(M)$, or*
   (b) *there is $p \in P$ where $I(p) = [a, b]$ and $b \in M(p)$ such that $t \in \mathsf{St}(M)$ for every $t \in p^\bullet$ where $b \in g((p, t))$.*
4. *For all $t \in \mathsf{St}(M) \setminus \mathsf{En}(M)$ either*
   (a) *there is $p \in {}^\bullet t$ such that $|\{x \in M(p) \mid x \in g((p, t))\}| < w((p, t))$ and*
      - *$t' \in \mathsf{St}(M)$ for all $t' \in {}^\bullet p$ where there is $p' \in {}^\bullet t'$ with $Type((t', p)) = Type((p', t')) = Transport_j$ and where $g((p', t')) \cap g((p, t)) \neq \emptyset$, and*



(a) Transitions t_1 and t_2 can disable resp. inhibit the urgent transition t



(b) Transition t_2 can remove the token of age 5 from p

Fig. 2: Cases for Condition 3

– *if* $0 \in g((p,t))$ *then also* ${}^\bullet p \subseteq \mathsf{St}(M)$*, or*

(b) there is $p \in {}^\circ t$ *where* $|M(p)| \geq w((p,t))$ *such that*

– $t' \in \mathsf{St}(M)$ *for all* $t' \in p^\bullet$ *where* $M(p) \cap g((p,t')) \neq \emptyset$*.*

5 For all $t \in \mathsf{St}(M) \cap \mathsf{En}(M)$ *we have*

(a) $t' \in \mathsf{St}(M)$ *for every* $t' \in p^\bullet$ *where* $p \in {}^\bullet t$ *and* $g((p,t)) \cap g((p,t')) \neq \emptyset$*, and*

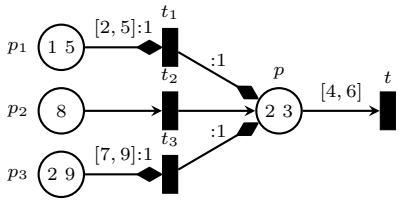
(b) $(t^\bullet)^\circ \subseteq \mathsf{St}(M)$*.*

Then St *satisfies* $\mathcal{Z}$*,* $\mathcal{D}$*,* $\mathcal{R}$*, and* $\mathcal{W}$*.*

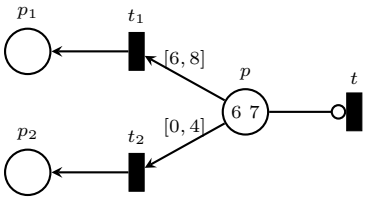
Let us now briefly discuss the conditions of Theorem 2. Clearly, Condition 1 ensures that if time can elapse, we include all enabled transitions into the stubborn set and Condition 2 guarantees that all interesting transitions (those that can potentially make the reachability proposition true) are included as well.

Condition 3 makes sure that that if time elapsing is disabled then any transition that can possibly enable time elapsing will be added to the stubborn set. There are two situations how time progress can be disabled. Either, there is an urgent enabled transition, like the transition t in Figure 2a. Since t_2 can add a token to p_2 and by that inhibit t, Condition 3a makes sure that t_2 is added into the stubborn set in order to satisfy $\mathcal{D}$. As t_1 can remove the token of age 3 from p_1 and hence disable t, we must add t_1 to the stubborn set too (guaranteed by Condition 5a). The other situation when time gets stopped is when a place with an age invariant contains a token that disallows time passing, like in Figure 2b where time is disabled because the place p has a token of age 5, which is the maximum possible age of tokens in p due to the age invariant. Since t_2 can remove the token of age 5 from p, we include it to the stubborn set due to Condition 3b. On the other hand t_1 does not have to be included in the stubborn set as its firing cannot remove the token of age 5 from p.

Condition 4 makes sure that an disabled stubborn transition can never be enabled by a non-stubborn transition. There are two reasons why a transition is disabled. Either, as in Figure 3a where t is disabled, there is an insufficient number of tokens of appropriate age to fire the transition. In this case, Condition 4a makes sure that transitions that can add tokens of a suitable age via transport arcs are included in the stubborn set. This is the case for the transition t_1 in our example, as $[2,5]$ has a nonempty intersection with $[4,6]$. On the other hand, t_3 does not have to be added. As the transition t_2 only adds fresh tokens of age 0

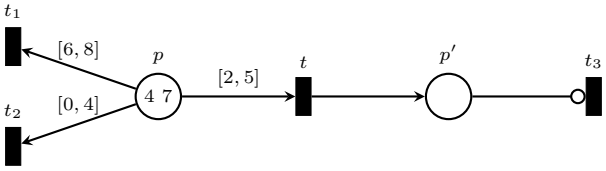


(a) Transition t_1 can transport well-aged tokens into p and enable t



(b) Transition t_1 can enable t by removing tokens from p

Fig. 3: Cases for Condition 4



(a) Stubborn transition t can disable both t_2 and t_3

Fig. 4: Cases for Condition 5

to p via normal arcs, there is no need to add t_2 into the stubborn set either. The other reason for a transition to be disabled is due to inhibitor arcs, as shown on the transition t in Figure 3b. Condition 4b makes sure that t_1 is added to the stubborn set, as it can enable t (the interval $[6, 8]$ has a nonempty intersection with the tokens of age 6 and 7 in the place p). As this is not the case for t_2, this transition can be left out from the stubborn set.

Finally, Condition 5 guarantees that enabled stubborn transitions can never disable any non-stubborn transitions. For an illustration, take a look at Figure 4a and assume that t is an enabled stubborn transition. Firing of t can remove the token of age 4 from p and disable t_2, hence t_2 must become stubborn by Condition 5a in order to satisfy $\mathcal{W}$. On the other hand, the intervals $[6, 8]$ and $[2, 5]$ have empty intersection, so there is no need to declare t_1 as a stubborn transition. Moreover, firing of t can also disable the transition t_3 due to the inhibitor arc, so we must add t_3 to the stubborn set by Condition 5b.

The conditions of Theorem 2 can be turned into an iterative saturation algorithm for the construction of stubborn sets as shown in Algorithm 1. When running this algorithm for the net in our running example, we can reduce the state space exploration for fireability of the transition t as depicted in Figure 1b. Our last theorem states that the algorithm returns stubborn subsets of enabled transitions that satisfy the four conditions of Theorem 1 and hence we preserve the reachability property as well as the minimum path to some reachable goal.

Theorem 3. *Algorithm 1 terminates and returns* $\mathsf{St}(M) \cap \mathsf{En}(M)$ *for some reduction* St *that satisfies* $\mathcal{Z}$*,* $\mathcal{D}$*,* $\mathcal{R}$*, and* $\mathcal{W}$*.*

Algorithm 1: Construction of a reachability preserving stubborn set

input : $N = (P, T, T_{urg}, IA, OA, g, w, Type, I)$, $M \in \mathcal{M}(N)$, $\varphi \in \Phi$
output : $\mathsf{St}(M) \cap \mathsf{En}(M)$

if $\neg\mathsf{zt}(M)$ **then**
  return $\mathsf{En}(M)$;
$X := \emptyset$; $Y := A_M(\varphi)$;
if $T_{urg} \cap \mathsf{En}(M) \neq \emptyset$ **then**
  pick any $t \in T_{urg} \cap \mathsf{En}(M)$;
  if $t \notin Y$ **then**
    $Y := Y \cup \{t\}$;
  $Y := Y \cup {}^\bullet({}^\circ t)$;
else
  pick any $p \in P$ where $I(p) = [a, b]$ and $b \in M(p)$
  forall the $t \in p^\bullet$ **do**
    if $b \in g((p, t))$ **then**
      $Y := Y \cup \{t\}$;
while $Y \neq \emptyset$ **do**
  pick any $t \in Y$;
  if $t \notin \mathsf{En}(M)$ **then**
    if $\exists p \in {}^\bullet t.\ |\{x \in M(p) \mid x \in g((p, t))\}| < w((p, t))$ **then**
      pick any such p;
      forall the $t' \in {}^\bullet p \setminus X$ **do**
        forall the $p' \in {}^\bullet t'$ **do**
          if $Type((t', p)) = Type((p', t')) = Transport_j \wedge g((p', t')) \cap g((p, t)) \neq \emptyset$ **then**
            $Y := Y \cup \{t'\}$;
      if $0 \in g((p, t))$ **then**
        $Y := Y \cup ({}^\bullet p \setminus X)$;
    else
      pick any $p \in {}^\circ t$ s.t. $|M(p)| \geq w((p, t))$;
      forall the $t' \in p^\bullet \setminus X$ **do**
        if $M(p) \cap g((p, t')) \neq \emptyset$ **then**
          $Y := Y \cup \{t'\}$;
  else
    forall the $p \in {}^\bullet t$ **do**
      $Y := Y \cup (\{t' \in p^\bullet | g((p, t)) \cap g((p, t')) \neq \emptyset\} \setminus X)$;
    $Y := Y \cup ((t^\bullet)^\circ \setminus X)$;
  $Y := Y \setminus \{t\}$;
  $X := X \cup \{t\}$;
return $X \cap \mathsf{En}(M)$;

5 Implementation and Experiments

We implemented our partial order method in C++ and integrated it within the model checker TAPAAL [19] and its discrete time engine `verifydtapn` [4, 11]. We evaluate our partial order reduction on a wide range of case studies.

PatientMonitoring. The patient monitoring system [17] models a medical system that through sensors periodically scans patient's vital functions, making sure that abnormal situations are detected and reported within given deadlines. The timed-arc Petri net model was described in [17] for two sensors monitoring patients pulse rate and oxygen saturation level. We scale the case study by adding additional sensors. *BloodTransfusion.* This case study models a larger blood transfusion workflow [16], the benchmarking case study of the little-JIL language. The timed-arc Petri net model was described in [10] and we verify that the workflow is free of deadlocks (unless all sub-workflows correctly terminate). The problem is scaled by the number of patients receiving a blood transfusion. *FireAlarm.* This case study uses a modified (due to trade secrets) fire alarm system owned by a German company [20, 21]. It models a four-channel round-robin frequency-hopping transmission scheduling in order to ensure a reliable communication between a number of wireless sensors (by which the case study is scaled) and a central control unit. The protocol is based on time-division multiple access (TDMA) channel access and we verify that for a given frequency-jammer, it takes never more than three cycles before a fire alarm is communicated to the central unit. *BAwPC.* Business Activity with Participant Completion (BAwPC) is a web-service coordination protocol from WS-BA specification [33] that ensures a consistent agreement on the outcome of long-running distributed applications. In [26] it was shown that the protocol is flawed and a correct, enhanced variant was suggested. We model check this enhanced protocol and scale it by the capacity of the communication buffer. *Fischer.* Here we consider a classical Fischers protocol for ensuring mutual exclusion for a number of timed processes. The timed-arc Petri net model is taken from [2] and it is scaled by the number of processes. *LynchShavit.* This is another timed-based mutual exclusion algorithm by Lynch and Shavit, with the timed-arc Petri net model taken from [1] and scaled by the number of processes. *MPEG2.* This case study describes the workflow of the MPEG-2 video encoding algorithm run on a multicore processor (the timed-arc Petri net model was published in [35]) and we verify the maximum duration of the workflow. The model is scaled by the number of B frames in the IB^nP frame sequence. *AlternatingBit.* This is a classical case study of alternating bit protocol, based on the timed-arc Petri net model given in [24]. The purpose of the protocol is to ensure a safe communication between a sender and a receiver over an unreliable medium. Messages are time-stamped in order to compensate (via retransmission) for the possibility of losing messages. The case study is scaled by the maximum number of messages in transfer.

All experiments were run on AMD Opteron 6376 Processors with 500 GB memory. In Table 3 we compare the time to verify a model without (NORMAL) and with (POR) partial order reduction, the number of explored markings (in thousands) and the percentage of time and memory reduction. We can observe

	Time (seconds)		Markings ×1000		Reduction	
Model	NORMAL	POR	NORMAL	POR	%Time	%Markings
PatientMonitoring 3	5.88	0.35	333	28	94	92
PatientMonitoring 4	22.06	0.48	1001	36	98	96
PatientMonitoring 5	80.76	0.65	3031	44	99	99
PatientMonitoring 6	305.72	0.85	9248	54	100	99
PatientMonitoring 7	5516.93	5.75	130172	318	100	100
BloodTransfusion 2	0.32	0.41	48	43	-28	11
BloodTransfusion 3	7.88	6.45	792	546	18	31
BloodTransfusion 4	225.18	109.30	14904	7564	51	49
BloodTransfusion 5	5256.01	1611.14	248312	94395	69	62
FireAlarm 10	28.95	14.17	796	498	51	37
FireAlarm 12	116.97	17.51	1726	526	85	70
FireAlarm 14	598.89	21.65	5367	554	96	90
FireAlarm 16	5029.25	29.48	19845	582	99	97
FireAlarm 18	27981.90	34.55	77675	610	100	99
FireAlarm 20	154495.29	41.47	308914	638	100	100
FireAlarm 80	$>$ 2 days	602.71	-	1522	-	-
FireAlarm 125	$>$ 2 days	1957.00	-	2260	-	-
BAwPC 2	0.21	0.41	19	16	-95	15
BAwPC 4	3.45	4.04	193	125	-17	35
BAwPC 6	23.01	17.08	900	452	26	50
BAwPC 8	73.73	39.29	2294	952	47	58
BAwPC 10	135.62	60.66	3819	1412	55	63
BAwPC 12	173.09	73.53	4736	1665	58	65
Fischer-9	3.24	2.37	281	233	27	17
Fischer-11	12.68	8.73	923	738	31	20
Fischer-13	42.52	28.53	2628	2041	33	22
Fischer-15	121.31	77.50	6700	5066	36	24
Fischer-17	313.69	198.36	15622	11536	37	26
Fischer-19	748.52	456.30	33843	24469	39	28
Fischer-21	1622.69	985.07	68934	48904	39	29
LynchShavit 9	3.98	3.31	282	234	17	17
LynchShavit 11	15.73	12.19	925	740	23	20
LynchShavit 13	51.08	37.97	2631	2043	26	22
LynchShavit 15	146.63	103.63	6703	5069	29	24
LynchShavit 17	384.52	258.09	15626	11540	33	26
LynchShavit 19	907.60	597.68	33848	24474	34	28
LynchShavit 21	2011.58	1307.72	68940	48910	35	29
MPEG2 3	13.17	15.43	2188	2187	-17	0
MPEG2 4	109.62	125.45	15190	15180	-14	0
MPEG2 5	755.54	840.84	87568	87478	-11	0
MPEG2 6	4463.19	5092.58	435023	434354	-14	0
AlternatingBit 20	9.17	9.51	617	617	-4	0
AlternatingBit 30	48.20	49.13	2804	2804	-2	0
AlternatingBit 40	161.18	162.94	8382	8382	-1	0
AlternatingBit 50	408.34	408.86	19781	19781	0	0

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

clear benefits of our technique on PatientMonitoring, BloodTransfusion and FireAlarm where we are both exponentially faster and explore only a fraction of all reachable markings. For example in FireAlarm, we are able to verify its correctness for all 125 sensors, as it is required by the German company [21]. This would be clearly unfeasible without the use of partial order reduction.

In BAwPC, we can notice that for the smallest instances, there is some computation overhead from computing the stubborn sets, however, it clearly pays off for the larger instances where the percentages of reduced state space are closely followed by the percentages of the verification times and in fact improve with the larger instances. Fischer and LynchShavit case studies demonstrate that even moderate reductions of the state space imply considerable reduction in the running time and computing the stubborn sets is well worth the extra effort.

MPEG2 is an example of a model that allows only negligible reduction of the state space size, and where we observe an actual slowdown in the running time due to the computation of the stubborn sets. Nevertheless, the overhead stays constant in the range of about 15%, even for increasing instance sizes. Finally, AlternatingBit protocol does not allow for any reduction of the state space (even though it contains age invariants) but the overhead in the running time is negligible.

We observed similar performance of our technique also for the cases where the reachability property does not hold and a counter example can be generated.

6 Conclusion

We suggested a simple, yet powerful and application-ready partial order reduction for timed systems. The reduction comes into effect as soon as the timed system enters an urgent configuration where time cannot elapse until a nonempty sequence of transitions gets executed. The method is implemented and fully integrated, including GUI support, into the open-source tool TAPAAL. We demonstrated its practical applicability on several case studies and conclude that computing the stubborn sets causes only a minimal overhead while providing large benefits for reducing the state space in numerous models. The method is not specific to stubborn reduction technique only and it preserves the shortest execution sequences. Moreover, once the time gets urgent, other classical (untimed) partial order approaches should be applicable too. Our method was instantiated to (unbounded) timed-arc Petri nets with discrete time semantics, however, we claim that the technique allows for general application to other modelling formalisms like timed automata and timed Petri nets, as well as an extension to continuous time. We are currently working on adapting the theory and providing an efficient implementation for UPPAAL-style timed automata with continuous time semantics.

Acknowledgements. We thank Mads Johannsen for his help with the GUI support for partial order reduction. The work was funded by the center IDEA4CPS, Innovation Fund Denmark center DiCyPS and ERC Advanced Grant LASSO. The last author is partially affiliated with FI MU in Brno.

References

1. P. Abdulla, J. Deneux, P. Mahata, and A. Nylén. Using forward reachability analysis for verification of timed Petri nets. *Nordic J. of Computing*, 14:1–42, 2007.
2. P. Abdulla and A. Nylén. Timed Petri nets and BQOs. In *Proceedings of the 22nd International Conference on Application and Theory of Petri Nets (ICATPN'01)*, volume 2075 of *LNCS*, pages 53–70. Springer-Verlag, 2001.
3. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, Apr. 1994.
4. M. Andersen, H. Larsen, J. Srba, M. Sørensen, and J. Taankvist. Verification of liveness properties on closed timed-arc Petri nets. In *MEMICS'13*, volume 7721 of *LNCS*, pages 69–81. Springer, 2013.
5. E. André, T. Chatain, and C. Rodríguez. Preserving partial-order runs in parametric time Petri nets. *ACM Trans. Embed. Comput. Syst.*, 16(2):43:1–43:26, 2017.
6. C. Baier and J.-P. Katoen. *Principles of Model Checking*. The MIT Press, 2008.
7. G. Behrmann, P. Bouyer, K. G. Larsen, and R. Pelánek. Lower and upper bounds in zone-based abstractions of timed automata. *STTT*, 8(3):204–215, 2006.
8. J. Bengtsson, B. Jonsson, J. Lilius, and W. Yi. Partial order reductions for timed systems. In *CONCUR'98*, volume 1466 of *LNCS*, pages 485–500. Springer, 1998.
9. B. Berthomieu and F. Vernadat. Time Petri nets analysis with TINA. In *Third International Conference on Quantitative Evaluation of Systems*, pages 123–124. IEEE Computer Society, 2006.
10. C. Bertolini, Z. Liu, and J. Srba. Verification of timed healthcare workflows using component timed-arc Petri nets. In *FHIES'12*, volume 7789 of *LNCS*, pages 19–36. Springer-Verlag, 2013.
11. S. Birch, T. Jacobsen, J. Jensen, C. Moesgaard, N. Samuelsen, and J. Srba. Interval abstraction refinement for model checking of timed-arc Petri nets. In *FORMATS'14*, volume 8711 of *LNCS*, pages 237–251. Springer, 2014.
12. H. Boucheneb and K. Barkaoui. Reducing interleaving semantics redundancy in reachability analysis of time Petri nets. *ACM Trans. Embed. Comput. Syst.*, 12(1):7:1–7:24, 2013.
13. H. Boucheneb and K. Barkaoui. Stubborn sets for time Petri nets. *ACM Trans. Embed. Comput. Syst.*, 14(1):11:1–11:25, 2015.
14. H. Boucheneb and K. Barkaoui. Delay-dependent partial order reduction technique for real time systems. *Real-Time Systems*, 2017.
15. M. Bozga, S. Graf, I. Ober, I. Ober, and J. Sifakis. The IF toolset. In *International School on Formal Methods for the Design of Computer, Communication and Software Systems (SFM-RT'04)*, volume 3185 of *LNCS*, pages 237–267. Springer, 2004.
16. S. Christov, G. Avrunin, A. Clarke, L. Osterweil, and E. Henneman. A benchmark for evaluating software engineering techniques for improving medical processes. In *SEHC'10*, pages 50–56. ACM, 2010.
17. F. Cicirelli, A. Furfaro, and L. Nigro. Model checking time-dependent system specifications using time stream Petri nets and UPPAAL. *Applied Mathematics and Computation*, 218(16):8160–8186, 2012.
18. D. Dams, R. Gerth, B. Knaack, and R. Kuiper. Partial-order reduction techniques for real-time model checking. *Formal Asp. Comput.*, 10(5-6):469–482, 1998.
19. A. David, L. Jacobsen, M. Jacobsen, K. Jørgensen, M. Møller, and J. Srba. TAPAAL 2.0: Integrated development environment for timed-arc Petri nets. In *TACAS'12*, volume 7214 of *LNCS*, pages 492–497. Springer, 2012.

20. 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*, pages 658–672. Springer International Publishing, Cham, 2014.
21. S. Feo-Arenis, B. Westphal, D. Dietsch, M. Muñiz, S. Andisha, and A. Podelski. Ready for testing: ensuring conformance to industrial standards through formal verification. *Formal Aspects of Computing*, 28(3):499–527, May 2016.
22. G. Gardey, D. Lime, M. Magnin, and O. Roux. Romeo: A tool for analyzing time Petri nets. In *Computer Aided Verification: 17th International Conference*, volume 3576 of *LNCS*, pages 261–272. Springer, 2005.
23. H. Hansen, S. Lin, Y. Liu, T. K. Nguyen, and J. Sun. Diamonds are a girl's best friend: Partial order reduction for timed automata with abstractions. In *CAV'14*, volume 8559 of *LNCS*, pages 391–406. Springer, 2014.
24. L. Jacobsen, M. Jacobsen, M. Møller, and J. Srba. Verification of timed-arc Petri nets. In *Proceedings of the 37th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'11)*, volume 6543 of *LNCS*, pages 46–72. Springer-Verlag, 2011.
25. P. Jensen, K. Larsen, and J. Srba. Discrete and continuous strategies for timed-arc Petri net games. *International Journal on Software Tools for Technology Transfer (STTT)*, pages 1–18, 2017. Online since September 2017. To appear.
26. A. M. Jr., A. Ravn, J. Srba, and S. Vighio. Model-checking web services business activity protocols. *International Journal on Software Tools for Technology Transfer (STTT)*, 15(2):125–147, 2012.
27. F. Kordon, H. Garavel, L. M. Hillah, F. Hulin-Hubard, G. Chiardo, A. Hamez, L. Jezequel, A. Miner, J. Meijer, E. Paviot-Adet, D. Racordon, C. Rodriguez, C. Rohr, J. Srba, Y. Thierry-Mieg, G. Trịnh, and K. Wolf. Complete Results for the 2016 Edition of the Model Checking Contest. http://mcc.lip6.fr/2016/results.php, June 2016.
28. L. M. Kristensen, K. Schmidt, and A. Valmari. Question-guided stubborn set methods for state properties. *Formal Methods in System Design*, 29(3):215–251, 2006.
29. J. Lilius. Efficient state space search for time Petri nets. *Electronic Notes in Theoretical Computer Science*, 18:113 – 133, 1998. MFCS'98 Workshop on Concurrency.
30. D. Lugiez, P. Niebert, and S. Zennou. A partial order semantics approach to the clock explosion problem of timed automata. *Theor. Comput. Sci.*, 345(1):27–59, 2005.
31. J. Mateo, J. Srba, and M. Sørensen. Soundness of timed-arc workflow nets in discrete and continuous-time semantics. *Fundamenta Informaticae*, 140(1):89–121, 2015.
32. M. Minea. Partial order reduction for model checking of timed automata. In *CONCUR'99*, volume 1664 of *LNCS*, pages 431–446. Springer, 1999.
33. E. Newcomer and I. Robinson. Web services business activity (WS-businessactivity) version 1.2, 2009. http://docs.oasis-open.org/ws-tx/wstx-wsba-1.2-spec-os/wstx-wsba-1.2-spec-os.html.
34. P. Niebert and H. Qu. Adding invariants to event zone automata. In *FORMATS'06*, volume 4202 of *LNCS*, pages 290–305. Springer, 2006.
35. F. Pelayo, F. Cuartero, V. Valero, H. Macia, and M. Pelayo. Applying timed-arc Petri nets to improve the performance of the MPEG-2 encoding algorithm. In *10th International Multimedia Modelling Conference*, pages 49–56. IEEE Computer Society, 2004.

36. M. Perin and J. Faure. Coupling timed plant and controller models with urgent transitions without introducing deadlocks. In *17th International Conference on Emerging Technologies & Factory Automation (ETFA'12)*, pages 1–9. IEEE, 2012.
37. R. B. Salah, M. Bozga, and O. Maler. On interleaving in timed automata. In *CONCUR'06*, LNCS, pages 465–476, 2006.
38. R. H. Sloan and U. Buy. Stubborn sets for real-time Petri nets. *Formal Methods in System Design*, 11(1):23–40, 1997.
39. A. Valmari and H. Hansen. *Stubborn Set Intuition Explained*, pages 140–165. Springer, 2017.
40. I. Virbitskaite and E. Pokozy. A partial order method for the verification of time Petri nets. In G. Ciobanu and G. Păun, editors, *Fundamentals of Computation Theory*, pages 547–558. Springer, 1999.
41. T. Yoneda and B.-H. Schlingloff. Efficient verification of parallel real-time systems. *Formal Methods in System Design*, 11(2):187–215, 1997.

A Proof of Lemma 1

Proof. Let $N = (P, T, T_{urg}, IA, OA, g, w, Type, I)$ be a TAPN and $M \in \mathcal{M}(N)$ a marking. Let $e \in E_N$ and let $M \xrightarrow{w} M'$ where $w = t_1 t_2 \dots t_n \in T^*$. The proof proceeds by structural induction on e. We only show the *incr* cases as it is analogous to *decr*.

- $e = c$: Trivial
- $e = p$: We here prove that if for all $i \in [1, n]$ we have $t_i \notin incr_M(p)$ then we have that $eval_M(p) \geq eval_{M'}(p)$, which implies Lemma 1 for this case. We know ${}^\bullet p \subseteq incr_M(p)$ by Table 2. Therefore for all $i \in [1, n]$ we have $t_i \notin {}^\bullet p$ and we must have that $|M(p)| \geq |M'(p)|$ and $eval_M(p) \geq eval_{M'}(p)$, since t_i cannot increase the number of tokens in p as it would otherwise contradict the definition of ${}^\bullet p$.
- $e = e_1 + e_2$: In Table 2 we see that $incr_M(e) = incr_M(e_1) \cup incr_M(e_2)$. By the induction hypothesis we know $eval_M(e_1) < eval_{M'}(e_1)$ and $eval_M(e_2) < eval_{M'}(e_2)$, and therefore also $eval_M(e) < eval_{M'}(e)$ since we have $eval_M(e) = eval_M(e_1) + eval_M(e_2) < eval_{M'}(e_1) + eval_{M'}(e_2) = eval_{M'}(e)$.
- $e = e_1 - e_2$: In Table 2 we see that $incr_M(e) = incr_M(e_1) \cup decr_M(e_2)$. By the induction hypothesis we know $eval_M(e_1) < eval_{M'}(e_1)$ and $eval_M(e_2) > eval_{M'}(e_2)$, and therefore also $eval_M(e) < eval_{M'}(e)$ since we have $eval_M(e) = eval_M(e_1) - eval_M(e_2) < eval_{M'}(e_1) - eval_{M'}(e_2) = eval_{M'}(e)$.
- $e = e_1 * e_2$: We know $eval_M(e) < eval_{M'}(e)$ and $incr_M(e) = incr_M(e_1) \cup decr_M(e_1) \cup incr_M(e_2) \cup decr_M(e_2)$ due to Table 2. If for all $i \in [1, n]$ we have $t_i \notin incr_M(e)$ then $eval_M(e_1) = eval_{M'}(e_1)$ and $eval_M(e_2) = eval_{M'}(e_2)$ and therefore $eval_M(e) = eval_{M'}(e)$. Since $eval_M(e) < eval_{M'}(e)$ we must have that there exists i, $1 \leq i \leq n$, such that $t_i \in incr_M(e)$.

□

B Proof of Lemma 2

Proof. Let $N = (P, T, T_{urg}, IA, OA, g, w, Type, I)$ be a TAPN, $M \in \mathcal{M}(N)$ be a marking, and $\varphi \in \Phi$ a given formula. Assume that $M \not\models \varphi$. The proof proceeds by structural induction on φ.

- $\varphi = deadlock$: If $M \not\models \varphi$ then there exists $t \in \mathsf{En}(M)$. To disable t we have to fire a transition $t' \in ({}^\bullet t)^\bullet \cup {}^\bullet({}^\circ t)$ to either remove or add tokens that will disable or inhib t, respectively. Since $({}^\bullet t)^\bullet \cup {}^\bullet({}^\circ t) \subseteq A_M(\varphi)$ we have $M' \not\models \varphi$.
- $\varphi = t$: If $M \not\models \varphi$ then $t \notin \mathsf{En}(M)$. Either there exists $p \in {}^\bullet t$ s.t. $|M(p)| < w((p, t))$ and we have to fire some $t' \in {}^\bullet p$ to enable t, or there exists $p \in {}^\circ t$ s.t. $|M(p)| \geq w((p, t))$ and we have to fire some $t'' \in p^\bullet$ to enabled t. Since ${}^\bullet p \subseteq A_M(\varphi)$ or $p^\bullet \subseteq A_M(\varphi)$, respectively, we have $M' \not\models \varphi$.
- $\varphi = e_1 < e_2$: If $M \not\models \varphi$ then $eval_M(e_1) \geq eval_M(e_2)$. Since $decr_M(e_1) \cup incr_M(e_2) \subseteq A_M(\varphi)$ we know $eval_M(e_1) \geq eval_{M'}(e_1)$ and $eval_M(e_2) \leq eval_{M'}(e_2)$ because of Lemma 1, and therefore $M' \not\models \varphi$.

- $\varphi = e_1 > e_2$: If $M \not\models \varphi$ then $eval_M(e_1) \leq eval_M(e_2)$. Since $incr_M(e_1) \cup decr_M(e_2) \subseteq A_M(\varphi)$ we know $eval_M(e_1) \leq eval_{M'}(e_1)$ and $eval_M(e_2) \geq eval_{M'}(e_2)$ because of Lemma 1, and therefore $M' \not\models \varphi$.
- $\varphi = \varphi_1 \wedge \varphi_2$: If $M \not\models \varphi$ then there exists $i \in \{1, 2\}$ s.t. $M \not\models \varphi_i$. We know $A_M(\varphi_i) \subseteq A_M(\varphi)$ which by the induction hypothesis implies $M' \not\models \varphi_i$. By the semantics of φ we also have that $M' \not\models \varphi$.
- $\varphi = \varphi_1 \vee \varphi_2$: If $M \not\models \varphi$ then $M \not\models \varphi_1$ and $M \not\models \varphi_2$. We know $A_M(\varphi_1) \cup A_M(\varphi_2) \subseteq A_M(\varphi)$ which by the induction hypothesis implies $M' \not\models \varphi_1$ and $M' \not\models \varphi_2$. By the semantics of φ we also have that $M' \not\models \varphi$.
- $\varphi = \neg t$: If $M \not\models \varphi$ then $t \in \mathsf{En}(M)$. To disable t we have to fire at least one transition $t' \in ({}^\bullet t)^\bullet \cup {}^\bullet({}^\circ t)$ to either remove or add tokens that will disable or inhib t, respectively. Since $({}^\bullet t)^\bullet \cup {}^\bullet({}^\circ t) \subseteq A_M(\varphi)$ we have $M' \not\models \varphi$.

The remaining cases and negation cases are analogous. □

C Proof of Theorem 2

Proof. We shall argue that any reduction St satisfying the conditions of the theorem has the properties $\mathcal{Z}$, $\mathcal{D}$, $\mathcal{R}$, and $\mathcal{W}$.

- ($\mathcal{Z}$): Follows from Condition 1.
- ($\mathcal{R}$): Follows from Lemma 2 and Condition 2.
- ($\mathcal{D}$): Let $M \in \mathcal{M}(N)$ be a marking and $w \in \overline{\mathsf{St}(M)}^*$ 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: $T_{urg} \cap \mathsf{En}(M) \neq \emptyset$ or there is $p \in P$ where $I(p) = [a, b]$ and $b \in M(p)$.
  In the first case (Figure 2a), there is $t \in T_{urg} \cap \mathsf{En}(M) \cap \mathsf{St}(M)$ and ${}^\bullet({}^\circ t) \subseteq \mathsf{St}(M)$ 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 5a we know for all $t' \in p^\bullet$ that $t' \in \mathsf{St}(M)$ 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 $w \in \overline{\mathsf{St}(M)}^*$. Due to ${}^\bullet({}^\circ t) \subseteq \mathsf{St}(M)$ in Condition 3a w cannot add any tokens to p' since $w \in \overline{\mathsf{St}(M)}^*$ and we have that $|M'(p')| < w((p',t))$. This implies $\mathsf{zt}(M')$.
  In the second case (Figure 2b), there is $p \in P$ where $I(p) = [a, b]$ and $b \in M(p)$. Due to Condition 3b for all $t \in p^\bullet$ where $b \in g((p,t))$ we have that $t \in \mathsf{St}(M)$. Therefore t can never occur in w since $w \in \overline{\mathsf{St}(M)}^*$ and we must have that $b \in M'(p)$, implying that $\mathsf{zt}(M')$.
- ($\mathcal{W}$): Let $M, M' \in \mathcal{M}(N)$ be markings, $t \in \mathsf{St}(M)$, and $w \in \overline{\mathsf{St}(M)}^*$. 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 $t \notin \mathsf{En}(M)$. 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 $p \in {}^\circ t$ such that $|M(p)| \geq w((p,t))$. In the first case (Figure 3a), due to Condition 4a all the transitions that can add tokens that are in the guard $g((p,t))$ to p are included in $\mathsf{St}(M)$. Since $w \in \overline{\mathsf{St}(M)}^*$ this implies that $|\{x \in M_w(p) \mid x \in g((p,t))\}| < w((p,t))$

and $t \notin \mathsf{En}(M_w)$ contradicting our assumption that $M_w \xrightarrow{t} M'$. In the second case (Figure 3b), due to Condition 4b all the transitions that can remove at least one token from p are included in $\mathsf{St}(M)$. Since $w \in \overline{\mathsf{St}(M)}^*$ this implies that $|M_w(p)| \geq w((p,t))$ and $t \notin \mathsf{En}(M_w)$, again contradicting our assumption that $M_w \xrightarrow{t} M'$. Therefore we must have that $t \in \mathsf{En}(M)$.

Since $t \in \mathsf{En}(M)$ 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 $p \in t^\bullet \cap {}^\circ t'$ (both in Figure 4a). In the first case, due to Condition 5a all the transitions that can remove tokens that are in the guard $g((p,t))$ from p are included in $\mathsf{St}(M)$, implying that $t' \in \mathsf{St}(M)$. Since $w \in \overline{\mathsf{St}(M)}^*$ such a t' cannot exist. In the second case, due to Condition 5b we know that $(t^\bullet)^\circ \subseteq \mathsf{St}(M)$, implying that $t' \in \mathsf{St}(M)$. Since $w \in \overline{\mathsf{St}(M)}^*$ such a t' cannot exist. Therefore we must have that $M_t \xrightarrow{w} M'$ and can conlude with $M \xrightarrow{tw} M'$ as requested. We note here that the execution sequences wt and tw must lead to the same marking as the order of transition firings does not impact the reachable marking.

This completes the proof of the theorem. □

D Proof of Theorem 3

Proof. Termination: If $\neg\mathsf{zt}(M)$ then we terminate in line 2. If $A_M(\varphi) = \emptyset$ or no transitions are added to Y in line 5-8 or in line 11-13 then $Y = \emptyset$, we never enter the while-loop, and we terminate. Otherwise $Y \neq \emptyset$ and we enter the while-loop. Notice that $X \cap Y = \emptyset$ is always the case in the execution of Algorithm 1. We never remove transitions from X after they have been added. Therefore, since in line 35 a new transition is added to X at the end of each loop iteration, the loop can iterate at most once for each transition. Since T is finite by the TAPN definition, the loop iterates a finite number of times, and we terminate.

Correctness: It was shown that the construction in Theorem 2 satisfies $\mathcal{Z}$, $\mathcal{D}$, $\mathcal{R}$, and $\mathcal{W}$. It is therefore sufficient to show that Algorithm 1 replicates the construction. Notice that every transition that is added to Y is eventually added to X in line 35 and returned in line 36. Let $t \in Y$.

Condition 1: If $\neg\mathsf{zt}(M)$ then we return $\mathsf{En}(M)$ in line 2.

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

Condition 3a: In line 5 we pick any $t \in \mathsf{En}(M) \cap T_{urg}$, and in line 7 we have $t \in Y$ and in line 8 we have ${}^\bullet({}^\circ t) \subseteq Y$.

Condition 3b: In line 10 we pick a p where there exist $b \in M(p)$ s.t. $I(p) = [a,b]$, and in line line 11-13 we add all the $t \in p^\bullet$ where $b \in g((p,t))$ to Y.

Condition 4a: In line 17 and 18 we pick a $p \in {}^\bullet t$ s.t. $|\{x \in M(p) \mid x \in g((p,t))\}| < w((p,t))$. In line 19 through 22 we iterate through the relevant

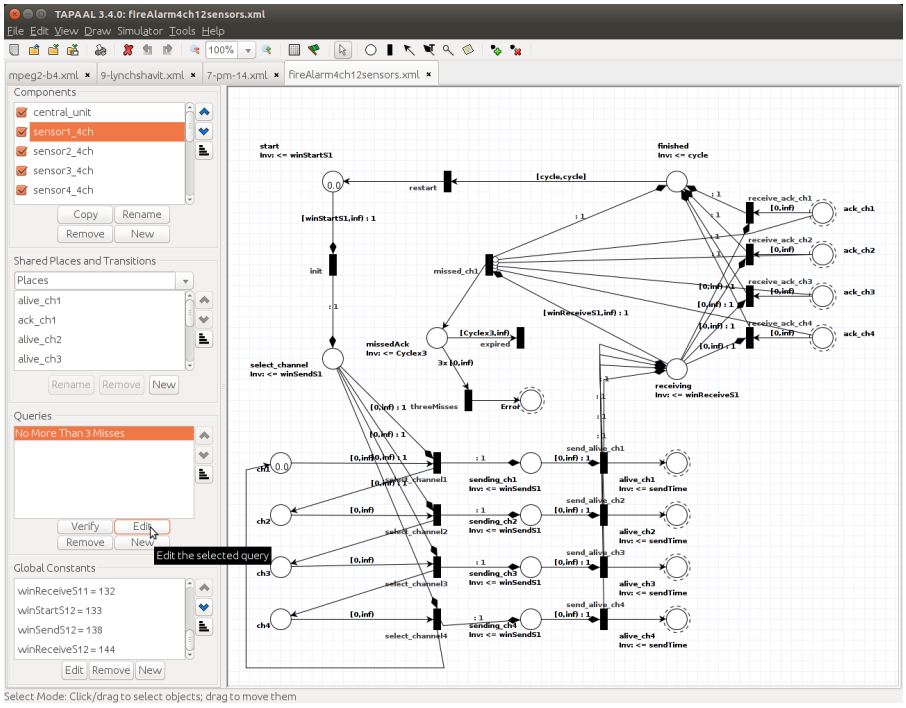


Fig. 5: The TAPAAL GUI.

transport arcs and add them to Y. We check for $0 \in g((p,t))$ in line 23 and add ${}^\bullet p$ to Y in line 24.

Condition 4b: In line 26 we pick any $p \in {}^\circ$ s.t. $|M(p)| \geq w((p,t))$. In line 27 through 29 we iterate over all $t' \in p^\bullet$ not already in X, and if t' is able to remove a token from p it is added to Y.

Condition 5a: In line 31 and 32 for all $p \in {}^\bullet t$ and all $t' \in p^\bullet$ we add t' to Y if $t' \notin X$ and $g((p,t)) \cap g((p,t')) \neq \emptyset$.

Condition 5b: In line 33 we add the transitions of $(t^\bullet)^\circ$ that are not already in X to Y. □

E User Manual for Partial Order Reduction in TAPAAL

The newest release 3.4.0 of the model checker TAPAAL includes the implementation of discrete-time partial order reduction for timed-arc Petri nets. It is available as precompiled binaries for Windows, Mac, and Linux operating systems at: http://www.tapaal.net/download. It requires Java run-time environment installed on the machine. At the bottom of the download page, one can obtain also the zip file `partial-order-experiments.zip` with all .xml model files for the case studies described in Section 5.

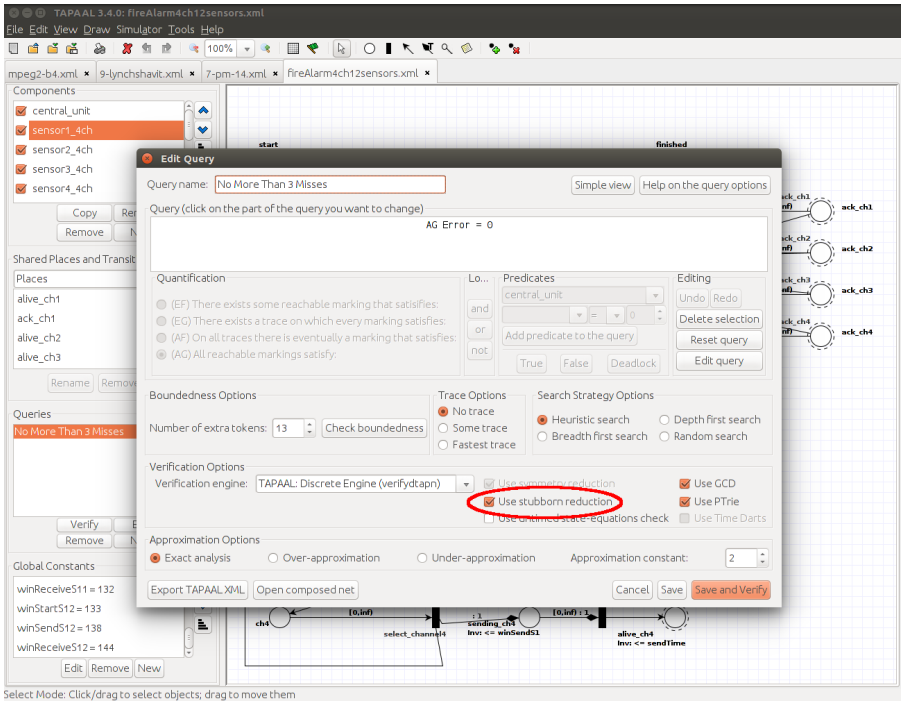
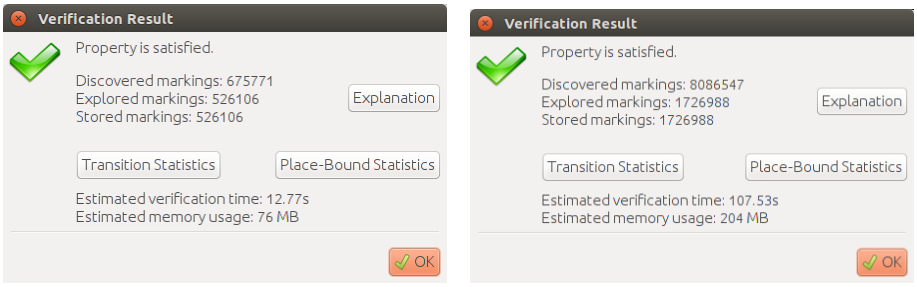


Fig. 6: Enabling and disabling partial order reduction

In Figure 5 we display a screenshot of TAPAAL 3.4.0 graphical user interface. To open a file, such as the FireAlarm-12 case study as seen in Figure 5, select the File dropdown menu, followed by the Open option, and then navigate to the appropriate directory and open the .xml model file. In the main work space in Figure 5, we see the sensor1_4ch compoment of the FireAlarm-12 model. The net can be modified using the toolbox menu at the top of the screen below the window options. Manual simulation mode can be enabled by selecting the green flag in the toolbox menu. Verification can be done by adding, or editing an existing query, in the Queries section of the interface, where the edit option is highlighted by the cursor in Figure 5. Selecting edit will open the query configuration window for the query "No More Than 3 Misses" as seen in Figure 6.

In this window it is possible to edit queries, such as choosing or changing a quantification, adding logical connectives such as conjunction, and predicates seen in the upper half of Figure 6, and selecting various options to customize the verification of the query seen in the lower half of Figure 6, such as search strategy, traces, etc. Make sure to select the "Advanced View" in order to see the verification options. Now various verification optimization techniques can be utilized to assist the verification engine, such as symmetry reductions and PTries. Among these, there is the discrete partial order reduction implementation which in Figure 6 corresponds to the "Use stubborn reduction" checkbox



(a) Partial order reduction enabled (b) Partial order reduction disabled

Fig. 7: Verification result dialog

option highlighted with an ellipse. It is currently enabled in the displayed query dialog. Selecting the "Save and Verify" option saves any changes made to the custimization of the query and begins the verification of the query using the verification engine. The output of the verifcation can be seen in Figure 7.

Figure 7 shows the output of verifying the query both with and without enabling the partial order reduction, seen in Figure 7a and Figure 7b, respectively. Both cases agree that the property described by the query is satisfied, and show some statistics of the verification. A total of 526106 markings was explored to verify the property with partial order reduction enabled, and a total of 1726988 markings was explored to verify it with partial order reduction disabled. Both correspond to the numbers seen in Table 3 for the FireAlarm-12 case study. The verification time is not the same due to being run on different machines.