Stubborn versus structural reductions for Petri nets

Bønneland, Frederik M.; Dyhr, Jakob; Jensen, Peter G.; Johannsen, Mads; Srba, Jiří

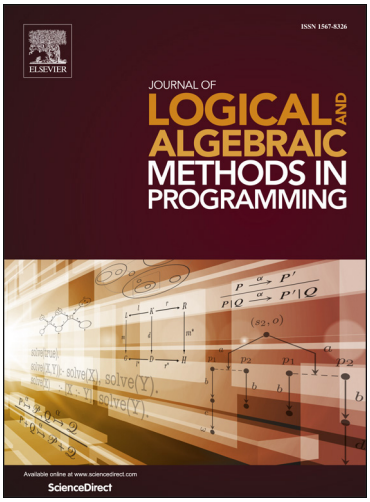
Published in:
Journal of Logic and Algebraic Programming

DOI (link to publication from Publisher):
10.1016/j.jlamp.2018.09.002

Publication date:
2019

Document Version
Accepted author manuscript, peer reviewed version

Accepted Manuscript



Stubborn Versus Structural Reductions for Petri Nets

Frederik M. Bønneland, Jakob Dyhr, Peter G. Jensen, Mads Johannsen, Jiří Srba

PII: S2352-2208(18)30035-X
DOI: https://doi.org/10.1016/j.jlamp.2018.09.002
Reference: JLAMP 411

To appear in: *Journal of Logical and Algebraic Methods in Programming*

Received date: 12 March 2018
Revised date: 6 July 2018
Accepted date: 7 September 2018

Please cite this article in press as: F.M. Bønneland et al., Stubborn Versus Structural Reductions for Petri Nets, *J. Log. Algebraic Methods Program.* (2018), https://doi.org/10.1016/j.jlamp.2018.09.002

Highlights

- Stubborn reductions are extendable to Petri nets with weighted inhibitor arcs.
- Structural reduction rules refined for weighted arcs to increase applicability.
- Combination of reduction techniques does not overlap.
- TAPAAL surpassing LoLA on answering reachability queries.

Stubborn Versus Structural Reductions for Petri Nets

Frederik M. Bønneland[a,*], Jakob Dyhr[a], Peter G. Jensen[a,*], Mads Johannsen[a], Jiří Srba[a,*]

[a]*Department of Computer Science, Aalborg University, Selma Lagerlöfs Vej 300, DK-9220 Aalborg East, Denmark*

Abstract

Partial order and structural reduction techniques are some of the most beneficial methods for state space reduction in reachability analysis of Petri nets. This is among others documented by the fact that these techniques are used by the leading tools in the annual Model Checking Contest (MCC) of Petri net tools. We suggest improved versions of a partial order reduction based on stubborn sets and of a structural reduction with additional new reduction rules, and we extend both methods for the application on Petri nets with weighted arcs and weighted inhibitor arcs. All algorithms are implemented in the open-source verification tool TAPAAL and evaluated on a large benchmark of Petri net models from MCC'17, including a comparison with the tool LoLA (the last year winner of the competition). The experiments document that both methods provide significant state space reductions and, even more importantly, that their combination is indeed beneficial as a further nontrivial state space reduction can be achieved.

Keywords: partial order reduction, stubborn sets, structural reductions, reachability analysis, Petri nets

1. Introduction

Model checking of large distributed and concurrent systems is often limited in its applicability due to the state space explosion problem. Components in concurrent systems may independently perform actions without being in conflict with other components, forcing an explicit state space analysis to explore every possible interleaving of the actions and hence creating an explosion in the number of executable action sequences. Petri nets are a popular formalism for modelling of concurrent systems [12], however, due to the state space explosion problem, essentially all interesting questions about their behaviour, including the reachability and coverability problems, are EXPSPACE-hard (see e.g. [4]).

*Corresponding author

Email addresses: frederikb@cs.aau.dk (Frederik M. Bønneland), pgj@cs.aau.dk (Peter G. Jensen), srba@cs.aau.dk (Jiří Srba)

Despite the discouraging complexity results, numerous techniques have been developed to improve the feasibility of reachability analysis, including methods based on reducing the state space by eliminating the interleaving in independent components (see e.g. [5, 1]). The focus of our work is on two such techniques: structural reductions [11] and stubborn set reductions [15], both applied to and evaluated on the model of weighted Petri nets with inhibitor arcs. *Structural reductions* preprocess the Petri net model by collapsing redundant places and transitions, while preserving the validity of the model checking question. The idea is that a smaller number of places and transitions in a net can help to reduce the degree of concurrency and eliminate some unnecessary interleavings. In partial order reductions, like e.g. *stubborn set reduction*, we identify transitions that are independent of each other and the order of their execution does not influence the model checking property in question. This can be considered as another method that can, in an on-the-fly manner, reduce the number of possible interleavings of independent actions. Both structural and stubborn reductions can be in a straightforward way combined, however, to the best of our knowledge, the effect of this combination has not previously been studied in detail.

We perform a comparative study of the effects of the two types of state space reduction techniques and their combination. For our experiments, we use the database of nets and reachability queries from the annual Model Checking Contest (MCC) [8] and conclude that while both techniques are clearly beneficial for the performance of the reachability analysis, the combination of the two methods demonstrates yet another degree of performance improvements. Apart from this experimental evaluation, we make several technical contributions to stubborn and structural reductions applied to the model of Petri nets. Both techniques are extended to work for the reachability logic used in MCC, while allowing us to use weighted arcs as well as weighted inhibitor arcs. In particular, the stubborn set reduction as well as the structural reduction were refined to take weighted arcs into account, in order to minimize the size of the state space that is necessary to explore for a given reachability query. In stubborn reduction, we refine the computation of dependencies between transitions so that instead of the traditional comparison of presets and postsets of places, we utilize a more detailed analysis of the increasing/decreasing effect of a transition on a given place. All techniques are proved correct and implemented in the model checker TAPAAL [3]. The experiments are encouraging as the improved techniques in their combination allow us to solve more reachability queries from MCC'17 [8] than the model checker LoLA [21], the last year winner in the reachability category.

Related work. The stubborn reduction technique is related to and based upon the seminal work on stubborn sets by Valmari et al. [18, 13, 9, 15, 16]. This includes write up/down sets [15], the closure procedure [9], and attractor sets [13]. We contribute by adding support for inhibitor arcs, extending the technique to a reachability logic used in MCC and presenting a different formulation of stubborn sets for reachability in the general setting of labelled transition systems.

Further analysis during the generation of stubborn sets can help to generate more optimal (smaller) stubborn sets, which can be done e.g. by extracting terminal strongly connected components from the derived transition dependency graph [18]. We choose to use instead heuristic methods for the generation of stubborn sets as they have smaller computational overhead and achieve better performance in our experiments.

Structural reductions of Petri nets were studied by Murata et al. [11, 10] with the main focus on preserving liveness, safety, and boundedness. The reduction rules were recently extended to include weighted nets with inhibitor arcs while preserving the reachability of cardinality queries [6]. We contribute by increasing the applicability of the four rules presented in [6] and refining them for the use with weighted arcs so that a more significant net reduction can be achieved compared to [6]. Moreover, we introduce five new reduction rules, allowing us to reduce the size of the input net even further.

Stubborn sets are also an important state space reduction technique used in the tool LoLA [21] that we compare against to in our experiments. Their stubborn set implementation have several approaches to reachability analysis, utilising up/down sets and terminal strongly connected components [9] to mention some. Approaches using terminal strongly connected components can present some performance problems due to concurrent cycles of invisible (or non-interesting) transitions, forcing the method to sometimes explore the full parallel composition [17, 19]. Remedies to this have been explored in the form of frozen actions [17], removing transitions from consideration if they are tagged as frozen. Besides LoLA's take on stubborn sets [13], their tool includes several other reduction and verification improvements such as symmetry reduction [14] and Counter Example Guided Abstraction Refinement (CEGAR) [20], however, LoLA does not employ structural reductions. Our experiments document that the refined and combined application of our stubborn and structural reduction techniques becomes competitive in performance compared with the tool LoLA.

2. Preliminaries

A *labelled transition system* (LTS) is a tuple $TS = (\mathcal{S}, A, \rightarrow)$ where $\mathcal{S}$ is a set of states, A is a set of actions (or labels), and $\rightarrow \subseteq \mathcal{S} \times A \times \mathcal{S}$ is a transition relation. We write $s \xrightarrow{a} s'$ whenever $(s, a, s') \in \rightarrow$ and say that a is *enabled* in s. The set of all enabled actions in a state s is denoted $en(s)$. A state s is a *deadlock* if $en(s) = \emptyset$. We write $s \rightarrow s'$ whenever there is an action a such that $s \xrightarrow{a} s'$. We inductively extend the relation $\xrightarrow{a}$ to sequences of transitions $w \in A^*$ such that $s \xrightarrow{\epsilon} s$ and $s \xrightarrow{wa} s'$ if $s \xrightarrow{w} s''$ and $s'' \xrightarrow{a} s'$. We write $s \rightarrow^n s'$ if there is $w \in T^*$ of length n such that $s \xrightarrow{w} s'$, and we write $s \rightarrow^* s'$ if $s \rightarrow^n s'$ for some $n \geq 0$.

The *reachability problem* is, given an LTS $TS = (\mathcal{S}, A, \rightarrow)$, an initial state $s \in \mathcal{S}$, and a set of goal states $G \subseteq \mathcal{S}$, to decide whether there is $s' \in G$ s.t. $s \rightarrow^* s'$.

2.1. Petri Nets

Let $\mathbb{N}^0 = \mathbb{N} \cup \{0\}$ be the set of natural numbers including 0. Let $\mathbb{N}^\infty = \mathbb{N} \cup \{\infty\}$ be the set of natural numbers including infinity.

Definition 1 (Petri Net with Inhibitor Arcs). *A Petri net is a tuple $N = (P, T, W, I)$ where P and T are finite and disjoint sets of places and transitions, W: $(P \times T) \cup (T \times P) \to \mathbb{N}^0$ is a weight function for regular arcs, and I : $(P \times T) \to \mathbb{N}^\infty$ is a weight function for inhibitor arcs.*

A *marking* M on N is a function $M : P \to \mathbb{N}^0$, where $M(p)$ denotes the number of tokens in place p. The set of all markings of a Petri net N is written as $\mathcal{M}(N)$. Let $M_0 \in \mathcal{M}(N)$ be a given *initial marking* of N.

A Petri net $N = (P, T, W, I)$ defines an LTS $TS(N) = (\mathcal{S}, A, \to)$ where $\mathcal{S} = \mathcal{M}(N)$ is the set of all markings, $A = T$ is the set of labels, and $M \xrightarrow{t} M'$ whenever for all $p \in P$ we have $M(p) < I((p,t))$ and $M(p) \geq W((p,t))$ such that $M'(p) = M(p) - W((p,t)) + W((t,p))$.

Example 1. An example of a Petri net is given in Figure 1a. We use the standard notation and denote places by circles, transitions by squares and the dots represent tokens in the initial marking. The weights of all arcs are implicitly fixed to 1, the only exception being the arc from p_3 to t_3 that requires two tokens in order to fire t_3. The arrow from p_5 to t_3 with cirle head-tip is an inhibitor arc (again with the default weight 1) and it inhibits the enabledness of t_3 as soon as p_5 contains at least one token. The labelled transition system (containing only markings reachable from the initial one) is depicted in Figure 1b. Here the notation e.g. $2p_1p_4$ represents the initial marking with two tokens in p_1 and one token in p_4.

The net contains lots of interleavings and the markings in dashed boxes are those that can be disregarded once we apply our stubborn set reduction for verifying the reachability of a marking with at least two tokens in the place p_3. In Figure 1c we display a reduced net where the place p_1 and transition t_1 are removed. This structural reduction preserves the reachability of the goal marking and we can see that the reachable state space gets significantly reduced as demonstrated in Figure 1d. An application of a stubborn set reduction on top of the structural reduction allows for an even greater state space reduction, as showed by the dashed boxes which can be removed by stubborn set reduction during the state space search of the reduced net. The details of these methods are explained in the remainder of this paper.

Let us first fix some useful notation. For a place or transition x, we denote the *preset* of x as ${}^\bullet x = \{y \mid W((y,x)) > 0\}$, and the *postset* of x as $x^\bullet = \{y \mid W((x,y)) > 0\}$. For a place p we define the *increasing preset* of p, containing all transitions that increase the number of tokens in p, as ${}^+p = \{t \in {}^\bullet p \mid W((t,p)) > W((p,t))\}$, and similarly the *decreasing postset* of p as $p^- = \{t \in p^\bullet \mid W((t,p)) < W((p,t))\}$. For a transition t, we denote the *inhibitor preset* of t as ${}^\circ t = \{p \mid I((p,t)) < \infty\}$, and for a place p, we denote the *inhibitor postset* of p as $p^\circ = \{t \mid I((p,t)) < \infty\}$. For a set X of either places or transitions, we

extend the notation as ${}^\bullet X = \bigcup_{x \in X} {}^\bullet x$ and $X^\bullet = \bigcup_{x \in X} x^\bullet$, and similarly for the other operators.

In order to syntactically define the set of goal states G for the reachability problem on Petri nets, we use the *reachability logic* from the Model Checking Contest [8]. The syntax of the logic is as follows:

$$\varphi ::= \mathit{true} \mid \mathit{false} \mid \alpha \mid \mathit{deadlock} \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi$$

$$\alpha ::= t \mid e_1 \bowtie e_2$$

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

where $t \in T$, $c \in \mathbb{N}^0$, $\bowtie \in \{<, \leq, =, \neq, >, \geq\}$, $p \in P$, and $\oplus \in \{+, -, \cdot\}$. The evaluation of an arithmetical expression e in a marking M is defined as $\mathit{eval}_M(c) = c$, $\mathit{eval}_M(p) = M(p)$ and $\mathit{eval}_M(e_1 \oplus e_2) = \mathit{eval}_M(e_1) \oplus \mathit{eval}_M(e_2)$. The semantics of a reachability formula φ in a marking M is given in Figure 2.

Formulae that do not use any atomic predicate t for transition firing and no predicate *deadlock* are called *cardinality formulae* and formulae that avoid the use of $e_1 \bowtie e_2$ and *deadlock* are called *fireability formulae*. The formula *deadlock* is called the *deadlock formula*.

3. Stubborn Reduction for Weighted Petri Nets with Inhibitor Arcs

We shall now introduce a general idea of a reduction on an LTS that reduces the size of the state space and later instantiate it to the case of Petri nets with inhibitor arcs. A reduction can be seen as a filter that, for each state, specifies a subset of actions that are sufficient to explore in order to reach a state satisfying a given reachability formula.

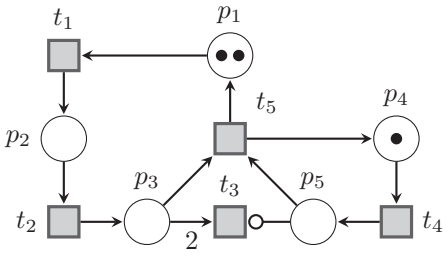
Definition 2 (Reduced Transition Relation). *Let $TS = (\mathcal{S}, A, \to)$ be an LTS. A* reduction *of the transition system TS is a function $St : \mathcal{S} \to 2^A$. A* reduced transition relation *is a relation $\xrightarrow[St]{} \subseteq \to$ such that $s \xrightarrow[St]{a} s'$ iff $s \xrightarrow{a} s'$ and $a \in St(s)$.*

Let $TS = (\mathcal{S}, A, \to)$ be an LTS, $s \in \mathcal{S}$ a state, and St a reduction of TS. Let $\overline{St(s)} = A \setminus St(s)$ be the set of all actions not in $St(s)$. We define the following property **W** of a reduction that guarantees that any action in $St(s)$ commutes with respect to actions from $\overline{St(s)}$.

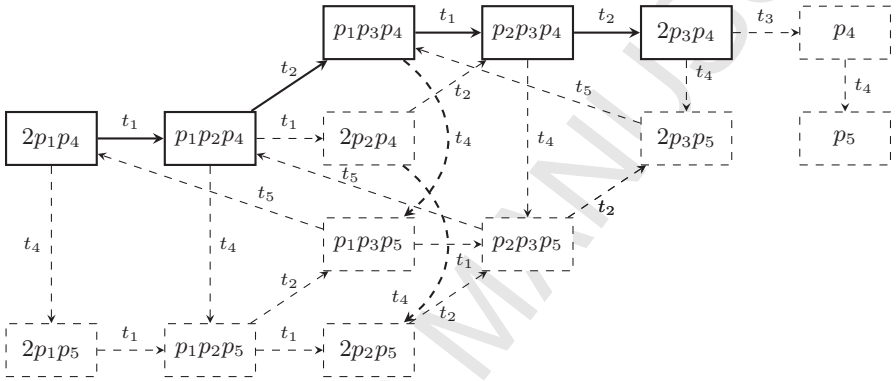
W For all $s \in \mathcal{S}$, all $a \in St(s)$, and all $w \in \overline{St(s)}^*$, if $s \xrightarrow{wa} s'$ then $s \xrightarrow{aw} s'$.

Reductions that satisfy **W** are called *(weak) semistubborn* reductions [15]. In the rest of the paper, we say that $St(s)$ is the *stubborn set* of s and that an action $a \in St(s)$ is a *stubborn action* in s.

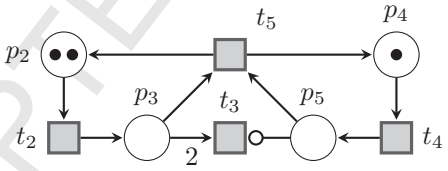
Let $TS = (\mathcal{S}, A, \to)$ be an LTS and $G \subseteq \mathcal{S}$ a set of goal states. For a reduction St to preserve the reachability of a goal state, we define the following sufficient condition on St.



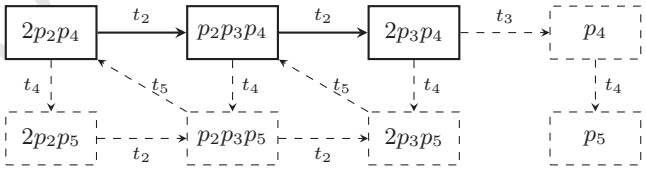
(a) A Petri net (the arc from p_5 to t_3 is the inhibitor arc and the weight of the arc from p_3 to t_3 is 2; all other arcs have the default weight 1)



(b) Reachable state space for the net from Figure 1a (the dashed boxes are removed by a possible stubborn set reduction preserving the reachability of $\varphi = p_3 \geq 2$)



(c) Petri Net from Figure 1a after the application of structural reductions which removed the place p_1 and transition t_1 while preserving the property $\varphi = p_3 \geq 2$



(d) Reachable state space for the net from Figure 1c (the dashed boxes are removed by a possible stubborn set reduction preserving the reachability of $\varphi = p_3 \geq 2$)

Figure 1: Combination of stubborn and structural reductions

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

Figure 2: Semantics of formulae

R For all $s \in \mathcal{S}$, if $s \notin G$ and $s \xrightarrow{w} s'$ where $w \in \overline{St(s)}^*$ then $s' \notin G$.

Condition **R** states that if we start in a non-goal state, the execution of non-stubborn transitions cannot reach any goal state in G. Hence it ensures that at least one stubborn action must be executed in order to reach a goal state. Any reduction that satisfies Conditions **W** and **R** also guarantees the preservation of reachability as stated by the following theorem.

Theorem 1 (Reachability Preservation). *Let $TS = (\mathcal{S}, A, \rightarrow)$ be an LTS and let $G \subseteq S$ be a set of goal states. Let St be a reduction of TS satisfying **W** and **R** and let $s \in \mathcal{S}$. If $s \rightarrow^n s'$ for some $s' \in G$ then $s \xrightarrow[St]{}^m s''$ for some $s'' \in G$ where $m \leq n$.*

Proof. Let $w \in A^*$ be a transition sequence such that $s \xrightarrow{w} s'$ for some $s' \in G$. The proof proceeds by induction on the length of w. *Base case:* If $|w| = 0$ then $s = s' \in G$ and the claim trivially holds.
Inductive case: Let $|w| > 0$. If $s \in G$ then the claim immediately holds as in the base case. If $s \notin G$ then by **R** we then get that at least one transition in w must belong to $St(s)$, otherwise it is impossible that $s' \in G$. Hence we can divide w into vau where $v \in \overline{St(s)}^*$ and $a \in St(s)$. Condition **W** now implies the existence of s_a such that $s \xrightarrow{a} s_a \xrightarrow{vu} s'$. If $s_a \in G$, the length of the path from s_0 to s_a is less than or equal to $|w|$ and we are done. Otherwise, by the induction hypothesis we get that $s_a \xrightarrow[St]{}^m s''$ for some $s'' \in G$ where $m \leq |vu|$. This together with $s \xrightarrow[St]{a} s_a$ gives that $s \xrightarrow[St]{}^{m+1} s''$ where $s'' \in G$ such that $m + 1 \leq |w|$ as requested. □

In the following subsection, we shall instantiate this general framework to the case of Petri nets, taking a particular care to account for weights on arcs in order to construct the smallest possible stubborn sets.

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

Table 2: Increasing and decreasing transitions of expression e

3.1. Stubborn Set Reduction of Petri Nets

Let $N = (P, T, W, I)$ be a fixed Petri net and φ a reachability formula. We are interested in the question, whether we can reach from the initial marking some of the goal markings from $G_\varphi = \{M \in \mathcal{M}(N) \mid M \models \varphi\}$. We first define the notion of *interesting transitions* $A_M(\varphi) \subseteq T$ for a marking M relative to φ such that whenever $M \xrightarrow{w} M'$ via the 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 the definition of $A_M(\varphi)$. 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, where E_N is the set of all arithmetic expressions that can be constructed for the net N. These functions, given an expression e, return all transitions that can possibly increase resp. decrease the evaluation of e. Formally, the required properties of the functions $incr_M(e)$ and $decr_M(e)$ are summarized in the next lemma.

Lemma 2. *Let e be an arithmetic expression, let $M, M' \in \mathcal{M}(N)$, and let $w = t_1 t_2 \dots t_n \in T^*$ be such that $M \xrightarrow{w} M'$.*

- *If $eval_M(e) < eval_{M'}(e)$ then there is i, $1 \leq i \leq n$, s.t. $t_i \in incr_M(e)$.*

- *If* $eval_M(e) > eval_{M'}(e)$ *then there is* i, $1 \leq i \leq n$, *s.t.* $t_i \in decr_M(e)$.

Proof. The proof follows from the definition of the functions by a straightforward structural induction on e. □

Let us by $\overline{A_M(\varphi)}$ denote the set $T \setminus A_M(\varphi)$ of non-interesting transitions. We can now formulate a lemma stating that at least one interesting transition must be executed before we can reach a goal marking.

Lemma 3. *Let* $N = (P, T, W, I)$ *be a Petri net,* $M \in \mathcal{M}(N)$ *a marking,* φ *a reachability formula, and* $w \in \overline{A_M(\varphi)}^*$ *a sequence of non-interesting transitions. If* $M \not\models \varphi$ *and* $M \xrightarrow{w} M'$ *then* $M' \not\models \varphi$.

Proof. Let $N = (P, T, W, I)$ be a Petri net, $M \in \mathcal{M}(N)$ be a marking, and φ 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 a $t \in en(M)$. To disable t we have to fire a transition $t' \in ({}^\bullet t)^- \cup {}^+({}^\circ t)$ to either remove or add tokens that will disable or inhibit t, respectively. Since $({}^\bullet t)^- \cup {}^+({}^\circ t) \subseteq A_M(\varphi)$ we have $M' \not\models \varphi$ as the selected transition t is still enabled.

- $\varphi = t$: If $M \not\models \varphi$ then $t \notin en(M)$. Either there exists $p \in {}^\bullet t$ such that $M(p) < W(p, t)$ and we have to fire some transition from ${}^+p$ to enable t, or there exists $p \in {}^\circ t$ such that $M(p) \geq I(p, t)$ and we have to fire some transition from p^- to enable t. Since ${}^+p \subseteq A_M(\varphi)$ or $p^- \subseteq A_M(\varphi)$, we get that $M' \not\models \varphi$.

- $\varphi = e_1 < e_2$: 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 that $eval_M(e_1) \leq eval_{M'}(e_1)$ as well as $eval_M(e_2) \geq eval_{M'}(e_2)$ because of Lemma 2, 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 that $eval_M(e_1) \geq eval_{M'}(e_1)$ as well as $eval_M(e_2) \leq eval_{M'}(e_2)$ because of Lemma 2, 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 that $A_M(\varphi_i) \subseteq A_M(\varphi)$ which by the induction hypothesis implies $M' \not\models \varphi_i$. By the semantics of conjunction 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 that $A_M(\varphi_1) \cup A_M(\varphi_2) \subseteq A_M(\varphi)$ which by the induction hypothesis implies $M' \not\models \varphi_1$ and $M' \not\models \varphi_2$. By the semantics of disjunction we also have that $M' \not\models \varphi$.

- $\varphi = \neg t$: If $M \not\models \varphi$ then $t \in en(M)$. To disable t we have to fire at least one transition from $({}^\bullet t)^- \cup {}^+({}^\circ t)$ to either remove or add tokens that will disable or inhibit t, respectively. Since $({}^\bullet t)^- \cup {}^+({}^\circ t) \subseteq A_M(\varphi)$ we have $M' \not\models \varphi$.

The remaining cases and negation cases are analogous. □

Lemma 3 allows us to satisfy Property **R** of Theorem 1 by including all interesting transitions in the stubborn set. Ensuring Property **W** is achieved by including further transitions according to the following theorem.

Theorem 4 (Reachability Preserving Closure). *Let $N = (P, T, W, I)$ be a Petri net, φ a formula, and St a reduction of $TS(N)$ such that for all $M \in \mathcal{M}(N)$ the following conditions hold.*

1 *We have $A_M(\varphi) \subseteq St(M)$.*

2 *For all $t \in St(M)$, if $t \notin en(M)$ then*

- *there is $p \in {}^\bullet t$ s.t. $M(p) < W(p,t)$ and ${}^+p \subseteq St(M)$, or*
- *there is $p \in t^\circ$ s.t. $M(p) \geq I(p,t)$ and $p^- \subseteq St(M)$.*

3 *For all $t \in St(M)$, if $t \in en(M)$ then*

- *for all $p \in {}^\bullet t$ where $t \in p^-$ we have $p^\bullet \subseteq St(M)$, and*
- *for all $p \in t^\bullet$ where $t \in {}^+p$ we have $p^\circ \subseteq St(M)$.*

*Then St satisfies **W** and **R**.*

Proof. From Condition 1 we know that $A_M(\varphi) \subseteq St(M)$ and hence Property **R** holds for St by Lemma 3. We will now argue that St satisfies Property **W**. Let $M \in \mathcal{M}(N)$ be a marking, $t \in T$ a transition such that $t \in St(M)$, and $w \in \overline{St(M)}^*$ a transition sequence of non-stubborn transitions. We want to show that if $M \xrightarrow{wt} M'$ then also $M \xrightarrow{tw} M'$.

Let $M_w \in \mathcal{M}(N)$ be a marking such that $M \xrightarrow{w} M_w$. Let us assume for the sake of contradiction that $t \notin en(M)$. Then either (i) there exists $p \in {}^\bullet t$ such that $M(p) < W(p,t)$ or (ii) there is $p \in t^\circ$ where $M(p) \geq I(p,t)$. In case (i) we get by Condition 2 that all transitions that can increase the number of tokens in p are stubborn. Since $w \in \overline{St(M)}^*$ this implies that $M_w(p) < W(p,t)$ and $t \notin en(M_w)$, contradicting our assumption that $M_w \xrightarrow{t} M'$. In case (ii) we get by Condition 2 that all transitions that can decrease the number of tokens in p are stubborn. Since $w \in \overline{St(M)}^*$ this implies that also $M_w(p) \geq I(p,t)$ and $t \notin en(M_w)$, again contradicting our assumption that $M_w \xrightarrow{t} M'$. Therefore we can conclude that $t \in en(M)$.

Since $t \in en(M)$ there is $M_t \in \mathcal{M}(N)$ such that $M \xrightarrow{t} M_t$. We also have to show that $M_t \xrightarrow{w} M'$. For the sake of contradiction, assume that this is not the case. Then there must exist a transition t' that occurs in w and that became disabled because t was fired before the sequence w. There are two cases how this can happen: (i) either t decreased the number of tokens in a shared pre-place $p \in {}^\bullet t \cap {}^\bullet t'$ (ii) or t increased the number of tokens in a place $p \in t^\bullet \cap {}^\circ t'$. In case (i), due to Condition 3, we know that for all $p \in {}^\bullet t$ if $t \in p^-$ then

Algorithm 1: Construction of a reachability preserving stubborn set

```
   input  : Net N = (P, T, W, I), M ∈ M(N) and a formula φ
   output : Stubborn set St(M) such that St satisfies W and R
1  X := ∅; unprocessed := A_M(φ);
2  while unprocessed ≠ ∅ do
3      pick any t ∈ unprocessed;
4      if t ∉ en(M) then
5          if ∃p ∈ •t such that M(p) < W(p,t) then
6              pick any p ∈ •t such that M(p) < W(p,t);
7              unprocessed := unprocessed ∪ (+p \ X);
8          else
9              pick any p ∈ °t such that M(p) ≥ I(p,t);
10             unprocessed := unprocessed ∪ (p⁻ \ X);
11     else
12         foreach p ∈ •t do
13             if t ∈ p⁻ then
14                 unprocessed := unprocessed ∪ (p• \ X);
15         foreach p ∈ t• do
16             if t ∈ +p then
17                 unprocessed := unprocessed ∪ (p° \ X);
18     unprocessed := unprocessed \ {t};
19     X := X ∪ {t};
20 return X;
```

$p^\bullet \subseteq St(M)$, implying that $t' \in St(M)$. Since $w \in \overline{St(M)}^*$ such a t' cannot exist in w. In case (ii), due to Condition 3, we know that for all $p \in t^\bullet$ if $t \in {}^+p$ then $p^\circ \subseteq St(M)$, implying that $t' \in St(M)$. Since $w \in \overline{St(M)}^*$ such a t' cannot exist in w either. Hence the sequence w is executable from M_t and because the firings of wt and tw from M both reach a unique marking, we conclude with $M \xrightarrow{tw} M'$ as requested. □

In Algorithm 1 we now provide, based on Theorem 4, a pseudocode for a construction of a reachability preserving stubborn set that satisfies **W** and **R** for a given marking M and a reachability formula φ.

Theorem 5. *Algorithm 1 terminates and computes a reduction St satisfying **W** and **R**.*

Proof. For the proof of termination, we first notice the while-loop invariant $X \cap unprocessed = \emptyset$. At the end of each iteration of the while-loop, we move one transition from *unprocessed* into X and this can happen only finitely many times (there are finitely many transitions) before the set *unprocessed* becomes empty

and the algorithm terminates. For the correctness argument, we notice that Algorithm 1 replicates exactly the closure operations described in Theorem 4 and guarantees that all conditions of Theorem 4 are met at the termination of the algorithm. □

Finally, we want to point out that there is nondeterminism in both generating the interesting set of transitions and applying the stubborn set closure. For the interesting set of transitions, this is whenever we resolve the deadlock or the fireability predicate, and in case of conjunction where none of the conjuncts hold in the given marking. For the stubborn set closure, there is a nondeterministic choice whenever we have to select either a disabling or inhibiting place for a disabled transition. As documented by practical examples, we often prefer to construct the smallest possible stubborn set in order to reduce the reachable state space (though this does not in general guarantee the smallest state space as demonstrated in [18]). In our implementation of Algorithm 1, we perform the nondeterministic choices such that they minimize the number of newly added transitions to the set *unprocessed*. In particular, for the nondeterminism in lines 6 and 9 of Algorithm 1, we experienced that selecting places such that their preset is already included in the closure greatly improves performance. In general, we only apply a heuristic approach to resolve the nondeterministic choices as our experiments showed that the computational overhead of finding a minimal stubborn set can be significant.

4. Structural Reductions for Weighted Petri Nets with Inhibitor Arcs

After refining the stubborn set reduction for the case of weighted Petri nets with inhibitor arcs, we shall now focus also on adapting the structural reduction techniques for this class of nets. In what follows, we extend the standard structural reduction rules as presented e.g. in [10] so that they allow to more efficiently reduce weighted nets (with inhibitor arcs). We also add five additional reduction rules in order to further reduce the size of the input net.

Let $N = (P, T, W, I)$ be a fixed Petri net. We shall first notice that all formulae involving the fireability predicate $t \in T$ can be rewritten into an equivalent cardinality formula

$$p_0 \geq W(p_0, t) \wedge \cdots \wedge p_n \geq W(p_n, t) \wedge p'_0 < I(p'_0, t) \wedge \cdots \wedge p'_m < I(p'_m, t)$$

where ${}^\bullet t = \{p_0, \ldots, p_n\}$ and $t^\circ = \{p'_0, \ldots, p'_m\}$. It is thus sufficient in the remainder of this section to consider only cardinality formulae. We will, at the end of this section, also discuss the correctness of the presented rules in relation to the deadlock formulae.

The rules presented in this section assume that $\mathit{places}(\varphi)$ denotes the set of

all places that occur in the cardinality formula φ such that

$$\begin{aligned} places(true) = places(false) = places(c) &= \emptyset \\ places(p) &= \{p\} \\ places(\neg\varphi) &= places(\varphi) \\ places(\varphi_1 \vee \varphi_2) = places(\varphi_1 \wedge \varphi_2) &= places(\varphi_1) \cup places(\varphi_2) \\ places(e_1 \bowtie e_2) = places(e_1 \oplus e_2) &= places(e_1) \cup places(e_2)\ . \end{aligned}$$

In each structural reduction rule, we fix some places and transitions that satisfy given preconditions and perform updates that can change the weight function and remove places and/or transitions (including all their connected arcs). Let us now give a general definition of correctness of a given rule X (where X is one of the rules A to I), stating that the reachability of a given cardinality formula φ is preserved.

Definition 3 (Correctness of Rule X). *Let $N = (P, T, W, I)$ be a Petri net and $M_0 \in \mathcal{M}(N)$ its initial marking. Let $N' = (P', T', W', I')$ and $M'_0 \in \mathcal{M}(N')$ be the modified N and M_0 after applying once Rule X for a cardinality formula φ. We say that Rule X is* correct *for a cardinality formula φ if there exists $M \in \mathcal{M}(N)$ s.t. $M_0 \rightarrow^* M$ and $M \models \varphi$ if and only if there exists $M' \in \mathcal{M}(N')$ s.t. $M'_0 \rightarrow^* M'$ and $M' \models \varphi$.*

Lemma 6. *Rule A in Figure 3 is correct for any cardinality formula φ.*

Proof. Assume a given net N, a marking M_0, and a cardinality formula φ. Let N' and M'_0 be the net and the initial marking after one application of Rule A. We shall argue that Rule A is correct. First, we define an equivalence relation $\equiv_A \subseteq \mathcal{M}(N) \times \mathcal{M}(N')$ such that $M \equiv_A M'$ if and only if

- $M'(p) = M(p)$ for all $p \in P \setminus \{p_0, p_1, \ldots, p_k\}$, and
- $M'(p) = M(p) + M(p_0) \cdot W(t_0, p)$ for all $p \in \{p_1, \ldots, p_k\}$.

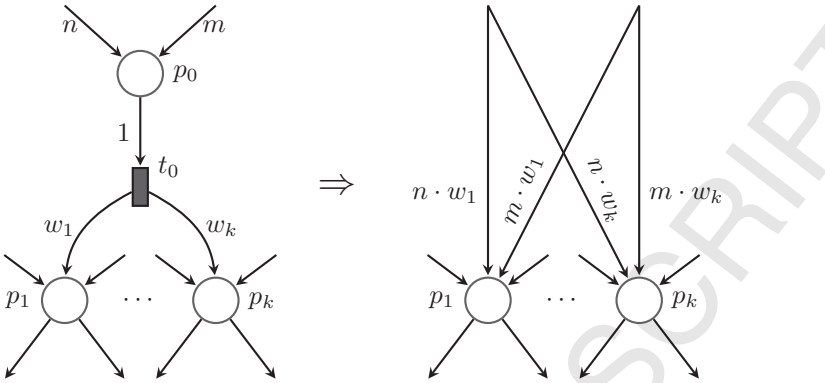
Let us first realize that $M \models \varphi$ iff $M' \models \varphi$ whenever $M \equiv_A M'$. This follows from Precondition A4 and the definition of $\equiv_A$. Moreover, due to Update UA1 we also have $M_0 \equiv_A M'_0$. Our lemma then follows from the next two properties. Let $M \equiv_A M'$ then

P1) if $M \xrightarrow{t} M_1$ then either $M_1 \equiv_A M'$ or $M' \xrightarrow{t} M'_1$ s.t. $M_1 \equiv_A M'_1$, and

P2) if $M' \xrightarrow{t} M'_1$ then $M \xrightarrow{t_0^n t} M_1$ for some $n \in \mathbb{N}^0$ s.t. $M_1 \equiv_A M'_1$.

Let us first argue for Property P1. There are two cases.

- Case $t = t_0$: We want to show that $M_1 \equiv_A M'$. For all $p \in P \setminus \{p_0, p_1, \ldots, p_k\}$ we clearly have $M'(p) = M_1(p)$ as firing t_0 only changes the number of tokens in $p_0, p_1, \ldots, p_k$. By Precondition A1 we observe

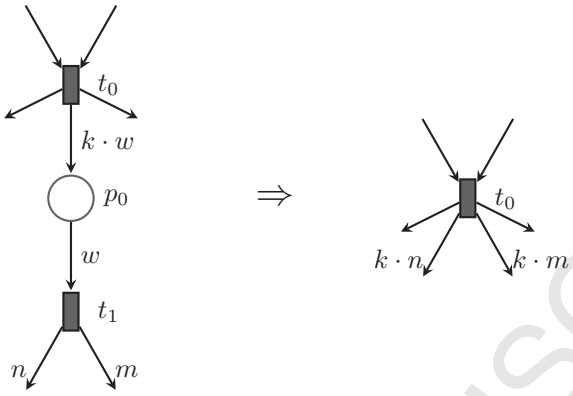


Precondition	Update
Fix p_0, t_0 where $t_0^\bullet = \{p_1, \ldots, p_k\}$ s.t. A1) ${}^\bullet t_0 = \{p_0\}$ and $W(p_0, t_0) = 1$. A2) $p_0^\bullet = \{t_0\}$ and $p_0 \notin \{p_1, \ldots, p_k\}$ A3) $p_0^\circ = p_1^\circ = \ldots = p_k^\circ = {}^\circ t_0 = \emptyset$ A4) $\{p_0, p_1, \ldots, p_k\} \cap places(\varphi) = \emptyset$	UA1) For all $p \in \{p_1, \ldots, p_k\}$ change the initial marking s.t. $M_0'(p) := M_0(p) + M_0(p_0) \cdot W(t_0, p)$ UA2) For all $t \in T \setminus \{t_0\}$ and all $p \in \{p_1, \ldots, p_k\}$ set $W'(t, p) := W(t, p_0) \cdot W(t_0, p) + W(t, p)$ UA3) Remove p_0 and t_0.

Figure 3: Rule A: Sequential Transition Removal

that firing of t_0 removes one token from p_0 and adds $W(t_0, p)$ tokens to p, for all $p \in \{p_1, \ldots, p_k\}$. Hence by our assumption that $M \equiv_A M'$ and the definition of $\equiv_A$ we have $M'(p) = M(p) + M(p_0) \cdot W(t_0, p) = M(p) + W(t_0, p) + (M(p_0) - 1) \cdot W(t_0, p) = M_1(p) + M_1(p_0) \cdot W(t_0, p)$ for all $p \in \{p_1, \ldots, p_k\}$. This means that $M'(p) = M_1(p) + M_1(p_0) \cdot W(t_0, p)$ for all $p \in \{p_1, \ldots, p_k\}$, implying that $M_1 \equiv_A M'$ as required.

- Case $t \neq t_0$: We want to show that $M' \xrightarrow{t} M_1'$ such that $M_1 \equiv_A M_1'$. As $M \equiv_A M'$ implies that $M(p) \leq M'(p)$ for all $p \in \{p_1, \ldots, p_k\}$ then together with A2 and A3 we get that $t \in en(M')$ and we can fire it such that $M' \xrightarrow{t} M_1'$. For all $p \in P \setminus \{p_0, p_1, \ldots, p_k\}$ we can easily notice that $M_1(p) = M_1'(p)$. Once t is fired from M', we get by UA2 that for all $p \in \{p_1, \ldots, p_k\}$ we have $M_1'(p) = M'(p) + W'(t, p) = M'(p) + W(t, p_0) \cdot W(t_0, p) + W(t, p)$. Because $M \equiv_A M'$ we know that for all $p \in \{p_1, \ldots, p_k\}$ also $M'(p) = M(p) + M(p_0) \cdot W(t_0, p)$. By substituting this to the equation above, we get $M_1'(p) = M(p) + M(p_0) \cdot W(t_0, p) + W(t, p_0) \cdot W(t_0, p) + W(t, p) = M(p) + W(t, p) + (M(p_0) + W(t, p_0)) \cdot W(t_0, p) = M_1(p) + M_1(p_0) \cdot W(t_0, p)$ which implies that $M_1 \equiv_A M_1'$ as required.



Precondition	Update
Fix p_0 and t_0, t_1 where $t_0 \neq t_1$ s.t. B1) ${}^\bullet p_0 = \{t_0\}$, $p_0^\bullet = \{t_1\}$, ${}^\bullet t_1 = \{p_0\}$ B2) $W(t_0, p_0) = k \cdot W(p_0, t_1)$ for $k \geq 1$ B3) $p_0^\circ = {}^\circ t_0 = {}^\circ t_1 = \emptyset$ B4) $p_0 \notin \mathit{places}(\varphi)$ B5) $p^\circ = \emptyset$ and $p \notin \mathit{places}(\varphi)$ for all $p \in t_1^\bullet$	UB1) For all $p \in P \setminus \{p_0\}$ set $M_0'(p) = M_0(p) + \lfloor M_0(p_0)/W(p_0, t_1) \rfloor \cdot W(t_1, p)$ UB2) For all $p \in P \setminus \{p_0\}$ set $W'(t_0, p) = W(t_0, p) + k \cdot W(t_1, p)$ UB3) Remove p_0 and t_1.

Figure 4: Rule B: Sequential place removal

Finally, let us finish the proof by arguing for Property P2. Let $M \equiv_A M'$ and we want to show that if $M' \xrightarrow{t} M_1'$ then we can fire the transition t_0 from M several times followed by the transition t and reach a marking M_1 such that $M_1 \equiv_A M_1'$. Clearly $t \neq t_0$ as the transition t_0 was removed in N'. Due to Precondition A1 we can fire t_0 in the marking M exactly $M(p_0)$ times so that we reach a marking with no tokens in p_0 and $M(p_0) \cdot W(t_0, p)$ additional tokens in each place $p \in \{p_1, \ldots, p_k\}$, so that they now contains exactly $M'(p)$ tokens. Now t must be enabled as we assume that it is enabled in M' and after firing t, we reach a marking M_1 such that $M_1 \equiv_A M_1'$. □

Lemma 7. *Rule B in Figure 4 is correct for any cardinality formula φ.*

Proof. Assume a given net N, a marking M_0, and a cardinality formula φ. Let N' and M_0' be the net and the initial marking after one application of Rule B. We shall argue that Rule B is correct. First, we define an equivalence relation $\equiv_B \subseteq \mathcal{M}(N) \times \mathcal{M}(N')$ such that $M \equiv_B M'$ if and only if $M'(p) = M(p) + \lfloor M(p_0)/W(p_0, t_1) \rfloor \cdot W(t_1, p)$ for all $p \in P \setminus \{p_0\}$.

Let us first realize that $M \models \varphi$ iff $M' \models \varphi$ whenever $M \equiv_B M'$. This follows from Precondition B4, B5, and the definition of $\equiv_B$. Moreover, due to Update UB1 we also have $M_0 \equiv_B M_0'$. Our lemma then follows from the next two properties. Let $M \equiv_B M'$ then

P1) if $M \xrightarrow{t} M_1$ then either $M_1 \equiv_B M'$ or $M' \xrightarrow{t} M_1'$ s.t. $M_1 \equiv_B M_1'$, and

P2) if $M' \xrightarrow{t} M_1'$ then $M \xrightarrow{t_1^n t} M_1$ for some $n \in \mathbb{N}^0$ s.t. $M_1 \equiv_B M_1'$.

Let us first argue for Property P1. There are three cases.

- Case $t = t_1$: We want to show that $M_1 \equiv_B M'$. For all $p \in P \setminus (\{p_0\} \cup t_1^\bullet)$ we clearly have $M'(p) = M_1(p)$ as firing of t_1 only changes the number of tokens in p_0 and places in $t_1^\bullet$. By Preconditions B1 and B2 we notice that firing of t_1 removes $W(p_0, t_1)$ tokens from p_0 and adds $W(t_1, p)$ tokens to p, for all $p \in t_1^\bullet$. Is it now easy to observe that $M_1 \equiv_B M'$.

- Case $t = t_0$: We want to show that $M_1 \equiv_B M_1'$ where $M' \xrightarrow{t_0} M_1'$. Observe that in the reduced net we have that $W'(t, p) = W(t, p) + k \cdot W(t_1, p)$ by Update UB2 for all $p \in t_1^\bullet$, which corresponds to also firing t_1 a total of k times. As before, we can see that $M_1 \equiv_B M_1'$.

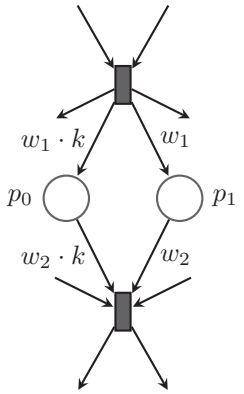
- Case $t \in T \setminus \{t_1, t_0\}$: As $M \equiv_B M'$ we know that $M(p) \leq M'(p)$ for all $p \in t_1^\bullet$ and thanks to B5, we notice that whenever $M \xrightarrow{t} M_1$ then also $M' \xrightarrow{t} M_1'$ and clearly $M_1 \equiv_B M_1'$.

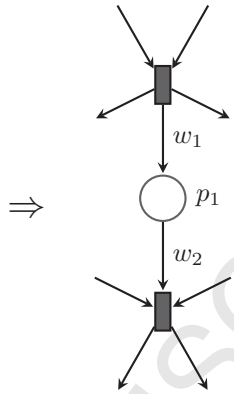
Let us finish the proof by arguing for Property P2. Let $M \equiv_B M'$ and we want to show that if $M' \xrightarrow{t} M_1'$ then we can fire from M the transition t_1 several times, followed by the transition t and reach a marking M_1 such that $M_1 \equiv_B M_1'$. Indeed, once we fire t_1 exactly $\lfloor M(p_0)/W(p_0, t_1) \rfloor$ times, the number of tokens in all places $p \in t_1^\bullet$ becomes equal to $M'(p)$ and thanks to B3 we can now fire t and reach a marking M_1 such that $M_1 \equiv_B M_1'$. □

Lemma 8. *Rule C in Figure 5 is correct for any cardinality formula φ.*

Proof. Assume a given net N, a marking M_0, and a cardinality formula φ. Let N' and M_0' be the net and the initial marking after one application of Rule C. We shall argue that Rule C is correct. Due to Precondition C3 we can see that $M(p_0) \geq M(p_1) \cdot k$ for every M such that $M_0 \to^* M$. Now any transition $t \in p_0^\bullet$ enabled from the marking M in the net with the place p_0 removed is also enabled in the original net and vice versa. Hence any marking M reachable from M_0 has a corresponding marking M' reachable from M_0' such that $M(p) = M'(p)$ for all $p \in P \setminus \{p_0\}$ and vice versa. By C1 and C2 we can now conclude that $M \models \varphi$ iff $M' \models \varphi$. □

Lemma 9. *Rule D in Figure 6 is correct for any cardinality formula φ.*





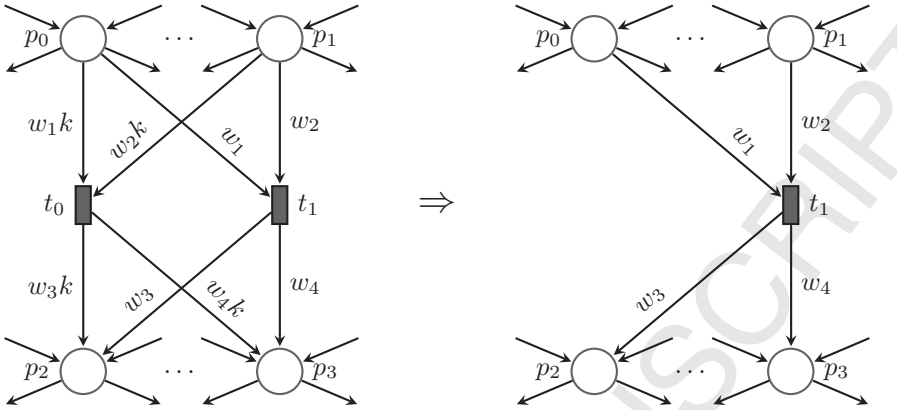
Precondition	Update
Fix p_0, p_1 where $p_0 \neq p_1$ s.t.	
C1) $p_0^\circ = p_1^\circ = \emptyset$	
C2) $p_0 \notin \mathit{places}(\varphi)$	
C3) there is $k \geq 1$ such that	UC1) Remove p_0.
(a) $M_0(p_0) \geq M_0(p_1) \cdot k$	
(b) $W(t, p_0) \geq W(t, p_1) \cdot k$ for all $t \in {}^\bullet p_0$	
(c) $W(p_0, t) \leq W(p_1, t) \cdot k$ for all $t \in p_0^\bullet$	

Figure 5: Rule C: Parallel place removal

Proof. Assume a given net N, a marking M_0, and a cardinality formula φ. Let N' and M_0' be the net and the initial marking after one application of Rule D. We shall argue that Rule D is correct. Obviously, $M_0 = M_0'$ since Rule D does not change token counts in places. Moreover, from Precondition D2 it follows that the behaviour of t_0 can be replicated by firing t_1 exactly k times. Let $M_0 \xrightarrow{w} M$ such that $M \models \varphi$ and let w' be w with all occurrences of t_0 replaced by t_1^k. From the observation above and by D1 we get that in the net N' we have $M_0' \xrightarrow{w'} M'$ such that $M = M'$ and hence $M' \models \varphi$. The other direction is trivial as any behaviour of N' can be directly mimicked by N. □

Lemma 10. *Rule E in Figure 7 is correct for any cardinality formula φ.*

Proof. Observe that by Preconditions E1 and E2 the transition t_0 is never enabled in any reachable marking from M_0 as $M_0(p_0) < W(p_0, t_0)$ and ${}^+p_0 = \emptyset$. Hence removing t_0 does not the change behaviour of N. Moreover, should the place p_0 become isolated after the removal of t_0, it can be removed too by UE1, provided that it is not used in the formula φ and it is not connected to any inhibitor arc. □



Precondition	Update
Fix t_0, t_1 where $t_0 \neq t_1$ s.t. D1) ${}^\circ t_0 = {}^\circ t_1 = \emptyset$ D2) there is $k \geq 1$ s.t. for all $p \in P$ (a) $W(p, t_0) = W(p, t_1) \cdot k$ (b) $W(t_0, p) = W(t_1, p) \cdot k$	UD1) Remove t_0.

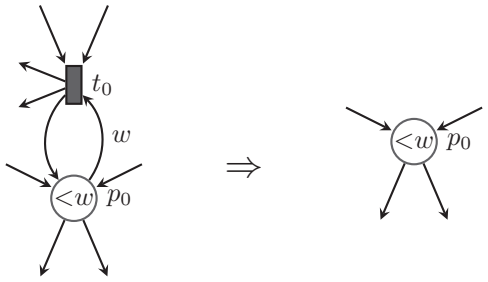
Figure 6: Rule D: Parallel transition removal

Lemma 11. *Rule F in Figure 8 is correct for any cardinality formula φ.*

Proof. By F3 we know that ${p_0}^- = \emptyset$ and $M(p_0) \geq W(p_0, t)$ for all $t \in T$. Hence the number of tokens in p_0 can never drop and p_0 never disables any transition connected to it. Due to Precondition F1 and F2 it is so safe to remove the place p_0 without changing the behaviour of the net. □

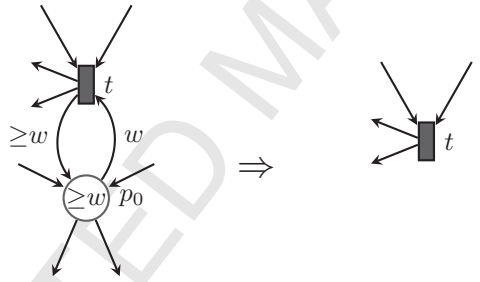
Lemma 12. *Rule G in Figure 9 is correct for any cardinality formula φ.*

Proof. Assume a given net N, a marking M_0, and a cardinality formula φ. Let N' and M_0' be the net and the initial marking after one application of Rule G. We shall argue that Rule G is correct. By G2 and G3 we get that $t_0 \notin {}^+p$ for all $p \in P$, and $t_0 \notin p^-$ for all $p \in \mathit{places}(\varphi)$. Hence the firing of t_0 does not change the number of tokens in any place that appears in φ and can only preserve or decrease the number of tokens in any other connected place. Let $M_0 \xrightarrow{w} M$ in N and let w' be w with all occurrences of t_0 removed. By G1 and the previous argument, we get that also $M_0' \xrightarrow{w'} M'$ and clearly $M(p) = M'(p)$ for all $p \in \mathit{places}(\varphi)$, implying that $M \models \varphi$ iff $M' \models \varphi$. The other direction where $M_0' \to^* M'$ in N' is trivial as the same marking M' can be reached also from M_0 in N. □



Precondition	Update
Fix p_0 and t_0 s.t. E1) $M_0(p_0) < W(p_0, t_0)$ E2) $W(t, p_0) \leq W(p_0, t)$ or $M_0(p_0) < W(p_0, t)$ for all $t \in T$	UE1) If $p_0^\bullet = \{t_0\}$, $p_0^\circ = \emptyset$ and $p_0 \notin \mathit{places}(\varphi)$ then remove p_0. UE2) Remove t_0.

Figure 7: Rule E: Dead transition removal

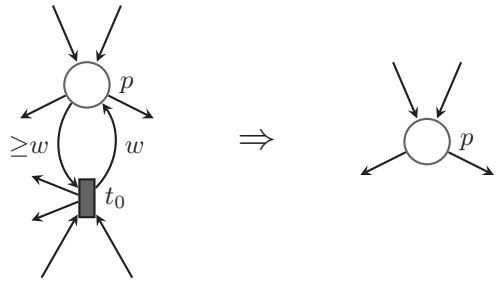


Precondition	Update
Fix p_0 s.t. F1) $p_0^\circ = \emptyset$ F2) $p_0 \notin \mathit{places}(\varphi)$ F3) $W(t, p_0) \geq W(p_0, t)$ and $M_0(p_0) \geq W(p_0, t)$ for all $t \in T$	UF1) Remove p_0.

Figure 8: Rule F: Redundant place removal

Lemma 13. *Rule H in Figure 10 is correct for any cardinality formula φ.*

Proof. Assume a given net N, a marking M_0, and a cardinality formula φ. Let N' and M_0' be the net and the initial marking after one application of Rule H. We shall argue that Rule H is correct. Let us define $\equiv_H \subseteq \mathcal{M}(N) \times \mathcal{M}(N')$ such that $M \equiv_H M'$ if and only if $M(p) = M'(p)$ for all $p \in P \setminus \{p_0, p_1\}$ and $M'(p_1) = M(p_1) + M(p_0)$. By Precondition H4 we know that $M \models \varphi$ iff $M' \models \varphi$



Precondition	Update
Fix t_0 s.t. G1) $^\circ t_0 = \emptyset$ and $p^\circ = \emptyset$ for all $p \in {}^\bullet t_0$ G2) $t_0^\bullet \subseteq {}^\bullet t_0$ G3) for all $p \in {}^\bullet t_0$ we have either • $W(p, t_0) = W(t_0, p)$, or • $W(p, t_0) > W(t_0, p)$ and $p \notin \mathit{places}(\varphi)$	UG1) Remove t_0.

Figure 9: Rule G: Redundant transition removal

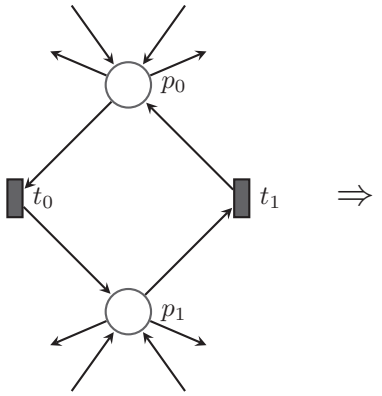
whenever $M \equiv_H M'$.

Let $M_0 \xrightarrow{w} M$ such that $M \models \varphi$. Let w' be w with all occurrences of t_0 removed. Due to the construction of the updated net and UH4, together with Precondition H3 we know that also $M_0' \xrightarrow{w'} M'$ such that $M \equiv_H M'$, giving us $M' \models \varphi$.

For the other direction, let $M_0' \xrightarrow{w'} M'$ such that $M' \models \varphi$. We want to find a sequence w such that $M_0 \xrightarrow{w} M$ and $M \equiv_H M'$, which implies that $M \models \varphi$. We prove this by induction on the length of w'. The base case follows from UH4. Let $M_0' \xrightarrow{w'} M' \xrightarrow{t} M_1'$ and assume by the induction hypothesis that $M_0 \xrightarrow{w} M$ for some w such that $M \equiv_H M'$. Clearly, if t is enabled from M then we let $M \xrightarrow{t} M_1$ and reach a marking M_1 such that $M_1 \equiv_H M_1'$. If t is not enabled from M then we can rearrange the tokens in the places p_0 and p_1 by firing the transitions t_0 and t_1 so that we reach a marking where t becomes enabled (this is possible due to the construction of the net N'). After firing t we get a marking M_1 such that $M_1 \equiv_H M_1'$ and we are done with the inductive argument. □

Finally, we present a slightly different structural reduction rule that in one run computes a set of places and transitions that are safe to remove for the validity of a given cardinality formula φ. This Rule I is given in Algorithm 2.

Lemma 14. *Rule I in Algorithm 2 is correct for any cardinality formula φ.*



Precondition	Update
Fix different p_0, p_1, t_0, t_1 s.t. H1) ${}^\bullet t_0 = t_1^\bullet = \{p_0\}$ H2) ${}^\bullet t_1 = t_0^\bullet = \{p_1\}$ H3) $p_0^\circ = p_1^\circ = {}^\circ t_0 = {}^\circ t_1 = \emptyset$ H4) $p_0 \notin \mathit{places}(\varphi), p_1 \notin \mathit{places}(\varphi)$ H5) $W(p_0,t_0) = W(t_0,p_1) = W(p_1,t_1) = W(t_1,p_0) = 1$	UH1) $W'(t,p_1) = W(t,p_1) + W(t,p_0)$ for all $t \in T$ UH2) $W'(p_1,t) = W(p_1,t) + W(p_0,t)$ for all $t \in T$ UH3) $W'(t_1,p_1) = W'(p_1,t_1) = 1$ UH4) $M_0'(p_1) = M_0(p_1) + M_0(p_0)$ UH5) Remove t_0. UH6) Remove p_0.

Figure 10: Rule H: Simple cycle removal

Proof. Assume a given net N, a marking M_0, and a cardinality formula φ. Let N' and M_0' be the net and the initial marking after the application of Rule I (Algorithm 2). Let P' and T' be the set of places and transitions in the net N', respectively. We shall argue that Rule I is correct. Let us first define a relation $\equiv_I \subseteq \mathcal{M}(N) \times \mathcal{M}(N')$ such that $M \equiv_I M'$ if and only if

- $M(p) = M'(p)$ for all $p \in \mathit{places}(\varphi)$,
- $M(p) \leq M'(p)$ for all $p \in {}^\bullet T'$, and
- $M(p) \geq M'(p)$ for all $p \in {}^\circ T'$.

Let $M \equiv_I M'$. Then clearly $M \models \varphi$ iff $M' \models \varphi$ due to the definition of $\equiv_I$ and the fact that $\mathit{places}(\varphi) \subseteq P'$. Moreover, trivially also $M_0 \equiv_I M_0'$.

Let $M \equiv_I M'$. We will show that if $M \xrightarrow{t} M_1$ then either $M' \xrightarrow{t} M_1'$ such that $M_1 \equiv_I M_1'$, or $M_1 \equiv_I M'$. There are two cases to consider.

- Case $t \in T'$. Due to the second and third condition in the definition of $\equiv_I$, we know that t is also enabled in the marking M' and we can fire $M' \xrightarrow{t} M_1'$. After firing t both from M and M' we clearly preserve all three conditions of $\equiv_I$ and $M_1 \equiv_I M_1'$.

Algorithm 2: Rule I: Removal of irrelevant places and transitions

input : A net $N = (P, T, W, I)$, initial marking M_0, and a cardinality formula φ.
output : A reduced net N' and its initial marking M'_0.

1 $X := \emptyset$; $unprocessed := \bigcup_{p \in places(\varphi)} p^- \cup {}^+p$;
2 **while** $unprocessed \neq \emptyset$ **do**
3 pick any $t \in unprocessed$;
4 $unprocessed := unprocessed \cup ({}^+({}^\bullet t) \setminus X)$;
5 $unprocessed := unprocessed \cup ((^\circ t)^- \setminus X)$;
6 $unprocessed := unprocessed \setminus \{t\}$;
7 $X := X \cup \{t\}$;
8 Let $P' = {}^\bullet X \cup {}^\circ X \cup places(\varphi)$.
9 Let $T' = X$.
10 Modify N by removing all places $P \setminus P'$ and all transitions $T \setminus T'$.
11 Let N' be the modified net and let $M'_0(p) = M_0(p)$ for all $p \in P'$.
12 **return** N' *and* M'_0

- Case $t \notin T'$. We want to argue that $M_1 \equiv_I M'$. We notice that Algorithm 2 returns a net where ${}^+({}^\bullet T') \subseteq T'$ and $({}^\circ T')^- \subseteq T'$. Hence firing of $t \notin T'$ in the net N cannot increase the number of tokens in any place from ${}^\bullet T'$ and cannot decrease the number of tokens in any place from ${}^\circ T'$. Moreover, the firing of t cannot change the number of tokens in $places(\varphi)$ as $p^- \cup {}^+p \subseteq T'$ for every $p \in places(\varphi)$. As a result, all three conditions of definition $\equiv_I$ are met and we can conclude that $M_1 \equiv_I M'$.

For the other direction, we notice that N' is a subnet of N. Hence whenever $M'_0 \xrightarrow{w} M'$ then also $M_0 \xrightarrow{w} M$ such that $M \equiv_I M'$. □

We can now summarize the correctness of the structural rules for any given cardinality formula in the following theorem. Moreover, we also notice that all rules, except for G and I, preserve also the presence of a reachable deadlock marking. An application of Rule G can create a deadlock in the modified net and Rule I can both create new deadlocks as well as remove existing deadlocks.

Theorem 15. *Rules A to I are correct for any given cardinality formulae. Rules A to F and H are moreover correct also for the deadlock formula.*

5. Experimental Evaluation

We implemented the stubborn set reduction and structural reduction rules in the verification engine `verifypn` [6] (source code is available at `https://code.launchpad.net/~verifypn-cpn/verifypn/struct_vs_stub`) as a part of the model checking tool TAPAAL [3]. Our experiments are executed using

		Number of solved queries			
	Queries	**Base**	**Stub**	**Struct**	**StubStruct**
RC	7008	3733	4807	4794	5325
RF	7008	4864	5503	5403	5820
RD	438	288	344	333	367
Total	14454	8885	10654	10530	11512

Table 3: Number of queries solved by each algorithm

the database of Petri net models and reachability queries from MCC'17 [8]. For each model (there are in total 438 Petri nets, including the known and surprise nets) there are three categories of queries: reachability cardinality (RC), reachability fireability (RF) and reachability deadlock (RD). In the RC and RF category there are 16 queries for each model and in RD there is only a single query. In some tables, we further subdivide the queries into RC+, RF+ and RD+ whenever there exists evidence (finite trace) proving the property and into RC–, RF– and RD– where the whole state space must be explored before the validity of the property can be established. All of the above experiments were run on AMD Opteron 6376 processors with a 14 GB memory limit.

5.1. Comparison of Stubborn vs. Structural Reduction

In Table 3 we can see the number of queries solved by each algorithm (we set a 20 minute timeout for RC and RF, and a 1 hour timeout for RD). Here **Base** stands for the standard TAPAAL without stubborn and structural reductions, **Stub** adds to **Base** the stubborn set reduction, and **Struct** adds to **Base** the structural reductions. Finally, **StubStruct** stands for the engine that first applies structural reductions while preprocessing the nets and then uses the stubborn set reduction during the state-space search. In all cases, we use the heuristic search strategy implemented in TAPAAL. The practical applicability of both the stubborn and structural reduction is clear as each method independently allows to solve more than 1600 supplementary queries. However, more interestingly, the combination of both methods solves 858 additional queries compared to **Stub** and 982 additional queries compared to **Struct**. This demonstrates, even though the two methods both reduce the concurrency present in the model, they are not conflicting with each other and it is beneficial to apply them both during model checking.

A detailed pairwise comparison of the methods is presented in Table 4. For each query, an algorithm gets a point relative to another algorithm, as follows. **Exclusive:** answers the query while the opponent algorithm does not provide any answer. **Time:** answers the query at least 50% faster, disregarding queries that are solved in less than 10 seconds by both algorithms. **Memory:** answers the query by using at least 50% less peak memory. If an algorithm solves a query exclusively, it also gets a point in the time and memory comparison. As already discussed, the addition of stubborn and structural reduction significantly improves the verification of **Base** as indicated in Table 4a and 4b. In both cases,

Base vs Stub						
	exclusive		time		memory	
RC+	33	336	97	553	52	627
RC–	5	776	6	901	5	913
RF+	16	490	78	867	41	948
RF–	5	170	7	248	5	256
RD+	2	4	8	12	6	11
RD–	0	52	0	68	0	69
SUM	61	1828	196	2649	109	2824

(a)

Base vs Struct						
	exclusive		time		memory	
RC+	11	304	97	578	40	571
RC–	0	768	0	907	0	904
RF+	12	367	133	739	37	723
RF–	0	184	0	286	0	279
RD+	0	1	8	5	0	7
RD–	0	42	0	52	0	53
SUM	23	1666	238	2567	77	2537

(b)

Base vs StubStruct						
	exclusive		time		memory	
RC+	13	470	96	797	43	854
RC–	5	1140	6	1308	5	1318
RF+	19	657	142	1122	52	1195
RF–	4	322	5	453	4	454
RD+	2	4	12	9	6	13
RD–	0	77	0	93	0	94
SUM	43	2670	261	3782	110	3928

(c)

Stub vs Struct						
	exclusive		time		memory	
RC+	149	139	325	342	346	264
RC–	266	263	336	330	346	315
RF+	256	137	537	332	527	263
RF–	105	124	149	180	163	166
RD+	4	3	16	9	7	7
RD–	25	15	37	25	39	25
SUM	805	681	1400	1218	1428	1040

(d)

Stub vs StubStruct						
	exclusive		time		memory	
RC+	10	164	91	392	38	322
RC–	0	364	35	460	0	473
RF+	18	182	151	431	47	382
RF–	0	153	0	229	0	232
RD+	0	0	8	5	0	2
RD–	0	25	0	36	0	37
SUM	28	888	285	1553	85	1448

(e)

Struct vs StubStruct						
	exclusive		time		memory	
RC+	21	185	42	306	39	402
RC–	7	374	22	528	7	570
RF+	29	312	54	503	58	611
RF–	10	144	13	219	10	254
RD+	3	4	6	7	7	8
RD–	0	35	0	49	0	51
SUM	70	1054	137	1612	121	1896

(f)

Table 4: Algorithms comparison (7008 queries in RC and RF and 438 queries in RD)

Base still provides some exclusive answers. For **Stub** this is the case as there are nets where the stubborn sets include almost all enabled transitions, meaning that the construction of stubborn sets leaves us only with an overhead. In the case of **Struct** there are only a few exclusive answers and only for the cases where there exists a witness trace. Due to the changed structure of the net after the reduction, the search strategy gets modified and in 23 cases **Base** was lucky to find the witness trace faster even though the state space is larger. Clearly, the combination of stubborn and structural reduction computes the largest number of exclusive answers as shown in Figure 4c. The comparison of **Stub** and **Struct** in Table 4d shows a slightly higher number of exclusive answers when only stubborn set reduction is used, which is reflected also by the points for the time and memory comparison. The advantage of the combination of both methods, compared to an independent use of each one, is documented by a high number of new exclusive answers in Tables 4f and 4e. Some exclusive answers are lost when combining both techniques, but as this is the case mainly for the queries with a witness trace. As before, we contribute this to the modified search strategy after combining the two methods.

The reason why the combination of the two techniques is indeed beneficial seems to be twofold: (i) in preprocessing a net by applying first the structural

	Rule applications (×1000)									% Reduction
Disabled Rule	A	B	C	D	E	F	G	H	I	Trans+Places
A	0	5557	115	10760	121	705	5294	598	8	38.9686
B	6613	0	181	11279	120	812	6086	624	11	42.9040
C	6594	1	0	11279	120	757	5957	624	10	42.4008
D	6318	1	129	0	120	733	5712	669	10	30.3359
E	6610	1	180	11279	0	825	6082	624	11	41.9651
F	6503	1	674	10727	250	0	6012	622	11	41.9089
G	6471	1	36	14879	104	664	0	636	15	39.9450
H	6598	1	129	11212	120	757	2824	0	6	37.2231
I	7213	1	199	11712	611	1648	8755	631	0	40.3852
None	6613	1	181	11279	120	812	6074	624	11	42.9053

Table 5: Number of applications of reductions rules (10 minutes timeout)

reduction, the size of the net usually decreases considerably and this implies less overhead and fewer dependencies when (on-the-fly) computing the stubborn sets, on the other hand (ii) structural reductions remove only the behaviour that is detectable statically (without the knowledge of the actual marking) whereas stubborn reduction computes the pruning of the state-space dynamically by considering also the given marking that we are exploring.

We also remark that our refinement of the stubborn set method via the increasing/decreasing presets/postsets of places demonstrates an average reduction in running time by 13% (measured without the employment of other reduction techniques). On some nets there is not any noticeable improvement while e.g. on the model RAFT-PT we achieve a speedup of 98% (average over all instances of the model).

5.2. Comparison of Different Structural Reduction Rules

As we provided a number of new or extended rules for structural reduction, we investigate their potential applicability in Table 5 across the whole database of models from MCC'17. In the last row of the table, we show the total number of times (in thousands) each rule was applied across all models in all categories. We also disable the application of each single rule and investigate how it influences the applicability of the remaining rules. The most obvious dependency is between rules A and B that can to a large degree substitute each other, in particular in the situations where a net contains a longer sequential chain of transition firings. As in our implementation rule A is applied (as long as possible) before we proceed to reduce by rule B, less than one thousand applications of rule B are observed. However, if rule A is disabled (the first row in the table) then rule B is applied approximately 5557.000 times. Another dependency is between the rules G and H. As seen the table, if rule H is disabled then the applicability of rule G drops considerably to 2824 thousands of applications, caused by the fact that rule H is creating new transition loops that rule G can remove (provided that the preconditions are satisfied). Otherwise the remaining rules are frequently used with rules D and G being the most applicable ones. The only exception is rule I. Due to the different nature of this rule, we achieve

		Solved queries	
	Queries	TAPAAL	LoLA
RC	7008	6640	6568
RF	7008	6376	6321
RD	438	377	364
Total	14455	13392	13253

Table 6: Comparison between TAPAAL and LoLA

a considerable reduction effect on several nets but in general as soon as rule I is applied once on a given net and query, it is unlikely that more than a few further applications become possible.

Finally, we run an experiment with different orders in which the reduction rules are applied. We enumerated all possible rule permutations and executed them on all models and a selected reachability cardinality query. In one day, 252 nets completed the reductions under all given permutations. In 228 nets the size of the reduced net did not depend on the order of application of the reduction rules and only in 24 nets there was a smaller difference in the size of the reduced net. The reduction order used in our experiments was not the optimal one in only 10 cases.

5.3. Comparison with LoLA

Finally, we also compare the performance of our tool with LoLA [21], the winner of the MCC'17 competition in the reachability category. We use the current development snapshots of LoLA (based on version 2.0) and Sara (based on version 1.14)—in MCC'17 LoLA was running in parallel with Sara. Here we use the same rules as above for awarding points, but we run two parallel processes (three in the RD-category) for each tool. For RC and RF we run LoLA using (i) `--stateequation=alone` which calls the tool Sara and (ii) `--stateequation=none` which calls the standard engine of LoLA. For RD we run LoLA using (i) `--symmetry --symmtimelimit=300 --stubborn=tarjan`, with (ii) `--symmetry --symmtimelimit=300 --findpath=alone` and with (iii) `--symmetry --symmtimelimit=300 --siphondepth=10 --siphontrap=alone` as suggested by LoLA developers as the recommended strategy for the tool. For our tool in the RC and RF categories we use in parallel (i) the default options and (ii) `-tar` enabling a trace abstraction refinement method based on [2]. For RD we run our tool in parallel using (i) the default options, (ii) `-tar` and (iii) `--siphon-trap 3600 --siphon-depth 10`. We terminate the parallel computation as soon as the fastest thread finishes its computation.

Table 6 proves that after implementing the stubborn and structural reductions in TAPAAL, we can solve a higher number of reachability queries than the tool LoLA. More precisely we can answer 139 additional queries over all three categories. A detailed comparison performed in Table 7 reveals that while LoLA is better in answering queries that have a witness trace, TAPAAL achieves a significant margin on the queries where the whole state space must be searched.

TAPAAL vs LoLA						
	exclusive		**time**		**memory**	
RC+	51	202	150	647	221	693
RC–	235	12	554	47	691	65
RF+	98	266	159	1346	381	826
RF–	207	19	314	150	362	47
RD+	4	20	43	29	39	53
RD–	30	1	48	9	62	7
SUM	625	520	1268	2228	1756	1691

Table 7: Pairwise score comparison between TAPAAL and LoLA

We contribute this to improved and extended structural reduction rules suggested in this paper. Moreover, LoLA runs in parallel the tool Sara that uses advanced state equations techniques instead of the explicit state space search, resulting in about 1000 extra points where LoLA solved the query faster. On the other hand, TAPAAL is showing a slightly better performance in terms of memory usage, likely due to the employment of the PTrie [7] data structure for storing the explored state space.

6. Conclusion

We described the stubborn set and structural reduction techniques for the use on reachability queries on weighted Petri nets with inhibitor arcs. The stubborn set reduction was first presented on general labelled transition systems and then specialized for the application on Petri nets. We extended the technique to deal with inhibitor arcs as well as refining its performance to take weighted arcs into account. Similarly, we extended some of the classical structural reduction rules to nets with weighted arcs and inhibitor arcs and suggested a number of additional reductions rules, while demonstrating their applicability on the nets from the annual model checking contest. Both techniques were proved correct and experimentally evaluated.

Our main conclusion is—while it may intuitively seem as contra productive to employ both stubborn and structural reductions at the same time as they both target similar phenomena in order to restrict concurrency—that the combination of the techniques is clearly beneficial and the possible overhead when computing the reductions pays off. As a result, we have an efficient implementation of our verification engine that is now part of the open source model checker TAPAAL, and our engine, in all three reachability subcategories, solves more queries than LoLA, the last year winner of the model checking contest.

In our future work, we plan to add a support for colored Petri nets and extend the techniques discussed in this paper so that they can be applied directly on colored nets before their unfolding into P/T nets.

Acknowledgements. We would like to thank Karsten Wolf and Torsten Liebke from Rostock University for providing us with the development snapshot of the latest version of LoLA and Sara and for their help with setting up their tool and answering our questions. 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, Brno.

References

[1] C. Baier and J.-P. Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[2] F. Cassez, P.G. Jensen, and K.G. Larsen. Refinement of Trace Abstraction for Real-Time Programs. In *International Workshop on Reachability Problems*, volume 10506 of *LNCS*, pages 42–58. Springer Cham, 2017.

[3] 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 *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 7214 of *LNCS*, pages 492–497. Springer Berlin Heidelberg, 2012.

[4] J. Esparza. *Decidability and complexity of Petri net problems — An introduction*, pages 374–428. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[5] P. Godefroid. *Partial-Order Methods for the Verification of Concurrent Systems: An Approach to the State-Explosion Problem*. Springer-Verlag, 1996.

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

[7] P.G. Jensen, K.G. Larsen, and J. Srba. PTrie: Data structure for compressing and storing sets via prefix sharing. In *Proceedings of the 14th International Colloquium on Theoretical Aspects of Computing (ICTAC'17)*, volume 10580 of *LNCS*, pages 248–265. Springer, 2017.

[8] F. Kordon, H. Garavel, L. M. Hillah, F. Hulin-Hubard, B. Berthomieu, G. Ciardo, M. Colange, S. Dal Zilio, E. Amparore, M. Beccuti, T. Liebke, J. Meijer, A. Miner, C. Rohr, J. Srba, Y. Thierry-Mieg, J. van de Pol, and K. Wolf. Complete Results for the 2017 Edition of the Model Checking Contest. http://mcc.lip6.fr/2017/results.php, June 2017.

[9] 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.

[10] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[11] T. Murata and J.Y. Koh. Reduction and expansion of live and safe marked graphs. *IEEE Transactions on on Circuits and Systems*, 27(1):68–71, 1980.

[12] C.A. Petri. *Kommunikation mit automaten*. PhD thesis, Institut für instrumentelle Mathematik, Bonn, 1962.

[13] K. Schmidt. Stubborn Sets for Standard Properties. In *International Conference on Application and Theory of Petri nets*, volume 1639 of *LNCS*, pages 46–65. Springer Berlin Heidelberg, 1999.

[14] K. Schmidt. Integrating Low Level Symmetries into Reachability Analysis. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 1785 of *LNCS*, pages 315–330. Springer Berlin Heidelberg, 2000.

[15] A. Valmari. Stubborn sets for reduced state space generation. In *International Conference on Application and Theory of Petri Nets*, volume 483 of *LNCS*, pages 491–515. Springer Berlin Heidelberg, 1989.

[16] A. Valmari. A stubborn attack on state explosion. *Formal Methods in System Design*, 1(4):297–322, 1992.

[17] A. Valmari. Stubborn Sets with Frozen Actions. In *International Workshop on Reachability Problems*, volume 10506 of *LNCS*, pages 160–175. Springer Berlin Heidelberg, 2017.

[18] A. Valmari and H. Hansen. Stubborn Set Intuition Explained. In *Transactions on Petri Nets and Other Models of Concurrency XII*, volume 10470 of *LNCS*, pages 140–165. Springer Berlin Heidelberg, 2017.

[19] A. Valmari and W. Vogler. Fair Testing and Stubborn Sets. In *International Symposium on Model Checking Software*, volume 9641 of *LNCS*, pages 225–243. Springer Berlin Heidelberg, 2016.

[20] H. Wimmel and K. Wolf. Applying CEGAR to the Petri Net State Equation. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 6605 of *LNCS*, pages 224–238. Springer Berlin Heidelberg, 2011.

[21] K. Wolf. Running LoLA 2.0 in a Model Checking Competition. In *Transactions on Petri Nets and Other Models of Concurrency XI*, volume 9930 of *LNCS*, pages 274–285. Springer Berlin Heidelberg, 2016.