# TAPAAL and Reachability Analysis of P/T Nets

Jonas F. Jensen, Thomas Nielsen, Lars K. Oestergaard, and Jiří Srba

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

**Abstract.** We discuss selected model checking techniques used in the tool TAPAAL for the reachability analysis of weighted Petri nets with inhibitor arcs. We focus on techniques that had the most significant effect at the 2015 Model Checking Contest (MCC). While the techniques are mostly well known, our contribution lies in their adaptation to the MCC reachability queries, their efficient implementation and the evaluation of their performance on a large variety of nets from MCC'15.

## 1 Introduction

Petri nets [15] are a popular formalism for a high level modelling of distributed systems. Currently, there are more than 80 tools registered in the database of Petri net tools [8] and an annual model checking contest aiming at comparing the performance of the different tools has been running since 2011. In the last two editions of the contest, MCC'14 [10] and MCC'15 [11], our model checker TAPAAL [4] won a second place in the reachability category. In this paper, we report on the main verification techniques implemented in our tool and demonstrate their performance on the class of Petri nets from the latest edition of the model checking contest.

TAPAAL is a tool suite that apart from the verification engine for P/T nets supports also the modelling and analysis of a timed extension of the Petri net formalism called timed-arc Petri nets (for more details see [9]). The tool supports both continuous and discrete time verification and while the details about the continuous-time engine [5] and the discrete-time engine [1] were previously published, the untimed verification engine has not been presented yet.

We focus here solely on the TAPAAL verification techniques directly related to our participation in the model checking contest. The details about the other participating tools and a report on the competition results can be found in [11]. In what follows, we first describe an efficient heuristic search technique for explicit exploration of the Petri net state-space, then we discuss the adaptation of the state-equation approach to the case of cardinality queries and finally we demonstrate the applicability of the sequential and parallel structural reduction rules into the context of checking cardinality queries on weighted nets with inhibitor arcs.

TAPAAL is open-source and publicly available at www.tapaal.net. Citations to the related work connected to the techniques used in our tool are given at the respective sections of the paper. All experiments reported in this paper use

the competition nets and queries from MCC'15 but the verification was rerun locally as we needed to compare the different options and techniques (the data for the different combinations of these parameters is not available at the MCC'15 web-page as we submitted there only the best working configuration of our tool).

## 2 Definitions

Let $\mathbb{N}_0$ denote the set of natural numbers including zero. A Petri net (PN) with inhibitor arcs is a tuple $N = (P, T, F, I)$ where

- $P$ is a finite, nonempty set of *places*,
- $T$ is a finite set of *transitions* such that $P \cap T = \emptyset$,
- $F : (P \times T) \cup (T \times P) \to \mathbb{N}_0$ is the *flow* function, and
- $I \subseteq P \times T$ is the set of inhibitor arcs such that $(p, t) \in I$ implies $F(p, t) = 0$.

Let $N = (P, T, F, I)$ be a PN. A *marking* is a mapping $M : P \to \mathbb{N}_0$ that assigns tokens to places. The set $\mathcal{M}(N)$ denotes the infinite set of all markings on $N$. A *marked PN* is a pair $(N, M_0)$ where $M_0 \in \mathcal{M}(N)$ is an initial marking.

The *preset* of a place/transition $y$ is defined as ${}^\bullet y \stackrel{def}{=} \{z \in P \cup T \mid F(z, y) > 0\}$. Likewise, the *postset* is $y^\bullet \stackrel{def}{=} \{z \in P \cup T \mid F(y, z) > 0\}$. We denote the set of inhibitor places of a transition $t$ as $I(t) \stackrel{def}{=} \{p \in P \mid (p, t) \in I\}$ and transitions that a place $p$ inhibits as $I(p) \stackrel{def}{=} \{t \in T \mid (p, t) \in I\}$.

A transition $t \in T$ is *enabled* in a marking $M$ if for all $p \in {}^\bullet t$ we have $F(p, t) \leq M(p)$ and $M(p) = 0$ for all $p \in I(t)$. A transition $t$ enabled in a marking $M$ can *fire* and produce a marking $M'$ such that $M'(p) = M(p) - F(p, t) + F(t, p)$ for all $p \in P$, written as $M \xrightarrow{t} M'$. This firing relation is in a natural way extended to a sequence of transitions $w \in T^*$ so that $M \xrightarrow{\epsilon} M$ and for $w = tw'$ we write $M \xrightarrow{w} M'$ if $M \xrightarrow{t} M''$ and $M'' \xrightarrow{w'} M'$. We also write $M \to M'$ if $M \xrightarrow{t} M'$ for some $t \in T$. The reflexive and transitive closure of $\to$ is denoted by $\to^*$. Finally, let $\mathcal{R}(M) = \{M' \mid M \to^* M'\}$ be the set of markings reachable from $M$.

As usual, Petri net places are denoted by circles and can contain dots representing tokens, transitions are drawn as rectangles, input and output arcs are depicted as arrows labelled with their weights (if a label is missing we assume the default weight 1) and inhibitor arcs are denoted by circle-headed arrows.

After having introduced the standard syntax and semantics of Petri nets, we shall now define the reachability problem for cardinality queries, as the main MCC'15 competition category in the reachability analysis.

A *cardinality formula* is given by the abstract syntax

$$\begin{aligned} \varphi &::= e \bowtie e \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \\ e &::= n \mid p \mid e + e \mid e - e \mid n \cdot e \end{aligned}$$

where $\bowtie \in \{\leq, <, =, \neq, >, \geq\}$, $n \in \mathbb{N}_0$ and $p \in P$.

The satisfaction relation $M \models \varphi$ for a given marking is defined in the natural way such that $M \models e_1 \bowtie e_2$ iff $\mathit{eval}(M, e_1) \bowtie \mathit{eval}(M, e_2)$ where $\mathit{eval}(M, e)$ is the evaluation of the arithmetical expression $e$ into a number, assuming that $\mathit{eval}(M, p) = M(p)$ for $p \in P$ (in other words, a place $p$ evaluates to the number of tokens currently present in it).

For a marked Petri net $(N, M_0)$, we write $(N, M_0) \models \mathit{EF}\,\varphi$ if there is a marking $M$ such that $M_0 \rightarrow^* M$ and $M \models \varphi$. As an example, the query $\mathit{EF}\,p \geq 5 \wedge q \neq 3$ asks whether we can reach a marking where the place $p$ contains at least 5 tokens and the number of tokens in the place $q$ is different from 3.

Note that the MCC'15 verification queries [11] also contain other types of reachability questions: (i) *reachability fireability* where we consider the atomic proposition $\mathit{fire}(t)$ that is true in a given marking iff the transition $t$ is fireable, (ii) *reachability compute bounds* where the expression $\mathit{bounds}(X)$ for $X \subseteq P$ is added as an atomic expression of $e$ and it reports the maximum number of tokens in the places from $X$ in any reachable marking and (iii) *reachability deadlock* where we ask if there is a reachable marking $M$ such that there is no $t \in T$ and no $M'$ where $M \xrightarrow{t} M'$.

We notice that fireability can be encoded as a cardinality query

$$\mathit{fire}(t) \equiv \bigwedge_{p \in {}^\bullet t} p \geq F(p,t) \;\wedge \bigwedge_{p \in I(t)} p = 0$$

and deadlock can be encoded as the cardinality query

$$\mathit{deadlock} \equiv \bigwedge_{t \in T} \neg \mathit{fire}(t)\ .$$

In TAPAAL, we indeed encode reachability fireability queries into the cardinality queries but we use a dedicated deadlock proposition in order to be able to apply structural reductions (see Section 5). The computation of bounds for a given set of places $X$ is done by exploring the whole state-space while still being able to apply some structural reduction rules. Details are discussed in Section 5.

## 3 Explicit Search Algorithm with Heuristic Distance

We shall now describe the explicit search algorithm used in TAPAAL for answering reachability cardinality queries. The search is based on the standard search algorithm using passed/waiting sets (see e.g. [3]) as given in Algorithm 1 but with the important addition of exploring first the markings with the shortest distance to a given cardinality query $\varphi$. The distance $\textsc{Distance}(M, \varphi)$ is computed in Algorithm 2 and it returns a nonnegative integer. If $M \models \varphi$ then the distance function returns 0, otherwise the distance tries to estimate how far away is the marking $M$ from satisfying the query $\varphi$.

This is achieved by first estimating the distance between two integer values w.r.t. a given comparison operator $\bowtie$, as defined by the $\Delta$ function in Algorithm 2. Intuitively, the function $\Delta(v_1, \bowtie, v_2)$ returns the smallest number by

**Algorithm 1** Best-First Reachability Search

```
1: function BEST-FIRST-REACHABILITY-SEARCH(N, M_0, φ)
2:     if M_0 ⊨ φ then
3:         return true
4:     end if
5:     Waiting := {M_0}                                   ▷ Priority queue
6:     Passed := {M_0}                                    ▷ Set of passed markings
7:     while Waiting ≠ ∅ do
8:         M := arg min_{M ∈ Waiting} DISTANCE(M, φ)       ▷ A shortest distance marking
9:         Waiting := Waiting ∖ {M}
10:        for M' such that M -t-> M' where t ∈ T do       ▷ For each successor marking
11:            if M' ∉ Passed then
12:                Passed := Passed ∪ {M'}
13:                if M' ⊨ φ then
14:                    return true                         ▷ Output true and terminate
15:                end if
16:                Waiting := Waiting ∪ {M'}               ▷ Marking M' should be explored
17:            end if
18:        end for
19:    end while
20:    return false                                        ▷ No reachable marking satisfying φ was found
21: end function
```

which either $v_1$ or $v_2$ must be changed in order to make the predicate $v_1 \bowtie v_2$ valid. The basic distance $\Delta$ is then extended to the logical connectives: for conjunction both conjuncts have to hold and hence we add the distances of the conjuncts together, and for disjunction where only one of the disjuncts needs to hold, we take the minimum. The negation is simply propagated down to the atomic predicates using De Morgan's laws.

The heuristics operates very satisfactory in many scenarios as it relies on the assumption that similar markings are likely to be just a few firings away from each other. Nevertheless, in some scenarios the heuristic estimate may degrade the search performance.

We performed a number of experiments comparing the heuristic search strategy against breadth-first-search (BFS) and depth-first-search (DFS) on the competition nets and queries from MCC'15 [11]. We selected a number of hard border-line instances of problems where we still expected to get a reasonable number of conclusive answers for positive reachability queries, resulting in 1296 executions (432 executions for each search strategy). Out of those, we selected models and queries where at least one search strategy found a reachable marking satisfying the given cardinality query and where at least one search strategy took more than 3 seconds (in order to filter out the trivial instances). This resulted in 492 executions (164 for each search strategy) and the results are presented in Figure 1.

**Algorithm 2** Distance Heuristics

```
1: function DISTANCE(M, φ)
2:     if φ = e1 ⋈ e2 then
3:         return Δ(eval(M, e1), ⋈, eval(M, e2))
4:     else if φ = φ1 ∧ φ2 then
5:         return DISTANCE(M, φ1) + DISTANCE(M, φ2)
6:     else if φ = φ1 ∨ φ2 then
7:         return min{DISTANCE(M, φ1), DISTANCE(M, φ2)}
8:     else if φ = ¬(e1 ⋈ e2) then
9:         return Δ(eval(M, e1), ⋈̄, eval(M, e2))
10:    else if φ = ¬(φ1 ∧ φ2) then
11:        return min{DISTANCE(M, ¬φ1), DISTANCE(M, ¬φ2)}
12:    else if φ = ¬(φ1 ∨ φ2) then
13:        return DISTANCE(M, ¬φ1) + DISTANCE(M, ¬φ2)
14:    else if φ = ¬(¬φ1) then
15:        return DISTANCE(M, φ1)
16:    end if
17: end function
```

where $\overline{\bowtie}$ is the dual arithmetical operation of $\bowtie$ (for example $\overline{<}$ is the notation for $\geq$) and where

$$\Delta(v_1, =, v_2) = |v_1 - v_2|$$
$$\Delta(v_1, \neq, v_2) = \begin{cases} 1 & \text{if } v_1 = v_2 \\ 0 & \text{otherwise} \end{cases}$$
$$\Delta(v_1, <, v_2) = \max\{v_1 - v_2 + 1, 0\} \quad \Delta(v_1, >, v_2) = \Delta(v_2, <, v_1)$$
$$\Delta(v_1, \leq, v_2) = \max\{v_1 - v_2, 0\} \quad \Delta(v_1, \geq, v_2) = \Delta(v_2, \leq, v_1)$$

The table shows that the heuristic search was the fastest one in 89 instances, which is more than the sum of cases where BFS or DFS won (75 instances in total). The heuristic strategy timed out in only 19 cases (where either BFS or DFS provided an answer) compared to the large number of runs where BFS and DFS did not find the answer. Finally, the heuristic strategy was in 17 cases the only one that found a marking satisfying the given cardinality query, whereas BFS provided a solo answer in 9 cases and DFS in only 2 cases.

In conclusion, if we use only a single-core for the verification, the heuristic search is preferable, however, in case of more available cores, it may be a good idea to run all three different search strategies independently.

## 4 State Equations for Cardinality Queries

In this section we present an adaptation of the technique based on integer programming (state-equations [12, 13]) which can be in some cases used to efficiently disprove reachability by over-approximating the state-space, hence avoiding the full state-space exploration. Let $N = (P, T, F, I)$ be a PN and let $M_0, M \in \mathcal{M}(N)$ be markings on $N$. If there is a sequence of transitions $w$ such that $M_0 \xrightarrow{w} M$

| Search Strategy | Winner | No. of Timeouts | Solo Answer |
|---|---|---|---|
| Heuristic | 89 | 19 | 17 |
| BFS | 26 | 70 | 9 |
| DFS | 49 | 53 | 2 |

Fig. 1: Heuristic, BFS and DFS search strategies (timeout at 5 minutes)

then a well-known fact (see e.g. [13]) says that there is a nonnegative solution to the following system of equations over the variables $\{x_t \mid t \in T\}$:

$$M_0(p) + \sum_{t \in T} (F(t,p) - F(p,t)) \cdot x_t = M(p) \quad \text{for all } p \in P \ .$$

Clearly, if we set $x_t$ to be the number of times $t$ was fired in the sequence $w$, then this gives us the requested solution. Conversely, if there is no solution to the state-equations then $M$ is not reachable from $M_0$. On the other hand, a solution to the state-equations does not in general imply that $M$ is reachable from the marking $M_0$.

Esparza and Melzer [7] proposed to use integer linear programming in order to solve the state-equations, ensuring that $x_t \in \mathbb{N}_0$ for all $t \in T$ and thus providing a more accurate approximation. We shall generalize this approach to cardinality queries which may require several calls to a linear program solver. A *restriction* is a function $r : P \to \mathbb{N}_0 \times (\mathbb{N}_0 \cup \{\infty\})$ from places to right-open intervals representing the allowed number of tokens in each of the places (if $r(p) = [0, \infty]$ then there is no restriction on the number of tokens in $p$). Given two restrictions $r_1$ and $r_2$, we introduce the combined restriction $\mathit{combine}(r_1, r_2)$ defined as $\mathit{combine}(r_1, r_2)(p) = r_1(p) \cap r_2(p)$ where we assume here the standard interval intersection operator. We use the notation $\langle p_1 \mapsto [a_1, b_1], \ldots p_n \mapsto [a_n, b_n] \rangle$ to represent a restriction $r$ such that $r(p_1) = [a_1, b_1]$, $\ldots$, $r(p_n) = [a_n, b_n]$ and $r(p) = [0, \infty]$ for all $p \in P \setminus \{p_1, \ldots, p_n\}$. For example, $\mathit{combine}(\langle p \mapsto [2, \infty], q \mapsto [2, 10] \rangle, \langle p \mapsto [0, 7] \rangle) = \langle p \mapsto [2, 7], q \mapsto [2, 10] \rangle$.

Let us now define the function *constraints* that for a given cardinality query $\varphi$ returns a set of restrictions. For simplicity, we assume that the negation has already been pushed (using De Morgan rules) all the way to the atomic propositions where the negation can be replaced by the dual atomic propositions.

$$\begin{aligned}
\mathit{constr}(p = n) &= \{\langle p \mapsto [n, n] \rangle\} \\
\mathit{constr}(p \neq n) &= \{\langle p \mapsto [0, n-1] \rangle, \langle p \mapsto [n+1, \infty] \rangle\} \\
\mathit{constr}(p \leq n) &= \{\langle p \mapsto [0, n] \rangle\} \\
\mathit{constr}(p \geq n) &= \{\langle p \mapsto [n, \infty] \rangle\} \\
\mathit{constr}(p < n) &= \{\langle p \mapsto [0, n-1] \rangle\} \\
\mathit{constr}(p > n) &= \{\langle p \mapsto [n+1, \infty] \rangle\}
\end{aligned}$$

$$\begin{aligned}
\mathit{constr}(\varphi_1 \vee \varphi_2) &= \mathit{constr}(\varphi_1) \cup \mathit{constr}(\varphi_2) \\
\mathit{constr}(\varphi_1 \wedge \varphi_2) &= \{\mathit{combine}(r_1, r_2) \mid r_1 \in \mathit{constr}(\varphi_1), r_2 \in \mathit{constr}(\varphi_2)\}
\end{aligned}$$

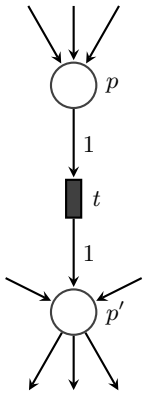
**Algorithm 3** Disproving Reachability Using Integer Programming

**function** DISPROVE-REACHABILITY($N, M_0, \varphi$)
  Let $N = (P, T, F, I)$.
  **for all** $r \in constr(\varphi)$ **do**
    $LP := \emptyset$ ▷ Let $LP$ be an empty system of inequations
    **for all** $p \in P$ **do**
      Let $[min, max] = r(p)$.
      $LP := LP \cup \{M_0(p) + \sum_{t \in T} (F(t,p) - F(p,t)) \cdot x_t \geq min\}$
      $LP := LP \cup \{M_0(p) + \sum_{t \in T} (F(t,p) - F(p,t)) \cdot x_t \leq max\}$
    **end for**
    **if** LP has an integer solution **then**
      **return** "Inconclusive"
    **end if**
  **end for**
  **return** "$M \not\models EF\,\varphi$"
**end function**

The actual use of state-equations in the setting of cardinality queries is now described in Algorithm 3.

Our implementation of the algorithm uses lpsolve [2] for the linear programming part and performs fast on most of the competition nets. We have selected two smallest instances of each scalable model from the known models used in MCC'15 in order to be able to make a full state-space search on most of these models for the purpose of our analysis. Then we ran the state-equation test for all cardinality queries, resulting in the total number of 1024 executions. If the over-approximation using state-equations succeeded (disproved reachability), we report this and terminate, otherwise we continue with the state-space search using the heuristic strategy with 5 minutes timeout. In 125 runs we did not get a conclusive answer and reached the timeout, in 405 runs the answer was negative (cardinality query was not reachable) and in the remaining 494 cases the query was reachable. Out of the 405 runs where the cardinality query was disproved, the state-equation technique succeeded in 118 cases (and hence the expensive state-space search was completely avoided). Moreover, it took on average only 0.15 seconds to perform the state-equation check, with only four tests exceeding 2 seconds. The most expensive over-approximation test was for the model PolyORBNT-S05J30 where it took 4.25 seconds.

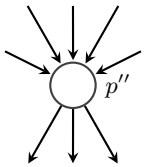
The over-approximation using state-equations is a fast and efficient method to disprove the reachability of cardinality queries and it manages in almost 30% of cases to provide a conclusive answer. In order to further increase the percentage of cases with conclusive answers, we plan to experiment with trap reduction [7] and other techniques in order to make the technique applicable to even more cardinality queries.




Conditions on $p$, $t$ and $p'$:

1. $p \neq p'$
2. $p^\bullet = \{t\}$, $^\bullet t = \{p\}$, $t^\bullet = \{p'\}$
3. $F(p,t) = F(t,p') = 1$
4. $M_0(p) = 0$ or $M_0(p') = 0$
5. $I(t) = I(p) = I(p') = \emptyset$
6. $p, p' \notin \mathit{places}(\varphi)$

$\Rightarrow$

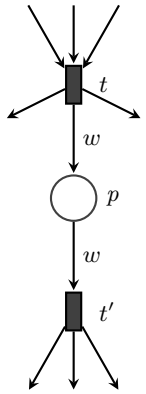



$$p'' := \begin{cases} p & \text{if } M_0(p') = 0 \\ p' & \text{otherwise} \end{cases}$$

Remove $t$ and $p'$ (if $M_0(p') = 0$) resp. $p$ (otherwise).

For all $t' \in T \setminus \{t\}$:
$F(t', p'') := F(t', p) + F(t', p')$
$F(p'', t') := F(p, t') + F(p', t')$

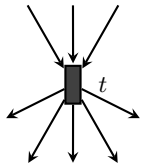
(a) Sequential transition removal




Conditions on $t$, $p$ and $t'$:

1. $t \neq t'$
2. $^\bullet p = \{t\}$, $p^\bullet = \{t'\}$, $^\bullet t' = \{p\}$
3. $F(t,p) = F(p,t') = w > 0$
4. $M_0(p) = 0$
5. $I(p) = I(t) = I(t') = \emptyset$
6. $p \notin \mathit{places}(\varphi)$
7. $I(p') = \emptyset$ for all $p' \in t'^\bullet$
8. $p' \notin \mathit{places}(\varphi)$ for all $p' \in t'^\bullet$

$\Rightarrow$




Remove $p$ and $t'$.

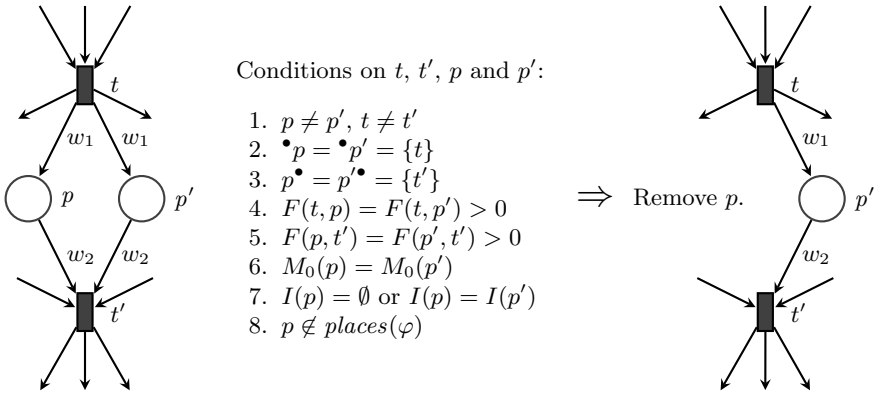
For all $p' \in P$:
$F(t, p') := F(t, p') + F(t', p')$

(b) Sequential place removal

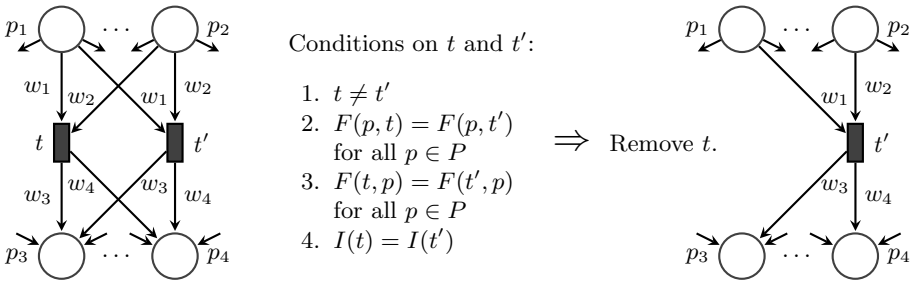
Fig. 2: Sequential rules for a cardinality formula $\varphi$ and initial marking $M_0$

## 5 Structural Reductions

We shall now present a set of structural reduction rules that allow us to reduce the net structure and decrease the size of the state-space, while preserving the answers to cardinality queries. The classical reduction rules for preserving liveness, safeness and boundedness were introduced in [14, 13]. We extend them to weighted nets with inhibitor arcs and specialize to the use for cardinality queries. The extension is not completely straightforward as a number of side conditions must be satisfied in order to preserve correctness—in fact TAPAAL was the only tool at MCC'15 that used structural reduction techniques. The rules are presented in Figures 2 and 3 and they are relative to a given initial marking $M_0$ and a cardinality query $\varphi$, where $\mathit{places}(\varphi)$ is the set of all places that occur in the query $\varphi$.




(a) Parallel place removal




(b) Parallel transition removal

Fig. 3: Parallel rules for a cardinality formula $\varphi$ and initial marking $M_0$

**Theorem 1.** *Let $(N, M_0)$ be a marked Petri net and let $\varphi$ be a cardinality query. Let $N'$ be the net $N$ after the application of some reduction rules from Figures 2 and 3. Then $(N, M_0) \models EF\,\varphi$ if and only if $(N', M_0) \models EF\,\varphi$.*

*Proof.* As cardinality queries are only concerned about the number of tokens in places, it is easy to see that the parallel transition rule in Figure 3b is harmless as the transitions $t$ and $t'$ are enabled at the same time and they have the same firing effect, so we can easily remove one of them without affecting the reachable markings. Similarly, the parallel places rule in Figure 3a ensures that the number of tokens in $p$ and $p'$ remain the same in any reachable marking (ensured by the assumption that $p$ and $p'$ contain the same number of tokens already in the initial marking). Now we can remove the place $p$, provided that $p$ is not used in the cardinality query $\varphi$ and either there are no inhibitor arcs connected to $p$ or the places $p$ and $p'$ inhibit exactly the same set of transitions.

For a given net $N$, let $N'$ be a net after one application of the sequential transition rule in Figure 2a that removed the transition $t$. We shall first argue that if $(N, M_0) \models EF\,\varphi$, meaning that $M_0 \xrightarrow{w} M$ for some sequence of transitions $w$ such that $M \models \varphi$, then also $(N', M_0) \models EF\,\varphi$. To show this, let $w'$ be the

transition sequence obtained from $w$ by removing all occurrences of the transition $t$. Observe now that due to the fact that no inhibitor arcs are connected to $p$ and $p'$ (condition 5), we can execute from $M_0$ in $N'$ the sequence $w'$ ($M_0$ is a valid marking also in $N'$ due to condition 4 requiring that the place we removed in $N'$ has no tokens in $M_0$) and obtain a marking $M'$ such that $M'(\overline{p}) = M(\overline{p})$ for all $\overline{p} \in P \setminus \{p, p'\}$ and $M'(p'') = M(p) + M(p')$. As the query $\varphi$ does not contain the places $p$ and $p'$ (condition 6), we can conclude that also $M' \models \varphi$ and hence $(N', M_0) \models EF\,\varphi$. For the opposite direction, assume that $M_0 \xrightarrow{w} M'$ in the net $N'$ such that $M' \models \varphi$. We shall now fire this transition sequence $w$ in the original net $N$ such that whenever the transition $t$ that was removed in $N'$ is enabled, we insert its firing into the sequence $w$ as long as it is enabled. This will guarantee that all tokens from $p$ are moved to $p'$ due to the requirement that the single input and output arcs of $t$ have weight 1 (conditions 2 and 3) and that $t$ is not connected with any inhibitor arcs (condition 5). As $p$ is not an input place for any other transition than $t$ (condition 2), moving the tokens from $p$ to $p'$ does not influence the firing of other transitions in $N$. Similarly, the configuration of tokens in $p$ and $p'$ cannot influence the firing of other transitions in $N'$ due to the absence of inhibitor arcs connected to $p$ and $p'$ (condition 5). Now, let $M$ be the marking reached in $N$ after firing the sequence of transitions described above. Clearly, $M(\overline{p}) = M'(\overline{p})$ for all $\overline{p} \in P \setminus \{p, p'\}$ and as $\varphi$ is not referring to the places $p$ and $p'$ (condition 6), we get $M \models \varphi$ implying that $(N, M_0) \models EF\,\varphi$.

The arguments for the rule in Figure 2b, omitted due to space limitations, are analogous to the sequential transition removal rule discussed above. □

Note that the more places occur in the query $\varphi$, the fewer reduction rules are in general applicable. The reachability of a deadlock can be expressed using a cardinality query but then all places connected to some transition will be mentioned in the query and hence the structural reduction rules will not be applicable. However, for deadlock we can reduce the net w.r.t. some trivial query that does not contain any places (e.g. $EF\,2 < 1$) and now $(N, M_0)$ is deadlock-free if and only if $(N', M_0)$ is deadlock-free.

**Theorem 2.** *Let $(N, M_0)$ be a marked Petri net. Let $N'$ be the net $N$ after the application of some reduction rules from Figures 2 and 3 for a query $\varphi = 2 < 1$. Then $(N, M_0)$ has a deadlock if and only if $(N', M_0)$ has a deadlock.*

*Proof.* The proof is very similar to the proof of Theorem 1 but some of the additional conditions like the requirement $p \neq p'$ in the rule from Figure 2a (condition 1) are important as removing the transition $t$ in case of $p = p'$ can create a new deadlock in $N'$ that is not present in $N$. □

For the competition queries that ask to compute the maximum number of tokens in the net, we may only use reduction rules from Figure 2a and 3b as the other two rules possibly decrease the maximum number of reachable tokens.

We have conducted experiments on the same nets as in Section 4 in order to see how many nets can be reduced and to what degree. The reductions were performed relative to a query that does not contain any places (as e.g. deadlock)

in order to see the maximal possible reduction. If a query contains many places, the number of applications of the reduction rules may be possibly lower. The data show that out of the 261 nets, 118 of them were reducible, with an average reduction of 35% of the net size (measured as the number of places plus the number of transitions). Some nets are reducible by only a few percent while others allow a reduction of up to 95% (e.g. the house construction net). As reducing the size of a net can imply up to an exponential decrease in the size of the state-space, the effect of the reductions significantly contributes to the performance of our verification engine.

## 6 Tool Implementation

The verification engine for P/T nets, employing the techniques described in earlier sections, has been efficiently implemented in C++ and made publicly available as a part of the tool suite TAPAAL [4]. It includes a GUI for drawing the nets, graphical query creation dialog and advanced debugging (simulation) options. The tool allows us to import the MCC competition nets in PNML format as well as the cardinality and deadlock queries, and process them either individually or in a batch processing mode.

Regarding the implementation details, our experiments showed that the incidence matrix representation of a Petri net is preferred over the linked list representation as even though on larger nets the linked list representation preserves some space, it is remarkably slower [6] (likely due to the cache coherence issues). Finally, it is important to remark that for larger nets with several hundreds of places and transitions, an efficient implementation of the structural reduction rules is of great importance as a naive coding of the rules using up to four nested loops (like the rule in Figure 3a) will use too much of the preprocessing time.

## 7 Conclusion

We described the most essential verification techniques used in the P/T net engine of TAPAAL. Each of the techniques has a significant performance effect, as documented by a number of experiments run on the nets and queries from MCC'15. We believe that it is the combination of these techniques and a relatively simple explicit search engine that contributed to the second place of our tool in the years 2014 and 2015. We are currently working on optimizing the performance of the successor generator, space optimizations and extending the reachability analysis to the full CTL model checking.

*Acknowledgments.* The fourth author is partially affiliated with FI MU, Brno, Czech Republic.

## References


[1] M. Andersen, H.G. Larsen, J. Srba, M.G. Sørensen, and J.H. Taankvist. Verification of liveness properties on closed timed-arc Petri nets. In *MEMICS'12*, volume 7721 of *LNCS*, pages 69–81. Springer-Verlag, 2013.

[2] M. Berkelaar, K. Eikland, and P. Notebaert. lp_solve 5.5, open source (mixed-integer) linear programming system. Software, May 1 2004. Available at http://lpsolve.sourceforge.net/5.5.

[3] A. David, G. Behrmann, K.G. Larsen, and W. Yi. A tool architecture for the next generation of Uppaal. In *Formal Methods at the Crossroads. From Panacea to Foundational Support*, volume 2757 of *LNCS*, pages 352–366. Springer, 2003.

[4] A. David, L. Jacobsen, M. Jacobsen, K.Y. Jørgensen, M.H. Møller, and J. Srba. TAPAAL 2.0: Integrated development environment for timed-arc Petri nets. In *TACAS'12*, volume 7214 of *LNCS*, pages 492–497. Springer-Verlag, 2012.

[5] A. David, L. Jacobsen, M. Jacobsen, and J. Srba. A forward reachability algorithm for bounded timed-arc Petri nets. In *SSV'12*, volume 102 of *EPTCS*, pages 125–140. Open Publishing Association, 2012.

[6] J. Dyhr, M. Johannsen, I. Kaufmann, and S.M. Nielsen. Multi-core model checking of Petri nets with precompiled successor generation. Bacherol thesis. Department of Computer Science, Aalborg University, Denmark., 2015.

[7] J. Esparza and S. Melzer. Verification of safety properties using integer programming: Beyond the state equation. *Form. Meth. in Syst. Design*, 16:159–189, 2000.

[8] F. Heitmann and D. Moldt. Petri nets tool database. http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/db.html, 2015.

[9] L. Jacobsen, M. Jacobsen, M.H. Møller, and J. Srba. Verification of timed-arc Petri nets. In *SOFSEM'11*, volume 6543 of *LNCS*, pages 46–72. Springer, 2011.

[10] F. Kordon, H. Garavel, L-M. Hillah, F. Hulin-Hubard, A. Linard, M. Beccuti, S. Evangelista, A. Hamez, N. Lohmann, E. Lopez, E. Paviot-Adet, C. Rodriguez, C. Rohr, and J. Srba. HTML results from the Model Checking Contest @ Petri Net (2014 edition). http://mcc.lip6.fr/2014, 2014.

[11] F. Kordon, H. Garavel, L. M. Hillah, F. Hulin-Hubard, A. Linard, M. Beccuti, A. Hamez, E. Lopez-Bobeda, L. Jezequel, J. Meijer, E. Paviot-Adet, C. Rodriguez, C. Rohr, J. Srba, Y. Thierry-Mieg, and K. Wolf. Complete Results for the 2015 Edition of the Model Checking Contest. http://mcc.lip6.fr/2015/, 2015.

[12] T. Murata. State equation, controllability, and maximal matching of Petri nets. *IEEE Transactions on Automatic Control*, 22(3):412–416, 1977.

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

[14] T. Murata and J.Y. Koh. Reduction and expansion of live and safe marked graphs. *IEEE Transactions on Circuits and Systems*, 27(1):68–70, 1980.

[15] C.A. Petri. *Kommunikation mit Automaten*. PhD thesis, Darmstadt, 1962.