# SAT-Based (Parametric) Reachability for Time Petri Nets *

Wojciech Penczek[1,2], Agata Półrola[4], and Andrzej Zbrzezny[3]

[1] Institute of Computer Science, PAS, Ordona 21, 01-237 Warsaw, Poland
[2] Institute of Informatics, Podlasie Academy, Sienkiewicza 51, 08-110 Siedlce, Poland
penczek@ipipan.waw.pl
[3] Jan Długosz University, IMCS, Armii Krajowej 13/15, 42-200 Częstochowa, Poland
[4] University of Łódź, FMCS, Banacha 22, 90-238 Łódź, Poland

**Abstract.** Formal methods - among them the model checking techniques - play an important role in the design and production of both systems and software. In this paper we deal with an adaptation of the bounded model checking methods for timed systems, developed for timed automata, to the case of time Petri nets. We focus on distributed time Petri nets and parametric reachability checking, but the approach can be easily adapted to verification of other kinds of properties for which the bounded model checking methods exist. A theoretical description is supported by some experimental results, generated using an extension of the model checker VerICS.

## 1 Introduction

The process of design and production of both systems and software – among others, the concurrent ones – involves testing whether the product conforms its specification. To this aim, various kinds of formal methods can be applied. One of the possible approaches, widely used and intensively developed, are *model checking techniques*.

In order to perform a formal verification, the system to be tested is usually modelled using a theoretical formalism, e.g., a version of automata, Petri nets, state diagrams etc. Obviously, the kind of the formalism depends on the features of the system to be described. One of the approaches, used to represent concurrent systems with timing dependencies [10, 11, 20], are *time Petri nets* (TPNs) by Merlin and Farber [21]. After modelling the system in the above way, a suitable verification method is applied.

The main problem to cope with while verifying timed systems is the so-called *state explosion*: in order to check whether the system satisfies a property we usually need to search through its state space, which in most cases is very large due to infinity of the dense time domain. Furthermore, in the case of concurrent systems the size of the state space is used to grow exponentially when the number of the components increases. So, searching for verification methods which are able to overcome the above problem is an important subject of research.

* Partly supported by the Polish Ministry of Science and Higher Education under the grant No. N N206 258035

Bounded Model Checking (BMC) is an efficient verification technique whose main idea consists in translating a model checking problem solvable on a fraction of a model into a test of propositional satisfiability, which is then made using a SAT-checker. The method has been successfully applied to verification of both timed and untimed systems [3, 4, 7, 12, 16, 27, 31, 35]. In this paper, we show how to adapt the BMC methods, presented in [27, 34–36] and developed for timed automata, to the case of time Petri nets. The adaptation exploits, in some sense, a method of translating a time Petri net to a timed automaton, described in [28]. However, we perform no structural translation between these two formalisms, but use directly the transition relation defined by the translation. In order to benefit from the concurrent structure of the system, we focus on *distributed* nets (i.e., sets of communicating processes), and exploit a non-standard approach to their concrete semantics, which consist in associating a clock with each of the processes [28]. In this work, we deal with testing whether the system (net) can ever be in a state satisfying certain properties (i.e., with *reachability* checking), but the presented solutions can be also easily adapted to verification of other classes of properties for which BMC methods exist (see [23] for a survey). The algorithm has been implemented as an extension of the model checker VerICS [13]. The next topic we dealt with was searching for bounds on which the property tested can be reached (searching for a value of the parameter $c$ in formulas $\mathrm{EF}^{\sim c}p$, corresponding to these considered in [14]). In the final part of the paper we provide some preliminary experimental results.

To our knowledge, no BMC method for time Petri nets has been defined so far, although some solutions for untimed Petri nets exist [16, 25]. Therefore, the main contribution of this work consists in showing how to apply and implement for TPNs the above technique of verification (a general idea of the approach has been already sketched in [23], but no details are given there). As a result, we obtain an efficient method of checking reachability, as well as searching for counterexamples for the properties expressible by formulas of the logics $\mathrm{ACTL}^*$ and TACTL. Although the adaptation of the BMC methods is almost straightforward, the practical consequences seem to be quite useful.

The rest of the paper is organised as follows: in Sect. 3 we introduce time Petri nets, and the abstraction of their state spaces, i.e., an *extended detailed region graphs*. In the further part we sketch the idea of reachability checking using BMC (Sect. 4), and show its implementation for time Petri nets (Sect. 5). Searching for bounds on time at which a state satisfying a property can be reached (parametric reachability) is considered in Sect. 6. Sections 7 and 8 contain experimental results and concluding remarks.

## 2 Related work

The methods of reachability checking for time Petri nets, mostly consisting in building an *abstract model* of the system, are widely studied in the literature [6, 5, 8, 9, 15, 19]. Detailed region graphs for time Petri nets, based on their standard semantics (i.e., the one associating a clock with each transition of the net) were presented in [22, 33]. Some BMC methods for (untimed) Petri Nets were described in [16, 26]. Parametric verification for time Petri nets was considered in [32].

The current work is a modification and extension of the paper [24] (published in proceedings of a local workshop with the status of a technical report).

## 3 Time Petri Nets

Let $\mathbb{R}_+$ denote the set of non-negative reals, $\mathbb{Q}$ the set of rationals, and $\mathbb{N}$ ($\mathbb{N}_+$) - the set of (positive) natural numbers. We start with a definition of time Petri nets:

**Definition 1.** *A time Petri net (TPN, for short) is a six-element tuple* $\mathcal{N} = (P, T, F, m^0, Eft, Lft)$*, where* $P = \{p_1, \ldots, p_{n_P}\}$ *is a finite set of* places*,* $T = \{t_1, \ldots, t_{n_T}\}$ *is a finite set of* transitions*,* $F \subseteq (P \times T) \cup (T \times P)$ *is the* flow relation*,* $m^0 \subseteq P$ *is the* initial marking *of* $\mathcal{N}$*, and* $Eft : T \to \mathbb{N}$*,* $Lft : T \to \mathbb{N} \cup \{\infty\}$ *are functions describing the* earliest *and the* latest firing time *of the transition; where for each* $t \in T$ *we have* $Eft(t) \leq Lft(t)$*.*

For a transition $t \in T$ we define its *preset* $\bullet t = \{p \in P \mid (p, t) \in F\}$ and *postset* $t\bullet = \{p \in P \mid (t, p) \in F\}$, and consider only the nets such that for each transition the preset and the postset are non-empty. We need also the following notations and definitions:

- a *marking* of $\mathcal{N}$ is any subset $m \subseteq P$;
- a transition $t \in T$ is *enabled* at $m$ ($m[t\rangle$ for short) if $\bullet t \subseteq m$ and $t\bullet \cap (m \setminus \bullet t) = \emptyset$; and *leads from* $m$ *to* $m'$*,* if it is enabled at $m$, and $m' = (m \setminus \bullet t) \cup t\bullet$. The marking $m'$ is denoted by $m[t\rangle$ as well, if this does not lead to misunderstanding;
- $en(m) = \{t \in T \mid m[t\rangle\}$ is the set of all the transitions enabled at the marking $m$ of $\mathcal{N}$;
- a marking $m \subseteq P$ is *reachable* if there exists a sequence of transitions $t_1, \ldots, t_l \in T$ and a sequence of markings $m_0, \ldots, m_l$ such that $m_0 = m^0$, $m_l = m$, and for each $i \in \{1, \ldots, l\}$ $t_i \in en(m_{i-1})$ and $m_i = m_{i-1}[t_i\rangle$;
- a marking $m$ *concurrently enables* two transitions $t, t' \in T$ if $t \in en(m)$ and $t' \in en(m \setminus \bullet t)$;
- a net is *sequential* if no reachable marking of $\mathcal{N}$ concurrently enables two transitions.

It should be mentioned that the time Petri nets defined as above are often called *1-safe* in the literature.

Next, we introduce the notion of a *distributed time Petri net*. The definition is an adaptation of the one from [17]:

**Definition 2.** *Let* $\mathcal{I} = \{\mathfrak{i}_1, \ldots, \mathfrak{i}_n\}$ *be a finite ordered set of indices, and let* $\mathfrak{N} = \{N_\mathfrak{i} \mid \mathfrak{i} \in \mathcal{I}\}$*, where* $N_\mathfrak{i} = (P_\mathfrak{i}, T_\mathfrak{i}, F_\mathfrak{i}, m^0_\mathfrak{i}, Eft_\mathfrak{i}, Lft_\mathfrak{i})$ *be a family of 1-safe, sequential time Petri nets (called* processes*), indexed with* $\mathcal{I}$*, with the pairwise disjoint sets* $P_\mathfrak{i}$ *of places, and satisfying the condition* $(\forall \mathfrak{i}_1, \mathfrak{i}_2 \in \mathcal{I})(\forall t \in T_{\mathfrak{i}_1} \cap T_{\mathfrak{i}_2})\ (Eft_{\mathfrak{i}_1}(t) = Eft_{\mathfrak{i}_2}(t) \ \wedge\ Lft_{\mathfrak{i}_1}(t) = Lft_{\mathfrak{i}_2}(t))$*. A* distributed time Petri net $\mathcal{N} = (P, T, F, m^0, Eft, Lft)$ *is the union of the* processes $N_\mathfrak{i}$*, i.e.,* $P = \bigcup_{\mathfrak{i} \in \mathcal{I}} P_\mathfrak{i}$, $T = \bigcup_{\mathfrak{i} \in \mathcal{I}} T_\mathfrak{i}$, $F = \bigcup_{\mathfrak{i} \in \mathcal{I}} F_\mathfrak{i}$, $m^0 = \bigcup_{\mathfrak{i} \in \mathcal{I}} m^0_\mathfrak{i}$, $Eft = \bigcup_{\mathfrak{i} \in \mathcal{I}} Eft_\mathfrak{i}$*, and* $Lft = \bigcup_{\mathfrak{i} \in \mathcal{I}} Lft_\mathfrak{i}$*.*

Notice that the function $Eft_{\mathfrak{i}_1}$ ($Lft_{\mathfrak{i}_1}$) coincides with $Eft_{\mathfrak{i}_2}$ ($Lft_{\mathfrak{i}_2}$, resp.) for the joint transitions of each two processes $\mathfrak{i}_1$ and $\mathfrak{i}_2$. The interpretation of such a system is a collection of sequential, non-deterministic processes with communication capabilities (via joint transitions).

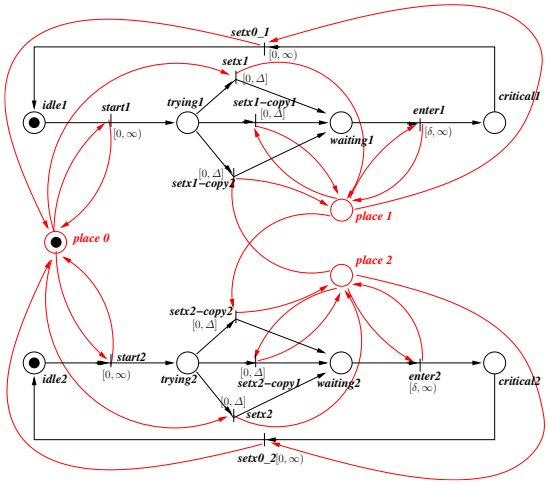


**Fig. 1.** A net for Fischer's mutual exclusion protocol for $n = 2$

An example of a distributed TPN (Fischer's mutual exclusion protocol) is shown in Fig. 1. The net consists of three communicating processes with the sets of places $P_i = \{idle_i, trying_i, enter_i, critical_i\}$ for $i = 1, 2$, and $P_3 = \{place0, place1, place2\}$. All the transitions of the process $N_1$ and all the transitions of the process $N_2$ are joint with the process $N_3$.

In what follows, we consider distributed nets only, and assume that all their processes are *state machines* (i.e., for each $\mathfrak{i} \in \mathcal{I}$ and each $t \in T_\mathfrak{i}$, $|\bullet t| = |t \bullet| = 1$), which implies that in any marking of $\mathcal{N}$, there is exactly one place of each process. It is important to mention that a large class of distributed nets can be decomposed to satisfy the above requirement [18]. Moreover, for $t \in T$ we define $I\mathcal{V}(t) = \{\mathfrak{i} \in \mathcal{I} \mid \bullet t \cap P_\mathfrak{i} \neq \emptyset\}$, and say that a process $N_\mathfrak{i}$ is *involved in a transition* $t$ iff $\mathfrak{i} \in I\mathcal{V}(t)$.

### 3.1 Concrete State Spaces and Models

The current state of the net is given by its marking and the time passed since each of the enabled transitions became enabled (which influences the future behaviour of the net). Thus, a *concrete state* $\sigma$ of $\mathcal{N}$ can be defined as an ordered pair $(m, clock)$, where $m$ is a marking, and $clock : \mathcal{I} \to \mathbb{R}_+$ is a function which for each index $\mathfrak{i}$ of a process of $\mathcal{N}$ gives the time elapsed since the marked place of this process became marked most recently [28]. The set of all the concrete states is denoted by $\Sigma$. The initial state of $\mathcal{N}$ is $\sigma^0 = (m^0, clock^0)$, where $m^0$ is the initial marking, and $clock^0(\mathfrak{i}) = 0$ for each $\mathfrak{i} \in \mathcal{I}$.

For $\delta \in \mathbb{R}_+$, let $clock + \delta$ denote the function given by $(clock + \delta)(\mathfrak{i}) = clock(\mathfrak{i}) + \delta$, and let $(m, clock) + \delta$ denote $(m, clock + \delta)$. The states of $\mathcal{N}$ can change when the time passes or a transition fires. In consequence, we introduce a labelled timed consecution relation $\rightarrow_c \subseteq \Sigma \times (T \cup \mathbb{R}_+) \times \Sigma$ given as follows:

– In a state $\sigma = (m, clock)$ a time $\delta \in \mathbb{R}_+$ can pass leading to a new state $\sigma' = (m, clock' + \delta)$ (denoted $\sigma \xrightarrow{\delta}_c \sigma'$) iff for each $t \in en(m)$ there exists $\mathfrak{i} \in I\mathcal{V}(t)$ such that $clock(\mathfrak{i}) + \delta \leq Lft(t)$ (*time-successor relation*);
– In a state $\sigma = (m, clock)$ a transition $t \in T$ can fire leading to a new state $\sigma' = (m', clock')$ (denoted $\sigma \xrightarrow{t}_c \sigma'$) if $t \in en(m)$, for each $\mathfrak{i} \in I\mathcal{V}(t)$ we have $clock(\mathfrak{i}) \geq Eft(t)$, and there is $\mathfrak{i} \in I\mathcal{V}(t)$ such that $clock(\mathfrak{i}) \leq Lft(t)$. Then, $m' = m[t\rangle$, and for all $\mathfrak{i} \in \mathcal{I}$ we have $clock'(\mathfrak{i}) = 0$ if $\mathfrak{i} \in I\mathcal{V}(t)$, and $clock'(\mathfrak{i}) = clock(\mathfrak{i})$ otherwise (*action-successor relation*).

Intuitively, the time-successor relation does not change the marking of the net, but increases the clocks of all the processes, provided that no enabled transition becomes disabled by passage of time (i.e., for each $t \in en(m)$ the clock of at least one process involved in the transition does not exceed $Lft(t)$). Firing of a transition $t$ takes no time - the action-successor relation does not increase the clocks, but only sets to zero the cloks of the involved processes (note that each of these processes contains exactly one input and one output place of $t$, as the processes are state machines); and is allowed provided that $t$ is enabled, the clocks of all the involved processes are greater than $Eft(t)$, and there is at least one such process whose clock does not exceed $Lft(t)$.

Then, we define a *timed run* of $\mathcal{N}$ starting at a state $\sigma_0 \in \Sigma$ ($\sigma_0$-*run*) as a maximal sequence of concrete states, transitions and time passings $\rho = \sigma_0 \xrightarrow{\delta_0}_c \sigma_0 + \delta_0 \xrightarrow{t_0}_c \sigma_1 \xrightarrow{\delta_1}_c \sigma_1 + \delta_1 \xrightarrow{t_1}_c \sigma_2 \xrightarrow{\delta_2}_c \ldots$, where $\sigma_i \in \Sigma$, $t_i \in T$ and $\delta_i \in \mathbb{R}_+$ for all $i \in \mathbb{N}$. A state $\sigma_* \in \Sigma$ is *reachable* if there exists a $\sigma^0$-run $\rho$ and $i \in \mathbb{N}$ such that $\sigma_* = \sigma_i + \delta_i$, where $\sigma_i + \delta_i$ is an element of $\rho$. The set of all the reachable states of $\mathcal{N}$ is denoted by $Reach_{\mathcal{N}}$.

Given a set of propositional variables $PV$, we introduce a *valuation function* $V_c : \Sigma \to 2^{PV}$ which assigns the same propositions to the states with the same markings. We assume the set $PV$ to be such that each $q \in PV$ corresponds to exactly one place $p \in P$, and use the same names for the propositions and the places. The function $V_c$ is defined by $p \in V_c(\sigma) \Leftrightarrow p \in m$ for each $\sigma = (m, \cdot)$. The structure $M_c(\mathcal{N}) = ((T \cup \mathbb{R}_+, \Sigma, \sigma^0, \to_c), V_c)$ is called a *concrete* (*dense*) *model of* $\mathcal{N}$. It is easy to see that concrete models are usually infinite.

### 3.2 Extended Detailed Region Graph

In order to deal with countable structures instead of uncountable ones, we introduce *extended detailed region graphs* for distributed TPNs. They correspond to the well-known graphs defined for timed automata in [1] and adapted for time Petri nets [22, 33], but involve disjunctions of constraints, the reflexive transitive closure of the time successor of [1], and make no use of the maximal constant appearing in the invariants and enabling conditions. To do this, we assign a clock to each of the processes of a net.

Given a distributed time Petri net $\mathcal{N}$ whose processes are indexed with a set of indices $\mathcal{I}$ with $|\mathcal{I}| = n$ for some $n \in \mathbb{N}_+$. Let $\mathcal{X} = \{x_1, \ldots, x_n\}$ be a finite set of real-valued variables, called *clocks*. A *clock valuation* on $\mathcal{X}$ is a $n$-tuple $v \in \mathbb{R}_+^n$. The value of a clock $x_\mathfrak{i}$ in $v$ is denoted by $v(x_\mathfrak{i})$. For a valuation $v$ and a subset of clocks $X \subseteq \mathcal{X}$, by $v[X := 0]$ we denote the valuation $v'$ such that $v'(x) = 0$ for all $x \in X$,

and $v'(x) = v(x)$ for all $x \in \mathcal{X} \setminus X$. Moreover, for some $\delta \in \mathbb{R}_+$, by $v + \delta$ we denote the valuation $v'$ such that $v'(x) = v(x) + \delta$ for all $x \in \mathcal{X}$. The set $\mathcal{C}_\mathcal{X}$ of *clock constraints* over $\mathcal{X}$ is defined by the following grammar:

$$\mathfrak{cc} := \ true \mid x_\mathfrak{i} \sim c \mid \mathfrak{cc} \wedge \mathfrak{cc} \mid \mathfrak{cc} \vee \mathfrak{cc},$$

where $x_\mathfrak{i} \in \mathcal{X}$, $\sim \in \{\leq, <, =, >, \geq\}$ and $c \in \mathbb{N}$. A valuation $v$ *satisfies* a constraint $\mathfrak{cc} \in \mathcal{C}_\mathcal{X}$ (denoted $v \models \mathfrak{cc}$) iff

- $\mathfrak{cc}$ is of the form $true$,
- $v(x_\mathfrak{i}) \sim c$, and $\mathfrak{cc}$ is of the form $x_\mathfrak{i} \sim c$,
- $v \models \mathfrak{cc}_1 \ \wedge \ v \models \mathfrak{cc}_2$, and $\mathfrak{cc}$ is of the form $\mathfrak{cc}_1 \wedge \mathfrak{cc}_2$,
- $v \models \mathfrak{cc}_1 \ \vee \ v \models \mathfrak{cc}_2$, and $\mathfrak{cc}$ is of the form $\mathfrak{cc}_1 \vee \mathfrak{cc}_2$.

The set of clock valuations satisfying a given constraint $\mathfrak{cc}$ is denoted by $[\![\mathfrak{cc}]\!]$ ($[\![\mathfrak{cc}]\!] \subseteq \mathbb{R}^n_+$).

We assume the clock valuations to be such that for any concrete state $\sigma = (m, clock)$, for each $\mathfrak{i} \in \mathcal{I}$ we have $v(x_\mathfrak{i}) = clock(\mathfrak{i})$. Thus, the clock constraint expressing the conditions under which the net can be in a marking $m$ (the *marking invariant*) can be written as

$$inv(m) = \bigwedge_{t \in en(m) \text{ s.t. } Lft(t) < \infty} \ \bigvee_{\mathfrak{i} \in I\mathcal{V}(t)} x_\mathfrak{i} \leq Lft(t),$$

if $\{t \in T \mid t \in en(m) \wedge Lft(t) < \infty\} \neq \emptyset$, and as $inv(m) = true$ otherwise, which intuitively means that staying in $m$ is allowed as long as for each enabled transition $t$ with finite latest firing time there is a process $N_\mathfrak{i}$, involved in this transition, whose clock is not greater than $Lft(t)$ (and therefore $t$ has not been disabled by passage of time). Moreover, for a marking $m$ and a transition $t \in en(m)$ we define the constraint

$$fire_t(m) = \bigwedge_{\mathfrak{i} \in I\mathcal{V}(t)} x_\mathfrak{i} \geq Eft(t)$$

which expresses the condition under which $t$ can be fired at $m$ (note that the marking invariant, which obviously holds if $\mathcal{N}$ is in the marking $m$, implies that at least one process involved in $t$ has the value of its clock not greater than $Lft(t)$). Given a marking $m$ and $t \in en(m)$, firing $t$ at $m$ results in assigning the value 0 to the clocks belonging to the set

$$reset(m, t) = \{x_\mathfrak{i} \in \mathcal{X} \mid \mathfrak{i} \in I\mathcal{V}(t)\}.$$

Having all the above components, we can introduce the extended detailed region graph for $\mathcal{N}$. Let $\mathcal{C}_\mathcal{N} \subseteq \mathcal{C}_\mathcal{X}$ be a non-empty set of constraints defined by

$$\mathfrak{cc} := \ x_\mathfrak{i} \geq Eft(t) \mid x_\mathfrak{i} \leq Lft(t') \mid \mathfrak{cc} \wedge \mathfrak{cc},$$

where $x_\mathfrak{i} \in \mathcal{X}$, and, for a given $i \in \mathcal{I}$, $t \in T_\mathfrak{i}$ and $t' \in T_\mathfrak{i} \cap \{t \in T \mid Lft(t) < \infty\}$. Moreover, let $frac(a)$ denote the fractional part of a number $a \in \mathbb{R}_+$, and $\lfloor a \rfloor$ denote its integral part. Then, we define equivalence classes of clock valuations [37]:

**Definition 3.** *For two clock valuations $v, v' \in \mathbb{R}^n_+$, $v \simeq_\mathcal{N} v'$ iff for all $x, x' \in \mathcal{X}$ the following conditions are met:*

1. $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$,
2. $frac(v(x)) = 0$ *iff* $frac(v'(x)) = 0$,
3. $frac(v(x)) < frac(v(x'))$ *iff* $frac(v'(x)) < frac(v'(x'))$.

The last condition implies that $frac(v(x)) = frac(v(x'))$ iff $frac(v'(x)) = frac(v'(x'))$.

We call the equivalence classes of the relation $\simeq_{\mathcal{N}}$ (*extended*) *detailed zones* for $\mathcal{X}$, and denote the set of all of them by $DZ(n)$. It is easy to see from the definition of $\simeq_{\mathcal{N}}$ that the number of extended detailed zones is countable, and that for each $\mathfrak{cc} \in \mathcal{C}_{\mathcal{N}}$ and each $Z \in DZ(n)$ either $v \models \mathfrak{cc}$ for all $v \in Z$, or $v \not\models \mathfrak{cc}$ for all $v \in Z$. We say that $Z \in DZ(n)$ *satisfies a clock constraint* $\mathfrak{cc} \in \mathcal{C}_{\mathcal{X}}$ (denoted by $Z \models \mathfrak{cc}$) iff we have $v \models \mathfrak{cc}$ for each $v \in Z$.

Given an extended detailed zone $Z \in DZ(n)$, we introduce the operation $Z[X := 0] = \{v[X := 0] \mid v \in Z\}$. Moreover, let $Z^0 = \{v \in \mathbb{R}_+^n \mid (\forall x \in \mathcal{X})\ v(x) = 0\}$. Then, we define a successor relation on zones:

**Definition 4 (Time successor).** *Let $Z$ and $Z'$ be two zones in $DZ(n)$. The zone $Z'$ is said to be the* time successor *of $Z$, denoted $\tau(Z)$, iff for each $v \in Z$ there exists $\delta \in \mathbb{R}_+$ such that $v + \delta \in Z'$.*

**Definition 5 (Action successor).** *Let $Z, Z' \in DZ(n)$. The zone $Z'$ is said to be the* action successor *of $Z$ by a transition $t \in T$, denoted $t(Z)$, if there exists a marking $m \subseteq P$ with $t \in en(m)$ such that $Z \models fire_t(m) \wedge inv(m)$ and $Z' = Z[reset(m,t) := 0]$.*

An (*extended detailed*) *region* is a pair $(m, Z)$, where $m \subseteq P$ and $Z \in DZ(n)$. Notice that the set of all the extended detailed regions is countable. Given a concrete state $\sigma = (m', clock')$ we define $\sigma \in (m, Z)$ if $m = m'$ and $v \in Z$, where $v$ is the clock valuation satisfying $v(x_{\mathfrak{i}}) = clock'(\mathfrak{i})$ for all $\mathfrak{i} \in \mathfrak{I}$. Next, we define a countable abstraction of the concrete state space of $\mathcal{N}$ - an *extended detailed region graph*.

**Definition 6.** *The* extended detailed region graph *for a net $\mathcal{N}$ is a structure $\Gamma(\mathcal{N}) = (T \cup \{\tau\}, W, w^0, \rightarrow)$, where $W = 2^P \times DZ(n)$, $w^0 = (m^0, Z^0)$, and the successor relation $\rightarrow \subseteq W \times (T \cup \{\tau\}) \times W$, where $\tau \notin T$, is defined in the following way:*

- $(m, Z) \xrightarrow{\tau} (m, Z')$ *iff* $Z, Z' \models inv(m)$ *and* $Z' = \tau(Z)$*;*
- *for* $t \in T$, $(m, Z) \xrightarrow{t} (m', Z)$ *iff* $t \in en(m)$, $m' = m[t\rangle$, $Z' = t(Z)$, $Z \models inv(m)$ *and* $Z' \models inv(m')$.

*By an* abstract model *based on $\Gamma(\mathcal{N})$ we mean a structure $M_\Gamma(\mathcal{N}) = (\Gamma(\mathcal{N}), V)$, where for each $w \in W$ and each $\sigma \in w$ we have $V(w) = V_c(\sigma)$.*

Notice that the definition of $\xrightarrow{\tau}$ is correct: in spite of a possibly non-convex form of $[\![inv(m)]\!]$, its definition ensures that if $Z, Z' \in DZ(n)$, $Z, Z' \models inv(m)$ and $(m, Z) \xrightarrow{\tau} (m, Z')$, then for any other $Z'' \in DZ(n)$ s.t. $Z'' = \tau(Z)$ and $Z' = \tau(Z'')$ (i.e., for a region $(m, Z'')$ "traversed" when the time passes between $(m, Z)$ and $(m, Z')$) the condition $Z'' \models inv(m)$ is satisfied as well. This follows from the fact that if in the zone $Z$ some $x_{\mathfrak{i}} \in \mathcal{X}$ satisfies the condition $v(x_{\mathfrak{i}}) > Lft(t)$, then the same holds also for all the time successors of $Z$, and, on the other hand, if it satisfies $v(x_{\mathfrak{i}}) \leq Lft(t)$

and this condition is violated for some $Z'' = \tau(Z)$, then there is no $Z' = \tau(Z'')$ for which it holds again.

In order to show that the model $M_\Gamma(\mathcal{N})$ preserves the behaviours of the net, we shall prove that it is *bisimulation equivalent* with $M_c(\mathcal{N})$, where the bisimulation equivalence is defined as follows:

**Definition 7.** *Let $M = ((L, S, s_0, \rightarrow), V)$ and $M' = ((L', S', s'_0, \rightarrow'), V')$ be two models of a time Petri net $\mathcal{N}$. A relation $\rightsquigarrow_s \subseteq S' \times S$ is a* simulation *from $M'$ to $M$ if the following conditions hold:*

- $s'_0 \rightsquigarrow_s s_0$,
- *for each $s \in S$ and $s' \in S'$, if $s' \rightsquigarrow_s s$, then $V(s) = V'(s')$, and for every $s_1 \in S$ such that $s \xrightarrow{l} s_1$ for some $l \in L$, there is $s'_1 \in S'$ such that $s' \xrightarrow{l'}{}' s'_1$ for some $l' \in L'$ and $s'_1 \rightsquigarrow_s s_1$.*

*The model $M'$* simulates *$M$ ($M' \rightsquigarrow_s M$) if there is a simulation from $M'$ to $M$. Two models $M$ and $M'$ are called* bisimulation equivalent *if $M' \rightsquigarrow_s M$ and $M(\rightsquigarrow_s)^{-1}M'$, where $(\rightsquigarrow_s)^{-1}$ is the inverse of $\rightsquigarrow_s$.*

Then, we can prove the following lemma:

**Lemma 1.** *For a given time Petri net $\mathcal{N}$ the models $M_c(\mathcal{N}) = ((T \cup \mathbb{R}_+, \Sigma, \sigma^0, \rightarrow_c), V_c)$ and $M_\Gamma(\mathcal{N}) = ((T \cup \{\tau\}, W, w^0, \rightarrow), V)$ are bisimulation equivalent.*

The proof can be found in the appendix.

## 4 Testing Reachability via BMC

The reachability problem for a system $S$ consists in checking, given a property $p$, whether $S$ can ever be in a state where $p$ holds (which can be described by the CTL formula $\mathrm{EF}p$ - "there exists a path s.t. at that path the property $p$ finally holds"). The property is expressed in terms of propositional variables. In the case the system $S$ is represented by a time Petri net $\mathcal{N}$, the propositions correspond to the set of its places $P$. Therefore, the reachability verification can be translated to testing whether the set $Reach_{\mathcal{N}}$ contains a state whose marking includes a given subset of $P$. Checking this can be performed by an explicit exploration of the concrete state space (model), but due to its infinity such an approach is usually very inefficient.

If a reachable state satisfying the property $p$ exists, this can be usually proven exploiting a part of the model only. This enables us to apply the bounded model checking approach. The basic idea of testing reachability using BMC consists in searching for a *reachability witness* of a bounded length $k$ (i.e., for a path of a length $k \in \mathbb{N}_+$, called a *$k$-path*, which leads from the initial state to a state satisfying $p$). Searching for a reachability witness is performed by generating a propositional formula that is satisfiable iff such a witness exists. Satisfiability of this formula is checked using a SAT-solver.

To apply the above procedure, we represent the states of a model $M(\mathcal{N})$ for a given time Petri net $\mathcal{N}$ as vectors of boolean variables, and express the transition relation

of the model in terms of propositional formulas. Then, we *encode* all the $k$-paths of $M(\mathcal{N})$ starting at its initial state as a propositional formula $\alpha_k$, and check satisfiability of a formula $\gamma_k$ which is the conjunction of $\alpha_k$ and a propositional formula expressing that the property $p$ holds at some state of a $k$-path. The above process is started from $k = 1$, and repeated iteratively up to $k = |M|$. It, however, can be stopped, since if for some $k$ the formula $\gamma_k$ is satisfiable, then reachability of a state is proven, and no further tests are necessary.

The above method can be inefficient if no state satisfying $p$ exists, since the length of the $k$-path strongly influences the size of its propositional encoding. Therefore, in order to prove unreachability of a state satisfying $p$, another solution, shown in [36], is applied. A sketch of the idea is as follows: using the BMC procedures, we search for a longest $k$-path starting from an arbitrary state of $M$ (a *free path*) such that $p$ holds only in the last state of this path. If such a path $\pi$ is found, then this means that in order to learn whether a state satisfying $p$ is reachable we need to explore the model only to the depth equal to the length of $\pi$.

## 5 Implementation for Time Petri Nets

In order to apply the above approach to verification of a particular distributed time Petri net $\mathcal{N}$, we deal with a model obtained by a *discretisation* of its extended detailed region graph. The model is of an infinite but countable structure, which, however, is sufficient for BMC (which deals with finite sequences of states only). Below, we show this discretisation, and then encode the transition relation of the model.

### 5.1 Discretisation of Extended Detailed Region Graphs

Let $\Gamma(\mathcal{N}) = (T \cup \{\tau\}, W, w^0, \rightarrow)$ be the extended detailed region graph for a distributed time Petri net $\mathcal{N}$, and $\mathcal{X}$ be the set of clocks corresponding to its processes. Instead of dealing with the whole extended detailed region graph $\Gamma(\mathcal{N})$, we *discretise* this structure, chosing for each region one or more appropriate representatives. The discretisation scheme is based on the one for timed automata [37], and preserves the qualitative behaviour of the underlying system.

Let $n$ be the number of clocks, and let $c_{max}(\mathcal{N})$ be the largest constant appearing in $\mathcal{C}_{\mathcal{N}}$ (i.e., the greatest finite value of $Eft$ and $Lft$). For each $m \in \mathbb{N}$, we define

$$\mathbb{D}_m = \{d \in \mathbb{Q} \mid (\exists k \in \mathbb{N})\ d \cdot 2^m = k\},$$

and

$$\mathbb{E}_m = \{e \in \mathbb{Q} \mid (\exists k \in \mathbb{N})\ e \cdot 2^m = k \wedge e \leq c_{max}(\mathcal{N}) + 1\}.$$

The *discretised clock space* is defined as $\mathbb{D}^n$, where $\mathbb{D} = \bigcup_{m=1}^{\infty} \mathbb{D}_m$. Similarly, the set of possible values of time passings is defined as $\mathbb{E} = \bigcup_{m=1}^{\infty} \mathbb{E}_m$. The above definitions give us that the maximal values of time passings are restricted to $c_{max}(\mathcal{N}) + 1$, which is sufficient to express the behaviour of the net. Moreover, such a clock space and the set of lengths of timed steps ensure that for any representative of an extended detailed region there is another representative of this region which can be reached by a time

step of a length $e \in \mathbb{E}$. It should be mentioned that such a solution (different than in [24]) allows us to compute precisely the time passed along a $k$-path (which was difficult while using the so-called "adjust transitions" of [24]).

Now, we can introduce discretised region graphs and models:

**Definition 8.** *The* extended discretised region graph *based on the extended detailed region graph $\Gamma(\mathcal{N})$, is a structure $\widetilde{\Gamma}(\mathcal{N}) = (T \cup \mathbb{E}, \widetilde{W}, w^0, \rightarrow_d)$, where $\widetilde{W} = 2^P \times \mathbb{D}^n$, $w^0 = (m^0, Z^0)$, and the labelled transition relation $\rightarrow_d \subseteq \widetilde{W} \times (T \cup \mathbb{E}) \times \widetilde{W}$ is defined as*

1. *for $t \in T$, $(m, v) \xrightarrow{t}_d (m', v')$ iff $t \in en(m)$, $m' = m[t\rangle$, $v \models fire_t(m) \wedge inv(m)$, $v' = v[reset(m, t) := 0]$, and $v' \models inv(m')$ (*action transition*);*
2. *for $\delta \in \mathbb{E}$, $(m, v) \xrightarrow{\delta}_d (m, v')$ iff $v' = v + \delta$ and $v, v' \models inv(m)$ (*time transition*).*

Given an abstract model $M_\Gamma(\mathcal{N}) = (\Gamma(\mathcal{N}), V)$ based on $\Gamma(\mathcal{N}) = (T \cup \{\tau\}, W, w^0, \rightarrow)$ and the discretised model $\widetilde{\Gamma}(\mathcal{N})$, we can define a *discretized model* based on $\widetilde{\Gamma}(\mathcal{N})$, which is a structure $\widetilde{M}_\Gamma(\mathcal{N}) = (\widetilde{\Gamma}(\mathcal{N}), \widetilde{V})$, where $\widetilde{V} : \widetilde{W} \rightarrow 2^{PV}$ is a valuation function such that for each $\widetilde{w} \in \widetilde{W}$ being a representative of $w \in W$ we have $\widetilde{V}(\widetilde{w}) = V(w)$. This model will be exploited in BMC-based reachability checking.

### 5.2 Encoding of the Transition Relation of the Discretized Model

In order to apply SAT-based verification methods described in Sec. 4, we need to represent (encode) the discretized model $\widetilde{M}_\Gamma(\mathcal{N})$ as a boolean formula. To do that, we assume that each state $w \in \widetilde{W}$ is given in a unique binary form, i.e., $\widetilde{w} \in \{0,1\}^{h(m)}$, where $h(m)$ is a function of the greatest exponent appearing in the denominators of clock values in $\widetilde{w}$ (see [37] for details). The digits in the binary form of $w$ are denoted by $w(1), \ldots, w(h)$. Therefore, the elements of $\widetilde{W}$ can be "generically" represented by a vector $\mathbf{w} = (\mathrm{w}[1], \ldots, \mathrm{w}[h(m)])$ of propositional variables (called a *symbolic state*), whose valuation (i.e., assignment of values to the variables) represents $w$ iff for each $j \in \{1, \ldots, h(m)\}$ we have $\mathrm{w}[j] = true$ iff $w(j) = 1$, and $\mathrm{w}[j] = false$ otherwise. Moreover, each $k$-path in $\widetilde{\Gamma}(\mathcal{N})$ can be represented by a finite sequence $\mathbf{w}_0, \ldots, \mathbf{w}_k$ of symbolic states, and again, such a representation is called a *symbolic $k$-path*.

In what follows, by *state variables* we mean propositional variables used to encode the states of $\widetilde{\Gamma}(\mathcal{N})$. The set of all the state variables, containing the symbols $true$ and $false$, will be denoted by $SV$, and the set of all the propositional formulas built over $SV$ - by $SF$. The elements of $SF$ are called *state formulas*.

In order to encode the transition relation of $\widetilde{M}_\Gamma(\mathcal{N})$, we introduce the following functions and propositional formulas:

- $lit : \{0,1\} \times SV \rightarrow SF$, which is defined by $lit(0, p) = \neg p$ and $lit(1, p) = p$;
- $I_w(\mathbf{w}) := \bigwedge_{j=1}^{h} lit(w(j), \mathrm{w}[j])$ which is true iff the vector $\mathbf{w}$ represents the state $w$;
- $\mathbf{T}(\mathbf{w}, \mathbf{w}')$ which is true iff for the states $w, w' \in \widetilde{W}$, represented by vectors $\mathbf{w}$ and $\mathbf{w}'$, respectively, it holds $w \xrightarrow{e}_d w'$ for some $e \in T \cup \mathbb{E}$.

The formula which encodes all the $k$-paths in $\widetilde{\Gamma}(\mathcal{N})$ starting at the initial state is of the form

$$\alpha_k := I_{w^0}(\mathbf{w}_0) \ \wedge \ \bigwedge_{j=0}^{k-1} \mathbf{T}(\mathbf{w}_j, \mathbf{w}_{j+1}),$$

where $\mathbf{w}_0, \ldots, \mathbf{w}_k$ is a symbolic $k$-path. In practice, we consider $k$-paths with some restrictions on repetition of the action and time transitions, and on lengths of the time steps (see [37] for details). Encoding the fact that a state satisfies a given property is straightforward.

## 6 Parametric Reachability Checking

Besides testing whether a state satisfying a property $p$ is reachable, one can be interested in finding a minimal time in which a state satisfying $p$ can be reached, or finding a minimal time after which $p$ does not hold. To this aim, *parametric reachability checking* can be used.

In order to be able to perform the above verification, we introduce an additional restriction on the nets under consideration, i.e., require they contain no cycle $C$ of transitions such that for each $t \in C$ we have $Eft(t) = 0$ (which guarantees that the time increases when the net progresses, and is a typical assumption when analysing timed systems). Moreover, we introduce the notations $\mathrm{EF}^{\sim c}p$, with $\sim \in \{\leq, <, >, \geq\}$ and $c \in \mathbb{N}$, which express that a state satisfying $p$ is reached in a time satisfying the constraint in the superscript[1]. The problems intuitively presented at the beginning of the section can be expressed respectively as finding a minimal $c$ such that $\mathrm{EF}^{<c}p$ (or $\mathrm{EF}^{\leq c}p$) holds, and finding a maximal $c$ such that $\mathrm{EF}^{>c}p$ (or $\mathrm{EF}^{\geq c}p$) holds.

An algorithm for finding a minimal $c$ such that $EF^{\leq c}p$ holds looks as follows:

1. Using the standard BMC approach, find a reachability witness of minimal length[2];
2. read from the witness the time required to reach $p$ (denoted $x$). Now, we know that $c \leq \lceil x \rceil$ (where $\lceil \cdot \rceil$ is the *ceiling* function);
3. extend the verified TPN with a new process $N$, which is composed of one transition $t$ s.t. $Eft(t) = Lft(t) = n$, and two places $p_{in}$, $p_{out}$ with $\bullet t = \{p_{in}\}$ and $t\bullet = \{p_{out}\}$ (see Fig. 2(a)),
4. set $n$ to $\lceil x \rceil - 1$,
5. Run BMC to test reachability of a state satisfying $p \wedge p_{in}$ in the extended TPN,
6. if such a state is reachable, set $n := n - 1$ and go to 5,
7. if such a state is unreachable, then $c := n + 1$, STOP.

Some comments on the above algorithm are in place. First of all, it should be explained that the BMC method described in Sec. 4 founds a reachability witness of a shortest length (i.e., involving the shortest possible $k$-path). However, the shortest path is not necessarily that of minimal time. An example can be seen in Fig. 3, where the

[1] The full version of the logic, for a discrete semantics and with $\sim$ restricted to $\leq$ only, can be found in [14].

[2] if we cannot find such a witness, then we try to prove unreachability of $p$.

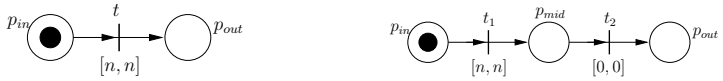


**Fig. 2.** The processes added to the nets to test parametric reachability

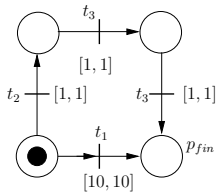


**Fig. 3.** An example net

shortest path leading to the place satisfying the property $p_{fin}$ consists of one time step and one action step (i.e., passing 10 time units and then firing $t_1$), whereas minimal time of reaching such a state is 3, which corresponds to firing $t_2$, $t_3$ and $t_4$, each of them preceded by passing one unit of time. Due to this, after finding a reachability witness for $p$ in step 1 of the algorithm, we test whether $p$ can be reached in a shorter time. Extending the net with a new process allows us to express the requirement that the time at which $p$ is reached is not greater than $n$ ($n \in \mathbb{N}$), since at time $n$ the transition $t$ has to fire, which unmarks the place $p_{in}$.

The second comment to the algorithm concerns the possible optimisations. Firstly, the algorithm can be optimized by applying one of the well-known searching algorithms instead of decreasing $n$ by one in each step. Secondly, it is easy to see that if BMC finds a reachability witness for $p$ of length $k$, then a witness for reaching $p$ in a smaller time cannot be shorter than $k$ (if such a witness existed, it would have been found previously). Thus, in step 5 of the algorithm the BMC method can start with $k$ equal to the length of the witness found in the previous run, instead of with $k = 1$.

Finally, step 7 of the algorithm should be explained. In order to decide that no state satisfying $p \wedge p_{in}$ is reachable, we should either prove unreachability of that state using the method of [36], or to find an upper bound on the length of the $k$-paths such that unreachability of $p \wedge p_{in}$ on the paths up to this length allows us to decide that no state of interest is reachable. We can do the latter in some cases only, i.e., when some restrictions on the nets considered are assumed. This is specified by the following two lemmas:

**Lemma 2.** *If a net $\mathcal{N}$ contains no transition $t$ with $Eft(t) = 0$, then the length of a reachability witness for $\mathrm{EF}^{\leq c}p$, in which time- and action steps alternate, is bounded by $2 \cdot c$.*

*Proof.* We make use of the result of [29], which states that each reachable marking of a TPN can be reached on a path whose time steps are of integer values only. Since from the stucture of the net and from the structure of the path we have that zero-time

steps are not allowed, the shortest time steps are of length one. The bound $2c$ is then straightforward.

**Lemma 3.** *Let $\mathcal{N}$ be a distributed net consisting of $n$ processes $N_\mathfrak{i} = (P_\mathfrak{i}, T_\mathfrak{i}, F_\mathfrak{i}, m^0_\mathfrak{i}, Eft_\mathfrak{i}, Lft_\mathfrak{i})$ ($\mathfrak{i} \in \mathcal{I} = \{1, \ldots, n\}$), each of which contains no cycle besides (possibly) being a cycle itself and satisfies the condition $\forall t_1, t_2 \in T_\mathfrak{i}$ ($\bullet t_1 \cap P_\mathfrak{i} = \bullet t_2 \cap P_\mathfrak{i} \iff t_1 \bullet \cap P_\mathfrak{i} = t_2 \bullet \cap P_\mathfrak{i}$). The length of a reachability witness for $\mathrm{EF}^{\leq c} p$, in which time- and action steps alternate, is bounded by $K = 2 \cdot \Sigma_{\mathfrak{i}=1}^{n} z_\mathfrak{i}$, where each $z_\mathfrak{i}$, for $\mathfrak{i} \in \mathcal{I}$, is computed according to the following algorithm:*

1. *set $g := 0$, $time := 0$, and $nextTrans$ to such $t \in T_\mathfrak{i}$ that $\bullet t = m^0_\mathfrak{i}$ and $Eft(t) = min(Eft(t') \mid t' \in T_\mathfrak{i} \wedge \bullet t' = m^0_\mathfrak{i})$,*
2. *do*
   - *$time := time + Eft(nextTrans)$;*
   - *if $time \leq c$ then set $g := g + 1$ and $s_g := nextTrans$;*
   - *set $nextTrans$ to such $t \in T_\mathfrak{i}$ that $\bullet t = s_{g-1}\bullet$ and $Eft(t) = min(Eft(t') \mid t' \in T_\mathfrak{i} \wedge \bullet t' = s_{g-1}\bullet)$,*

   *while $time <= c$ and $s_g \bullet \cap P_\mathfrak{i} \neq \emptyset$,*
3. *while $Eft(s_g) = 0$ and $(\bullet s_g \cap P_\mathfrak{i}) \notin Prop(p)$, where $Prop(p)$ is the set of propositions occuring in the property $p$, do $g := g - 1$;*
4. *$setz_\mathfrak{i} := g$.*

*Proof.* From the structure of a process of $\mathcal{N}$, we have that the algorithm for $z_\mathfrak{i}$ computes first the number of transitions which can be executed in time $c$ provided that $\mathcal{N}_\mathfrak{i}$ proceeds as fast as possible, and then optimises the value obtained by removing a number of final steps which influence neither the time nor reaching the property tested. The length of the path in which time- and action steps alternate is therefore equal to $2z_\mathfrak{i}$. Taking the sum of these values for all the processes corresponds to considernig the worst case, in which all the processes proceed independently, performing as many steps as possible.

An algorithm for finding a minimal $c$ such that $\mathrm{EF}^{<c} p$ holds is similar to the previous one:

1. Using the standard BMC approach, find a reachability witness of minimal length[3];
2. read from the witness the time required to reach $p$ (denoted $x$). Now, we know that $c \leq \lceil x \rceil$;
3. extend the verified TPN with a new process $N$, which is composed of two transitions $t_1$, $t_2$ s.t. $Eft(t_1) = Lft(t_1) = n$, $Eft(t_2) = Lft(t_2) = 0$, $\bullet t_1 = \{p_{in}\}$, $t_1\bullet = \bullet t_2 = \{p_{mid}\}$ and $t_2\bullet = \{p_{out}\}$ (see Fig. 2(b)),
4. set $n$ to $\lceil x \rceil - 1$,
5. run BMC to test reachability of a state satisfying $p \wedge p_{in}$ in the extended TPN,
6. if such a state is reachable, set $n := n - 1$ and go to 5,
7. if such a state is unreachable, set $n := n + 1$ and run BMC to test reachability of a state satisfying $p \wedge p_{mid}$ in the extended TPN,
8. if such a state is reachable, then $c := n + 1$, STOP,
9. if such a state is unreachable, then $c := n$, STOP.

[3] if we cannot find such a witness, then we try to prove unreachability of $p$.

In this case, the additional process contains the place which can be marked only if the time passed since the net started is equal to $n$. The algorithm proceeds in the following way: the steps 1 - 6 (analogous as in the previous algorithm) are aimed at finding a minimal $n$ such that $\mathrm{EF}^{\leq n}p$ holds. Then, it is tested whether $p$ can be reached exactly at time $n$. Depending on the result of this test, the bound returned is either $n$ or $n+1$ (which follows from the result of [30] stating that the minimal time duration of a transition sequence is an integer value). The improvements to the algorithms, as well as methods of deciding unreachability in steps 7 and 9, are the same as in the previous case.

The next pair of the algorithms is aimed at finding a minimal time after which no state satisfying $p$ is reachable. This can be done by searching for a maximal $c$ for which $\mathrm{EF}^{\geq c}p$ (or $\mathrm{EF}^{>c}p$) holds. The algorithm for $\mathrm{EF}^{\geq c}p$ is as follows:

1. using a standard BMC approach, test whether there is a $k$-path $\pi$ such that $p$ is reachable from its arbitrary state (i.e., whether for $\pi$ the CTL formula $\mathrm{EGEF}p$ holds),
2. if such a $k$-path can be found, then no maximal $c$ exists, STOP.
3. if such a $k$-path cannot be found then, using the standard BMC approach, find a reachability witness for $p$ of a minimal length[4].
4. read from the witness the time $x$ required to reach $p$,
5. extend the verified TPN with a new process which is composed of one transition $t$ s.t. $Eft(t) = Lft(t) = n$, and two places $p_{in}$, $p_{out}$ with $t\bullet = \{p_{out}\}$ and $\bullet t = \{p_{in}\}$,
6. set $n$ to $\lceil x \rceil$, and set an upper bound $b$ ($b \geq n$) on $c$ to be searched for[5],
7. run BMC to test reachability of a state satisfying $p \wedge p_{out}$ in the extended TPN,
8. if such a state is reachable and $n + 1 < b$, then set $n := n + 1$ and go to 7,
9. if such a state cannot be found or $n + 1 \geq b$, then set $c := n - 1$, STOP.

Testing whether there is a $k$-path s.t. $p$ is reachable from its arbitrary state (testing $\mathrm{EGEF}p$) is done by checking whether there is a path which has a loop, and there is a state of this loop at which $p$ holds. In order to ensure that there is no maximal $c$, we need also the path to be progressive, i.e., such that its loop contains at least one non-zero time step[6].

Again, some optimisations to the algorithm can be introduced. The first one can consist in applying a well-known searching technique instead of increasing $n$ by one in each step. The second is based on an observation that each reachability witness for $\mathrm{EF}^{\geq n}p$ is also a reachability witness for $\mathrm{EF}^{\geq n-1}p$. Thus, no witness for $\mathrm{EF}^{\geq n}p$ can be shorther than the shortest one found for $\mathrm{EF}^{\geq n-1}p$ (if a shorter witness existed, it would have been found while searching for a witness for $\mathrm{EF}^{\geq n-1}p$). Thus, while running

[4] if we cannot find such a $k$, then we try to prove unreachability of $p$

[5] the value $b$ can be also a parameter of the algorithm

[6] Formally, let $\pi$ be a $k$-path, $\pi(i)$ be the $i$-th state of the path, $\delta_\pi(i, i+1)$ be the time passed while moving from $\pi(i)$ to $\pi(i+1)$, $loop(\pi) = \{h \mid 0 \leq h \leq k \ \wedge \ \pi(k) \rightarrow \pi(h)\}$, and $\Pi_k(s)$ be the set of all the $k$-paths starting at $s$. The bounded semantics for $EGEF\alpha$ is as follows: $s \models EGEF\alpha \iff (\exists \pi \in \Pi_k(s))(loop(\pi) \neq \emptyset \wedge (\exists l \in loop(\pi)(\exists l \leq j \leq k)(\pi(j) \models \alpha \wedge \Sigma_{l \leq j < k} \delta_p i(j, j+1) > 0))$.

step 7 of the algorithm, we can start with $k$ equal to the length of the witness found in the previous run, instead of with $k = 1$.

It should be noticed that, contrary to the former cases, we cannot set any upper bound on the length of $k$-paths to be tested in step 9, besides the one which follows from the value $b$ assumed in the algorithm. In this case, computing the bound is done analogously as we shown in the description of the algorithm for $\mathrm{EF}^{\leq c}p$.

An algorithm for checking $EF^{>c}p$ (and searching for a maximal $c$) is as follows:

1. using a standard BMC approach, test whether there is a $k$-path $\pi$ such that $p$ is reachable from its arbitrary state (i.e., whether for $\pi$ the CTL formula $\mathrm{EGEF}p$ holds),
2. if such a $k$-path can be found, then no maximal $c$ exists, STOP.
3. if such a $k$-path cannot be found then, using the standard BMC approach, find a reachability witness for $p$ of a minimal length[7].
4. read from the witness the time $x$ required to reach $p$,
5. extend the verified TPN with a new process $N$, which is composed of two transitions $t_1$, $t_2$ s.t. $Eft(t_1) = Lft(t_1) = n$, $Eft(t_2) = Lft(t_2) = 0$, $\bullet t_1 = \{p_{in}\}$, $t_1\bullet = \bullet t_2 = \{p_{mid}\}$ and $t_2\bullet = \{p_{out}\}$,
6. set $n$ to $\lceil x \rceil$, and set an upper bound $b$ ($b \geq n$) on $c$ to be searched for[8],
7. run BMC to test reachability of a state satisfying $p \wedge p_{out}$ in the extended TPN,
8. if such a state is reachable and $n + 1 < b$, then set $n := n + 1$ and go to 7,
9. if such a state is unreachable or $n + 1 > b$, set $n := n - 1$ and run BMC to test reachability of $p \wedge p_{mid}$) in the extended TPN,
10. if such a state is reachable, then $c := n - 1$, STOP;
11. if such a state is unreachable, then $c := n$, STOP.

The idea behind the algorithm is similar to the previous approaches: first a maximal $n$ for which $\mathrm{EF}^{\geq n}p$ is found, then the algorithm tests whether reaching $p$ at time $n$ is possible. The final result depends on the answer to the latter question.

It should be mentioned that in practice all the above methods are not complete (as the BMC itself is not). It can happen that we are not able to prove unreachability of a state, compute an upper bound on the length of a $k$-path to be tested, or, in spite of finding such an upper bound, are not able to test the paths up to this length using the resources given. However, the preliminary experiments show that the methods can give quite good results.

## 7 Experimental Results

The experimental results presented below are preliminary, since some methods mentioned in the previous sections are not implemented yet. We are going to complete the implementation to the final version of this paper.

We have performed our experiments on the computer equipped with Intel Pentium Dual CPU (2.00 GHz), 2 GB main memory and the operating system Linux 2.6.28. We

[7] if we cannot find such a $k$, then we try to prove unreachability of $p$

[8] the value $b$ can be also a parameter of the algorithm

have tested some distributed time Petri nets for the standard *Fischer's mutual exclusion protocol* (mutex) [2]. The system consists of $n$ time Petri nets, each one modelling a process, plus one additional net used to coordinate their access to the critical sections. A distributed TPN modelling the system is shown in Figure 1, for the case of $n = 2$. *Mutual exclusion* means that no two processes are in their critical sections at the same time. The preservation of this property depends on the relative values of the time-delay constants $\delta$ and $\Delta$. In particular, the following holds: *"Fischer's protocol ensures mutual exclusion iff $\Delta < \delta$"*.

Our first aim was to check that if $\Delta \geq \delta$, then the mutual exclusion is violated. We considered the case with $\Delta = 2$ and $\delta = 1$. It turned out that the conjunction of the propositional formula encoding the $k$-path and the negation of the mutual exclusion property (denoted $p$) is unsatisfiable for every $k < 12$. The witness was found for $k = 12$. We were able to test $40$ processes. The results are shown in Fig. 4 (left).

Our second aim was to search for a minimal $c$ such that $\mathrm{EF}^{\leq c} p$ holds. The results are presented in Fig. 4 (right). In the case of this net, we are not able to compute an upper bound on the length of the $k$-path. Unfortunately, we also could not test unreachability, since the method is not implemented yet. Again, we considered the case with $\Delta = 2$ and $\delta = 1$, and the net of $25$ processes. The witness was found for $k = 12$, and the time of the path found was between $8$ and $9$. The column $n$ shows the values of the parameter in the additional component. For $n = 1$ and $k = 12$ unsatisfiability was returned, and testing the property on a longer path could not be completed in a reasonable time.

The next two (scalable) examples were the networks shown in Fig. 5. The net (a) shown in the left-hand side of the figure was scaled by increasing $Eft(t_2)$ and $Lft(t_2)$, according to the schema $A = 2u$, $B = 4u$, for $u = 1, 2, \ldots$. The property tested was $\mathrm{EF}(p_3 \wedge p_6)$. The net (b) shown on the right was scaled by increasing the number of components $\mathcal{N}_\mathrm{i}$ ($\mathrm{i} = 1, 2, \ldots$). In this case, reachability of a state satisfying $p_3 \wedge \bigwedge_{\mathrm{i}=1}^{j} p_3^\mathrm{i}$ was checked (where $j$ is a number of identical processes). For both the nets we searched

| | | tpnBMC | | | | RSat | | |
|---|---|---|---|---|---|---|---|---|
| **k** | **n** | **variables** | **clauses** | **sec** | **MB** | **sec** | **MB** | **sat** |
| 0 | - | 1937 | 5302 | 0.2 | 3.5 | 0.0 | 1.7 | NO |
| 2 | - | 36448 | 107684 | 1.4 | 7.9 | 0.4 | 9.5 | NO |
| 4 | - | 74338 | 220335 | 2.9 | 12.8 | 3.3 | 21.5 | NO |
| 6 | - | 112227 | 332884 | 4.2 | 17.6 | 14.3 | 37.3 | NO |
| 8 | - | 156051 | 463062 | 6.1 | 23.3 | 257.9 | 218.6 | NO |
| 10 | - | 197566 | 586144 | 7.8 | 28.5 | 2603.8 | 1153.2 | NO |
| 12 | - | 240317 | 712744 | 9.7 | 34.0 | 87.4 | 140.8 | YES |
| | | | | 32.4 | 34.0 | 2967.1 | 1153.2 | |

| | | tpnBMC | | | | RSat | | |
|---|---|---|---|---|---|---|---|---|
| **k** | **n** | **variables** | **clauses** | **sec** | **MB** | **sec** | **MB** | **sat** |
| 0 | - | 840 | 2194 | 0.0 | 3.2 | 0.0 | 1.4 | NO |
| 2 | - | 16263 | 47707 | 0.5 | 5.2 | 0.1 | 4.9 | NO |
| 4 | - | 33835 | 99739 | 1.0 | 7.3 | 0.6 | 9.1 | NO |
| 6 | - | 51406 | 151699 | 1.6 | 9.6 | 1.8 | 13.8 | NO |
| 8 | - | 72752 | 214853 | 2.4 | 12.3 | 20.6 | 27.7 | NO |
| 10 | - | 92629 | 273491 | 3.0 | 14.8 | 321.4 | 200.8 | NO |
| 12 | - | 113292 | 334357 | 3.7 | 17.5 | 14.3 | 39.0 | YES |
| 12 | 7 | 120042 | 354571 | 4.1 | 18.3 | 45.7 | 59.3 | YES |
| 12 | 6 | 120054 | 354613 | 4.0 | 18.3 | 312.7 | 206.8 | YES |
| 12 | 5 | 120102 | 354763 | 4.0 | 18.3 | 64.0 | 77.7 | YES |
| 12 | 4 | 120054 | 354601 | 4.1 | 18.3 | 8.8 | 35.0 | YES |
| 12 | 3 | 115475 | 340834 | 3.9 | 17.7 | 24.2 | 45.0 | YES |
| 12 | 2 | 115481 | 340852 | 3.9 | 17.8 | 138.7 | 100.8 | YES |
| 12 | 1 | 115529 | 341008 | 3.9 | 17.7 | 2355.4 | 433.4 | NO |
| | | | | 40.1 | 18.3 | 3308.3 | 433.4 | |

**Fig. 4.** Results for mutex, $\Delta = 2$, $\delta = 1$, mutual exclusion violated. Left: proving reachability for $40$ processes, right: parametric verification for $25$ processes.

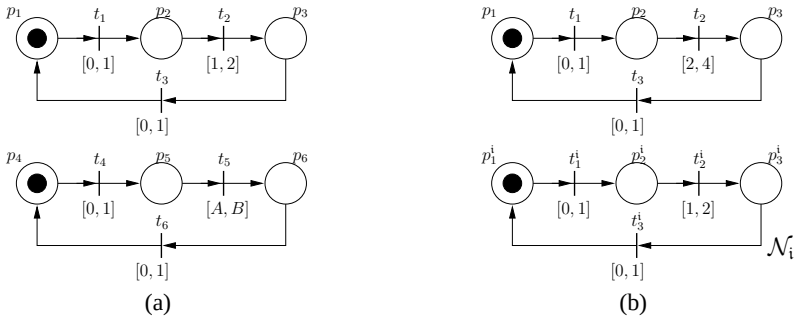


**Fig. 5.** Time Petri nets tested in experiments

for a minimal time $c$ at which a given property can be reached, and for both of them we were able to compute an upper bound on the length of the $k$-path to be tested while checking reachability in a time not exceeding $n$. For the net (a) the bound is $K = 2(z_1 + z_2 + z_3)$, where $z_1 = 3n - 1$, $z_2 = 3\lceil n/(2u)\rceil - 1$ and $z_3 = 1$ (where the third process is that added to test reachability in time $n$); whereas for the net (b) containing $j$ identical components it is given by $K = z_0 + \Sigma_{i=1}^{j} z_i + z_{j+1}$, where the bound for the first process is $z_1 = 3\lceil n/2\rceil - 1$, the bound for each of the identical processes is $z_i = 3n - 1$ ($i = 1, \ldots, j$), and the bound for the additional process is $z_{j+1} = 1$. The results for the net (a), with the values of the coefficient $u$ given, are presented in Fig. 6. In the case of $u = 4$ we were able to test the $k$-paths up to the upper bound $K = 46$, and to show that the parameter searched for is $c = 8$; for $u = 5$ we can only assume that the value of $c$ is 10, since we were not able to test all the $k$-paths of the lengths up to $K = 58$. Concerning the net (b), we were able to test the net containing 6 identical processes and to show that $c = 2$; the results are given in Fig. 7.

## 8 Final Remarks

We have shown that the BMC method for checking reachability properties of TPNs is feasible. Our preliminary experimental results prove the efficiency of the method. However, it would be interesting to check practical applicability of BMC for other examples of time Petri nets. On the other hand, it would be also interesting to check efficiency of the above solutions for other (non-distributed) nets (which could be done by applying the translations from [28]).

## References

1. R. Alur, C. Courcoubetis, and D. Dill. Model checking for real-time systems. In *Proc. of the 5th Symp. on Logic in Computer Science (LICS'90)*, pages 414–425. IEEE Computer Society, 1990.
2. R. Alur, C. Courcoubetis, D. Dill, N. Halbwachs, and H. Wong-Toi. An implementation of three algorithms for timing verification based on automata emptiness. In *Proc. of the 13th*

| | | tpnBMC | | | | RSat | | |
|---|---|---|---|---|---|---|---|---|
| **k** | **n** | **variables** | **clauses** | **sec** | **MB** | **sec** | **MB** | **sat** |
| 0 | - | 21 | 34 | 0.0 | 3.0 | 0.0 | 1.3 | NO |
| 2 | - | 603 | 1653 | 0.0 | 3.0 | 0.0 | 1.4 | NO |
| 4 | - | 1396 | 3909 | 0.0 | 3.1 | 0.0 | 1.6 | NO |
| 6 | - | 2174 | 6097 | 0.0 | 3.2 | 0.0 | 1.7 | NO |
| 8 | - | 3347 | 9429 | 0.1 | 3.3 | 0.0 | 2.0 | NO |
| 10 | - | 4345 | 12213 | 0.1 | 3.5 | 0.0 | 2.2 | NO |
| 12 | - | 5413 | 15175 | 0.1 | 3.6 | 0.0 | 2.5 | NO |
| 14 | - | 6551 | 18315 | 0.1 | 3.7 | 0.1 | 2.8 | YES |
| 14 | 7 | 8812 | 24886 | 0.1 | 4.0 | 0.1 | 3.3 | NO |
| 16 | 7 | 11299 | 31987 | 0.2 | 4.2 | 0.1 | 3.8 | NO |
| 18 | 7 | 13151 | 37159 | 0.2 | 4.5 | 0.2 | 4.2 | NO |
| 20 | 7 | 15103 | 42595 | 0.2 | 4.8 | 118.9 | 17.1 | NO |
| 22 | 7 | 17155 | 48295 | 0.3 | 5.0 | 15.6 | 8.3 | NO |
| 24 | 7 | 19307 | 54259 | 0.3 | 5.2 | 18.9 | 9.7 | NO |
| 26 | 7 | 21559 | 60487 | 0.3 | 5.5 | 133.5 | 19.0 | NO |
| 28 | 7 | 23911 | 66979 | 0.4 | 5.8 | 167.1 | 26.5 | NO |
| 30 | 7 | 27930 | 78424 | 0.4 | 6.2 | 96.4 | 18.5 | NO |
| 32 | 7 | 30586 | 85756 | 0.5 | 6.5 | 224.9 | 32.6 | NO |
| 34 | 7 | 33342 | 93352 | 0.5 | 6.8 | 339.8 | 36.4 | NO |
| 36 | 7 | 36198 | 101212 | 0.5 | 7.2 | 549.8 | 50.2 | NO |
| 38 | 7 | 39154 | 109336 | 0.6 | 7.5 | 339.8 | 50.7 | NO |
| 40 | 7 | 42210 | 117724 | 0.6 | 7.9 | 266.5 | 45.6 | NO |
| 42 | 7 | 45366 | 126376 | 0.7 | 8.2 | 1026.9 | 85.0 | NO |
| 44 | 7 | 48622 | 135292 | 0.7 | 8.6 | 558.6 | 83.3 | NO |
| 46 | 7 | 51978 | 144472 | 0.8 | 9.0 | 574.7 | 75.6 | NO |
| | | | | 7.6 | 9.0 | 4431.9 | 85.0 | |

| | | tpnBMC | | | | RSat | | |
|---|---|---|---|---|---|---|---|---|
| **k** | **n** | **variables** | **clauses** | **sec** | **MB** | **sec** | **MB** | **sat** |
| 0 | - | 21 | 34 | 0.0 | 3.0 | 0.0 | 1.3 | NO |
| 2 | - | 619 | 1707 | 0.0 | 3.0 | 0.0 | 1.4 | NO |
| 4 | - | 1428 | 4017 | 0.0 | 3.1 | 0.0 | 1.6 | NO |
| 6 | - | 2467 | 6985 | 0.0 | 3.2 | 0.0 | 1.8 | NO |
| 8 | - | 3411 | 9645 | 0.0 | 3.3 | 0.0 | 2.0 | NO |
| 10 | - | 4425 | 12483 | 0.1 | 3.5 | 0.0 | 2.2 | NO |
| 12 | - | 5994 | 16945 | 0.1 | 3.6 | 0.0 | 2.6 | NO |
| 14 | - | 7228 | 20379 | 0.1 | 3.7 | 0.1 | 2.9 | NO |
| 16 | - | 8532 | 23991 | 0.1 | 3.9 | 0.1 | 3.2 | NO |
| 18 | - | 9906 | 27781 | 0.1 | 4.1 | 0.1 | 3.5 | NO |
| 20 | - | 11350 | 31749 | 0.2 | 4.2 | 0.1 | 3.8 | YES |
| 20 | 10 | 15223 | 42995 | 0.2 | 4.8 | 0.2 | 4.8 | YES |
| 20 | 9 | 15303 | 43255 | 0.2 | 4.8 | 5.1 | 6.0 | NO |
| 22 | 9 | 17375 | 49021 | 0.3 | 5.0 | 58.9 | 10.9 | NO |
| 24 | 9 | 20802 | 58804 | 0.3 | 5.4 | 9.1 | 9.7 | NO |
| 26 | 9 | 23178 | 65410 | 0.3 | 5.7 | 73.4 | 16.4 | NO |
| 28 | 9 | 25654 | 72280 | 0.4 | 5.9 | 139.2 | 21.1 | NO |
| 30 | 9 | 28230 | 79414 | 0.4 | 6.3 | 185.2 | 22.6 | NO |
| 32 | 9 | 30906 | 86812 | 0.5 | 6.6 | 1974.1 | 115.3 | NO |
| 34 | 9 | 33682 | 94474 | 0.5 | 6.9 | 566.0 | 64.9 | NO |
| 36 | 9 | 36558 | 102400 | 0.5 | 7.2 | 955.7 | 67.8 | NO |
| 38 | 9 | 39534 | 110590 | 0.6 | 7.6 | 2931.3 | 160.6 | NO |
| 40 | 9 | 42610 | 119044 | 0.7 | 8.0 | 4771.1 | 187.6 | NO |
| | | | | 5.7 | 8.0 | 11669.7 | 187.6 | |

**Fig. 6.** Results for net (a). Left: $u = 4$, $K = 46$ (unreachability proven). Right: $u = 5$, $K = 58$ (unreachability not proven)


*IEEE Real-Time Systems Symposium (RTSS'92)*, pages 157–166. IEEE Computer Society, 1992.

3. G. Audemard, A. Cimatti, A. Kornilowicz, and R. Sebastiani. Bounded model checking for timed systems. In *Proc. of the 22nd Int. Conf. on Formal Techniques for Networked and Distributed Systems (FORTE'02)*, volume 2529 of *LNCS*, pages 243–259. Springer-Verlag, 2002.
4. M. Benedetti and A. Cimatti. Bounded model checking for Past LTL. In *Proc. of the 9th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'03)*, volume 2619 of *LNCS*, pages 18–33. Springer-Verlag, 2003.
5. B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE Trans. on Software Eng.*, 17(3):259–273, 1991.
6. B. Berthomieu and M. Menasche. An enumerative approach for analyzing time Petri nets. In *Proc. of the 9th IFIP World Computer Congress*, volume 9 of *Information Processing*, pages 41–46. North Holland/ IFIP, September 1983.
7. A. Biere, A. Cimatti, E. Clarke, M.Fujita, and Y. Zhu. Symbolic model checking using SAT procedures instead of BDDs. In *Proc. of the ACM/IEEE Design Automation Conference (DAC'99)*, pages 317–320, 1999.
8. H. Boucheneb and K. Barkaoui. Relevant timed schedules / clock valuations for constructing time Petri net reachability graph. In *Proc. of the 6th Int. Workshop on Formal Analysis and Modeling of Timed Systems (FORMATS'08)*, volume 5215 of *LNCS*, pages 265–279. Springer-Verlag, 2008.

| | | tpnBMC | | | | RSat | | |
|---|---|---|---|---|---|---|---|---|
| **k** | **n** | **variables** | **clauses** | **sec** | **MB** | **sec** | **MB** | **sat** |
| 0 | - | 71 | 124 | 0.0 | 3.0 | 0.0 | 1.3 | NO |
| 2 | - | 1567 | 4394 | 0.0 | 3.1 | 0.0 | 1.6 | NO |
| 4 | - | 3754 | 10753 | 0.1 | 3.3 | 0.0 | 2.1 | NO |
| 6 | - | 6661 | 19240 | 0.1 | 3.7 | 0.0 | 2.8 | NO |
| 8 | - | 9273 | 26794 | 0.2 | 4.0 | 0.0 | 3.4 | NO |
| 10 | - | 12105 | 34956 | 0.2 | 4.4 | 0.1 | 4.0 | NO |
| 12 | - | 16672 | 48307 | 0.3 | 4.9 | 0.2 | 5.1 | NO |
| 14 | - | 20194 | 58445 | 0.4 | 5.4 | 0.5 | 5.9 | NO |
| 16 | - | 23936 | 69191 | 0.4 | 5.8 | 1.4 | 6.8 | NO |
| 18 | - | 27898 | 80545 | 0.5 | 6.3 | 2.9 | 8.0 | NO |
| 20 | - | 32080 | 92507 | 0.6 | 6.8 | 10.1 | 10.2 | NO |
| 22 | - | 39247 | 113458 | 0.7 | 7.7 | 88.9 | 19.0 | NO |
| 24 | - | 44119 | 127396 | 0.8 | 8.3 | 341.1 | 26.9 | NO |
| 26 | - | 49211 | 141942 | 0.8 | 8.9 | 489.3 | 42.4 | NO |
| 28 | - | 54523 | 157096 | 0.9 | 9.6 | 6.2 | 14.5 | YES |
| 28 | 2 | 60704 | 175021 | 1.0 | 10.3 | 40.6 | 24.0 | YES |
| 28 | 1 | 60816 | 175385 | 1.1 | 10.4 | 3027.4 | 243.0 | NO |
| 30 | 1 | 67021 | 193096 | 1.1 | 11.1 | 393.9 | 75.2 | NO |
| | | | | 9.4 | 11.1 | 4402.6 | 243.0 | |

**Fig. 7.** Results for net (b) containing $6$ identical processes; the bound $K = 30$.


9. H. Boucheneb and G. Berthelot. Towards a simplified building of time Petri nets reachability graph. In *Proc. of the 5th Int. Workshop on Petri Nets and Performance Models*, pages 46–55, October 1993.
10. G. Bucci, A. Fedeli, L. Sassoli, and E. Vicaro. Modeling flexible real time systems with preemptive time Petri nets. In *Proc. of the 15th Euromicro Conference on Real-Time Systems (ECRTS'03)*, pages 279–286. IEEE Computer Society, 2003.
11. G. Bucci and E. Vicaro. Compositional validation of time-critical systems using communicating time Petri nets. *IEEE Trans. on Software Eng.*, 21(12):969–992, 1995.
12. E. Clarke, A. Biere, R. Raimi, and Y. Zhu. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, 19(1):7–34, 2001.
13. P. Dembiński, A. Janowska, P. Janowski, W. Penczek, A. Półrola, M. Szreter, B. Woźna, and A. Zbrzezny. VerICS: A tool for verifying timed automata and Estelle specifications. In *Proc. of the 9th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'03)*, volume 2619 of *LNCS*, pages 278–283. Springer-Verlag, 2003.
14. E. A. Emerson and R. Trefler. Parametric quantitative temporal reasoning. In *Proc. of the 14th Symp. on Logic in Computer Science (LICS'99)*, pages 336–343. IEEE Computer Society, July 1999.
15. G. Gardey, O. H. Roux, and O. F. Roux. Using zone graph method for computing the state space of a time Petri net. In *Proc. of the 1st Int. Workshop on Formal Analysis and Modeling of Timed Systems (FORMATS'03)*, volume 2791 of *LNCS*, pages 246–259. Springer-Verlag, 2004.
16. K. Heljanko. Bounded reachability checking with process semantics. In *Proc. of the 12th Int. Conf. on Concurrency Theory (CONCUR'01)*, volume 2154 of *LNCS*, pages 218–232. Springer-Verlag, 2001.
17. M. Huhn, P. Niebert, and F. Wallner. Verification based on local states. In *Proc. of the 4th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'98)*, volume 1384 of *LNCS*, pages 36–51. Springer-Verlag, 1998.
18. R. Janicki. Nets, sequential components and concurrency relations. *Theoretical Computer Science*, 29:87–121, 1984.

19. J. Lilius. Efficient state space search for time Petri nets. In *Proc. of MFCS Workshop on Concurrency, Brno'98*, volume 18 of *ENTCS*. Elsevier, 1999.
20. R. Mascarenhas, D. Karumuri, U. Buy, and R. Kenyon. Modeling and analysis of a virtual reality system with time Petri nets. In *Proc. of the 20th Int. Conf. on Software Engineering (ICSE'98)*, pages 33–42. IEEE Computer Society, 1998.
21. P. Merlin and D. J. Farber. Recoverability of communication protocols – implication of a theoretical study. *IEEE Trans. on Communications*, 24(9):1036–1043, 1976.
22. Y. Okawa and T. Yoneda. Symbolic CTL model checking of time Petri nets. *Electronics and Communications in Japan, Scripta Technica*, 80(4):11–20, 1997.
23. W. Penczek and A. Półrola. Specification and model checking of temporal properties in time Petri nets and timed automata. In *Proc. of the 25th Int. Conf. on Applications and Theory of Petri Nets (ICATPN'04)*, volume 3099 of *LNCS*, pages 37–76. Springer-Verlag, 2004.
24. W. Penczek, A. Półrola, B. Woźna, and A. Zbrzezny. Bounded model checking for reachability testing in time Petri nets. In *Proc. of the Int. Workshop on Concurrency, Specification and Programming (CS&P'04)*, volume 170(1) of *Informatik-Berichte*, pages 124–135. Humboldt University, 2004.
25. W. Penczek, B. Woźna, and A. Zbrzezny. Bounded model checking for the universal fragment of CTL. *Fundamenta Informaticae*, 51(1-2):135–156, 2002.
26. W. Penczek, B. Woźna, and A. Zbrzezny. Branching time bounded model checking for elementary net systems. Technical Report 940, ICS PAS, Ordona 21, 01-237 Warsaw, January 2002.
27. W. Penczek, B. Woźna, and A. Zbrzezny. Towards bounded model checking for the universal fragment of TCTL. In *Proc. of the 7th Int. Symp. on Formal Techniques in Real-Time and Fault Tolerant Systems (FTRTFT'02)*, volume 2469 of *LNCS*, pages 265–288. Springer-Verlag, 2002.
28. A. Półrola and W. Penczek. Minimization algorithms for time Petri nets. *Fundamenta Informaticae*, 60(1-4):307–331, 2004.
29. L. Popova. On time Petri nets. *Elektronische Informationsverarbeitung und Kybernetik*, 27(4):227–244, 1991.
30. L. Popova-Zeugmann and D. Schlatter. Analyzing paths in time Petri nets. *Fundamenta Informaticae*, 37(3):311–327, 1999.
31. O. Strichman. Tuning SAT checkers for bounded model checking. In *Proc. of the 12th Int. Conf. on Computer Aided Verification (CAV'00)*, volume 1855 of *LNCS*, pages 480–494. Springer-Verlag, 2000.
32. L-M. Tranouez, D. Lime, and O. H. Roux. Parametric model checking of time Petri nets with stopwatches using the state-class graph. In *Proc. of the 6th Int. Workshop on Formal Analysis and Modeling of Timed Systems (FORMATS'08)*, volume 5215 of *LNCS*, pages 280–294. Springer-Verlag, 2008.
33. I. B. Virbitskaite and E. A. Pokozy. A partial order method for the verification of time Petri nets. In *Fundamental of Computation Theory*, volume 1684 of *LNCS*, pages 547–558. Springer-Verlag, 1999.
34. B. Woźna. ACTL$^*$ properties and bounded model checking. *Fundamenta Informaticae*, 63(1):65–87, 2004.
35. B. Woźna, A. Zbrzezny, and W. Penczek. Checking reachability properties for timed automata via SAT. *Fundamenta Informaticae*, 55(2):223–241, 2003.
36. A. Zbrzezny. Improvements in SAT-based reachability analysis for timed automata. *Fundamenta Informaticae*, 60(1-4):417–434, 2004.
37. A. Zbrzezny. SAT-based reachability checking for timed automata with diagonal constraints. *Fundamenta Informaticae*, 67(1-3):303–322, 2005.

# 9 Appendix

Below, we provide a proof of Lemma 1:

*Proof.* We shall show that the relation $\mathcal{R} = \{(\sigma, w) \mid \sigma \in w\}$ is a bisimulation. It is easy to see that $\sigma^0 \mathcal{R} w^0$, and that for each $\sigma \in w$ we have $V_c(\sigma) = V(w)$, since the markings of the related states are equal. Thus, consider $\sigma = (m, clock) \in \Sigma$ and $w = (m, Z) \in W$ such that $\sigma \mathcal{R} w$.

- If $w \xrightarrow{\tau} w'$, where $w' = (m', Z') \in W$, then from $Z' = \tau(Z)$ we have that for each $v \in Z$ (and therefore for that given by $v(x_{\mathfrak{i}}) = clock(\mathfrak{i})$ for all $\mathfrak{i} \in \mathcal{I}$) there exists $\delta \in \mathbb{R}_+$ such that $v + \delta \in Z'$. Moreover, the condition $Z' \models inv(m)$ implies that for each $t \in en(m)$ there is $\mathfrak{i} \in I\mathcal{V}(t)$ such that $(v + \delta)(x_{\mathfrak{i}}) \leq Lft(t)$. Thus, there exists a state $\sigma' \in \Sigma$, given by $\sigma' = (m, clock + \delta)$, satisfying $\sigma \xrightarrow{\delta}_c \sigma'$ and $\sigma' \in w'$ (i.e., $\sigma' \mathcal{R} w'$).
- On the other hand, if $\sigma \xrightarrow{\delta} \sigma'$ for some $\sigma' = (m, clock') \in \Sigma$ and $\delta \in \mathbb{R}_+$, then for each $\sigma_1 = (m, clock_1) \in w$ one can find $\delta' \in \mathbb{R}_+$ such that the clock valuation $v_1'$ given by $v_1'(x_{\mathfrak{i}}) = clock_1(\mathfrak{i}) + \delta'$ is equivalent to the clock valuation $v'$ given by $v'(x_{\mathfrak{i}}) = clock'(\mathfrak{i})$ (intuitively, $\delta'$ should be chosen such that the increase from $clock_1(\mathfrak{i})$ to $clock_1(\mathfrak{i}) + \delta'$ should "cross" as many integer bounds as the increase from $clock(\mathfrak{i})$ to $clock(\mathfrak{i}) + \delta$, for each $\mathfrak{i} \in \mathcal{I}$). Moreover, from the definition of the time-successor relation we have that for each $t \in en(m)$ there is $\mathfrak{i} \in I\mathcal{V}(t)$ such that $clock'(\mathfrak{i}) \leq Lft(t)$, and therefore from the definition of $\simeq_{\mathcal{N}}$ it holds also $clock_1(\mathfrak{i}) + \delta' \leq Lft(t)$. Thus, for the extended detailed region $w' = (m, Z')$ such that $\sigma' \in w'$ (and therefore $w' \mathcal{R}^{-1} \sigma'$) we have $Z' = \tau(Z)$ and $Z' \models inv(m)$, which implies $w \xrightarrow{\tau} w'$.
- If $w \xrightarrow{t} w'$ for some transition $t \in T$, where $w' = (m[t\rangle, Z') \in W$, then $t \in en(m)$ and $Z \models fire_t(m) \wedge inv(m)$. Thus, it is easy to see that the transition $t$ can be fired also at the state $\sigma$, which leads to $\sigma' = (m', clock') \in \Sigma$, with $m' = m[t\rangle$ and $clock'(\mathfrak{i}) = 0$ for $\mathfrak{i} \in I\mathcal{V}(t)$, and $clock'(\mathfrak{i}) = clock(\mathfrak{i})$ otherwise. Therefore, the clock valuation $v'$ given by $v'(x_{\mathfrak{i}}) = clock'(\mathfrak{i})$ belongs to the zone $Z[reset(t, m) := 0]$, which implies $\sigma' \in w'$ (and therefore $\sigma' \mathcal{R} w'$).
- If $\sigma \xrightarrow{t} \sigma'$ for some transition $t \in T$ and $\sigma' = (m', clock') \in \Sigma$, then $t \in en(m)$, $clock(\mathfrak{i}) \geq Eft(t)$ for every $\mathfrak{i} \in I\mathcal{V}(t)$, and there exists $\mathfrak{i} \in I\mathcal{V}(t)$ such that $clock(\mathfrak{i}) \leq Lft(t)$. Thus, from the definition of $\simeq_{\mathcal{N}}$ the zone $Z$ satisfies the constraints $fire_t(m)$ and $inv(m)$. Considering $w' = (m', Z')$ such that $\sigma' \in w'$, it is easy to see from the definition of $\simeq_{\mathcal{N}}$ that $Z' = Z[reset(m, t) := 0]$ (the zone $Z$ collects the clock valuations equivalent to $v$ given by $v(x_{\mathfrak{i}}) = clock(\mathfrak{i})$ for each $\mathfrak{i} \in \mathcal{I}$; therefore from $\sigma \xrightarrow{t} \sigma'$ and from the definition of $\simeq_{\mathcal{N}}$ the zone $Z'$ collects the valuations which are like the elements of $Z$ but with the clocks $x_{\mathfrak{i}}$ with $\mathfrak{i} \in I\mathcal{V}(t)$ set to zero). Thus, $Z' = t(Z)$. Moreover, $Z' \models inv(m')$ in an obvious way (we have $m' = m[t\rangle$; if a transition $t' \in en(m')$ became enabled by firing $t$ then there exists $\mathfrak{i} \in I\mathcal{V}(t')$ such that for all $v' \in Z'$ $v'(x_{\mathfrak{i}}) = 0$ (and therefore $v(x_{\mathfrak{i}}) \leq Lft(t')$), whereas for all the other transitions $t \in en(m')$ the existence of $\mathfrak{i} \in \mathcal{I}$ s.t. $v(x_{\mathfrak{i}}) \leq Lft(t)$ follows from $Z \models inv(m)$, since the values of clocks

have not been increased). Thus, for the detailed region $w'$ such that $\sigma' \in w'$ (and therefore $w'\mathcal{R}^{-1}\sigma'$) we have $w \xrightarrow{t} w'$, which ends the proof.