Using Integer Time Steps for Checking Branching Time Properties of Time Petri Nets

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

[1] Institute of Informatics, University of Warsaw, Banacha 2, 02-097 Warsaw, Poland
janowska@mimuw.edu.pl
[2] Institute of Computer Science, PAS, Ordona 21, 01-237 Warsaw, Poland
penczek@ipipan.waw.pl
[3] University of Łódź, FMCS, Banacha 22, 90-238 Łódź, Poland
polrola@math.uni.lodz.pl
[4] Jan Długosz University, IMCS, Armii Krajowej 13/15, 42-200 Częstochowa, Poland
a.zbrzezny@ajd.czest.pl

Abstract. Verification of timed systems is an important subject of research, and one of its crucial aspects is the efficiency of the methods developed. Extending the result of Popova which states that integer time steps are sufficient to test reachability properties of time Petri nets [5, 8], in our work we prove that the discrete-time semantics is also sufficient to verify ECTL* and ACTL* properties of TPNs with the dense semantics. To show that considering this semantics instead of the dense one is profitable, we compare the results for SAT-based bounded model checking of ACTL$_{-\mathrm{X}}$ properties and the class of distributed time Petri nets.

1 Introduction

Verification of time-dependent systems is an important subject of research. The crucial problem to deal with is the state explosion: the state spaces of these systems are usually very large due to infinity of the dense time domain, and are likely to grow exponentially in the number of concurrent components of the system. This influences strongly the efficiency of the model checking methods.

The papers of Popova [5, 8] show that in the case of checking reachability for systems modelled by time Petri nets (i.e., while testing whether a marking of a net is reachable) one can use discrete (integer) time steps instead of real-valued ones. This reduces the state space to be searched. The aim of our work is to investigate whether the result of Popova can be extended, i.e., whether the discrete-time semantics can replace the dense-time one also while verifying a wider class of properties of dense-time Petri net systems. In this paper we present our preliminary result, i.e., prove that the discrete-time model can be used instead of the dense-time one while verifying ECTL* and ACTL* properties. To show that such an approach can be profitable we perform some experiments, using an implementation for SAT-based bounded model checking of ACTL$_{-\mathrm{X}}$

and the class of distributed time Petri nets with the discrete-time semantics [4], as well as its modification for the dense-time case.

The rest of the paper is organised as follows: Sec. 2 discusses the related work. Sec. 3 introduces time Petri nets and their dense and discrete models. Sec. 4 presents the logics ECTL* and ACTL*. Sec. 5 deals with the theoretical considerations, while Sec. 6 presents the experimental results. Sec. 7 contains final remarks and sketches directions of the further work.

2 Related Works

We would like to stress that in our work we are interested in branching time properties. To our best knowledge the fact that the discrete-time semantics is sufficient to verify ECTL* or ACTL* properties of time Petri nets (TPNs) with the dense-time semantics has never been proven before.

The topic of verification of dense-time Petri nets using integer time steps has been studied in several publications. In paper [5] it is shown how to construct a reachability graph whose vertices are reachable integer states for time Petri nets in which all the latest firing times are finite. The main theorem of [5] (Thm 3.2) states that for each run of a TPN, starting at its initial state, it is possible to find a corresponding run which starts at the initial state as well, and visits integer states only. Due to this theorem a discrete analysis of boundedness and liveness of a TPN is possible. The work [6] extends the results of [5] to arbitrary TPNs. It uses the idea of "freezing" the clock values of transitions with infinite Lft just as their Eft is reached. This way a reduced (finite) reachability graph of "essential" (integer) states is obtained.

In [8] and [7] the state space of a TPN is characterised parametrically. The main theorems (Thm 3.1 and Thm 3.2) of [8] state that for an arbitrary feasible execution path where the clocks have real values it is possible to replace these real values by integer ones and obtain another feasible path. The differences between the clocks values of each enabled transition at a given marking in the former and the latter path are always smaller than 1, and so are the differences between total times of both the executions. The main idea of the proof is as follows: the integer values are constructed out of the given assignments of real values by successive transforming all the non-integer numbers to nearby integers in $n + 1$ steps, where n is the length of the path. According to the theorems the minimal and maximal time duration of a transition sequence are integer values. In the paper [7] an enumerative procedure for reducing the state space is introduced. The idea is to divide a problem into a finite number of smaller problems, which can be solved recursively with a methodology inspired from dynamic programming. Moreover, it extends the method of [8] to the nets with real-valued time steps (in [8] rational time steps were allowed only) and infinite latest firing times.

The authors of the above-mentioned papers claim that the knowledge of the reachable integer states is sufficient to determine the entire behaviour of the net at any point in time. However, all these papers show the trace equivalence

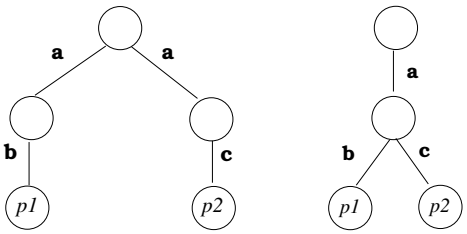

Fig. 1. Two trace equivalent models which are not (bi)similar. A formula distinguishing them is e.g. $\varphi = \mathrm{EF}(\mathrm{EX}p1 \wedge \mathrm{EX}p2)$ which holds for the model on the right only

between a continuous model and a (restricted) discrete one. It is very well known that trace equivalence preserves linear time properties, but it does not preserve branching time properties (see Fig. 1 and [1]), so the word "behaviour" should probably be understood in a way following from a fragment of [7]: *"The properties of a Petri net, both the classical one as well as the TPN, can be divided into two parts: There are static properties, like being pure, ordinary, free choice, extended simple, conservative, etc., and there are dynamic properties like being bounded, live, reachable, and having place- or transitions invariants, deadlocks, etc. While it is easy to prove the static behavior of a net using only the static definition, the dynamic behavior depends on both the static and dynamic definitions and is quite complicated to prove."*, so as the dynamic properties listed. Moreover, the result of the papers [5, 6] does not imply bisimulation between both the models, as the construction given in these papers cannot be used to prove it. We discuss this on p. 27, showing that the relation $\mathcal{R}$ used in our proof and derived from the result of [5, 6] cannot be used to prove bisimulation. This follows from the fact that the integer run π' "justifying" $\sigma'\mathcal{R}\sigma$ (generated according to the construction of [5]) and the dense run π occurring in the relation do not need to "branch" in the same way. Similarly, the result of [8, 7] does not imply (bi)simulation as well. Although it is not stated directly, the construction given in these papers is based on a parametric description of the classes of the *forward-reachability graph* for a net considered (i.e., a structure in which the initial state class contains the initial state of the net and all the time successors of this state, and given a state class C_x corresponding to firing a sequence of transitions x, its successor class on a transition t contains all the concrete states which can be obtained by firing t at a concrete state $\sigma \in C_x$ and then passing some time not disabling any enabled transition). It is well known that such a structure preserves reachability and linear time properties, but it does not preserve branching time properties. The discrete runs constructed in both the papers are retrieved from the dense ones to preserve nothing but visiting the same state classes as the runs they correspond to.

The current paper is a modified and improved version of our work [3] (published in the proceedings of a local workshop, and containing a completely different proof which does not define simulation explicitely).

3 Time Petri Nets

We start from introducing some basic definitions related to time Petri nets. For simplicity of the presentation we focus on 1-safe time Petri nets only. However, our result applies also to unbounded nets, which is explained in more details in the final section.

Let $\mathbb{N}$ be the set of natural numbers (including zero), and $\mathbb{R}$ ($\mathbb{R}_+$) be the set of (nonnegative) reals. Time Petri nets are defined as follows:

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

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 nonempty. 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.
- $en(m) = \{t \in T \mid m[t\rangle\}$;
- for $t \in en(m)$, $newly\_en(m, t) = \{u \in T \mid u \in en(m[t\rangle) \ \wedge \ (t \bullet \cap \bullet u \neq \emptyset \vee u \bullet \cap \bullet t \neq \emptyset)\}$.

Concerning the behaviour of time Petri nets, it is possible to consider a dense-time semantics, i.e. the one in which the time steps can be of an arbitrary (nonnegative) real-valued length, and the discrete one which considers integer time passings only. Below we define both of them.

3.1 Dense-Time Semantics

In the dense-time semantics (the *dense semantics* in short) a *concrete state* σ of a net $\mathcal{N}$ is defined as a pair $(m, clock)$, where m is a marking, and $clock : T \to \mathbb{R}_+$ is a function which for each transition $t \in en(m)$ gives the time elapsed since t became enabled most recently, and assigns zero to other transitions. Given a state $(m, clock)$ and $\delta \in \mathbb{R}_+$, denote by $clock + \delta$ the function defined by $(clock + \delta)(t) = clock(t) + \delta$ for each $t \in en(m)$, and $(clock + \delta)(t) = 0$ otherwise. By $(m, clock) + \delta$ we denote $(m, clock + \delta)$. The *dense concrete state space* of $\mathcal{N}$ is a structure $(T \cup \mathbb{R}_+, \Sigma, \sigma^0, \to_r)$, where Σ is the set of all the concrete states of $\mathcal{N}$, $\sigma^0 = (m^0, clock^0)$ with $clock^0(t) = 0$ for each $t \in T$ is the initial state of $\mathcal{N}$, and $\to_r \subseteq \Sigma \times (T \cup \mathbb{R}_+) \times \Sigma$ is a timed consecution relation defined by:

- for $\delta \in \mathbb{R}_+$, $(m, clock) \xrightarrow{\delta}_r (m, clock + \delta)$ iff $(clock + \delta)(t) \leq Lft(t)$ for all $t \in en(m)$ (*time successor*),
- for $t \in T$, $(m, clock) \xrightarrow{t}_r (m', clock')$ iff $t \in en(m)$, $Eft(t) \leq clock(t) \leq Lft(t)$, $m' = m[t\rangle$, and for all $u \in T$ we have $clock'(u) = 0$ for $u \in newly\_en(m, t)$ and $clock'(u) = clock(u)$ otherwise (*action successor*).

Notice that firing of a transition takes no time.

Given a set of propositional variables PV, we introduce a valuation function $V : \Sigma \to 2^{PV}$ 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 $p \in P$, and use the same names for the propositions and the places. The function V is then defined by $p \in V(\sigma)$ iff $p \in m$ for each $\sigma = (m, \cdot)$. The structure $M_r(\mathcal{N}) = (T \cup \mathbb{R}_+, \Sigma, \sigma^0, \to_r, V)$ is a *dense concrete model* of $\mathcal{N}$.

A *dense σ-run* of TPN $\mathcal{N}$ is a (maximal) sequence of states: $\sigma_0 \xrightarrow{a_0}_r \sigma_1 \xrightarrow{a_1}_r \sigma_2 \xrightarrow{a_2}_r \ldots$, where $\sigma_0 = \sigma \in \Sigma$ and $a_i \in T \cup \mathbb{R}_+$ for each $i \geq 0$. A state σ is reachable in $M_r(\mathcal{N})$ if there is a dense σ^0-run $\sigma_0 \xrightarrow{a_0}_r \sigma_1 \xrightarrow{a_1}_r \sigma_2 \xrightarrow{a_2}_r \ldots$ such that $\sigma = \sigma_i$ for some $i \in \mathbb{N}$.

3.2 Discrete-Time Semantics

Alternatively, one can consider integer time passings only. In such a *discrete-time semantics* (*discrete semantics* in short) a (*discrete*) *concrete state* σ_n of a net $\mathcal{N}$ is a pair $(m, clock_n)$, where m is a marking, and $clock_n : T \to \mathbb{N}$ is a function which for each transition $t \in en(m)$ gives the time elapsed since t became enabled most recently, and assigns zero to the other transitions. Given a state $(m, clock_n)$ and $\delta \in \mathbb{N}$, we define $clock_n + \delta$ and $(m, clock_n) + \delta$ analogously as in the dense-time case. The *discrete concrete state space* of $\mathcal{N}$ is a structure $(T \cup \mathbb{N}, \Sigma_n, {\sigma_n}^0, \to_n)$, where Σ_n is the set of all the discrete concrete states of $\mathcal{N}$, ${\sigma_n}^0 = (m^0, {clock_n}^0)$ with ${clock_n}^0(t) = 0$ for each $t \in T$ is the initial state of $\mathcal{N}$, and $\to_n \subseteq \Sigma_n \times (T \cup \mathbb{N}) \times \Sigma_n$ is a timed consecution relation defined by:

- for $\delta \in \mathbb{N}$, $(m, clock_n) \xrightarrow{\delta}_n (m, clock_n + \delta)$ iff $(clock_n + \delta)(t) \leq Lft(t)$ for all $t \in en(m)$ (*time successor*),
- for $t \in T$, $(m, clock_n) \xrightarrow{t}_n (m', {clock_n}')$ iff $t \in en(m)$, $Eft(t) \leq clock_n(t) \leq Lft(t)$, $m' = m[t\rangle$, and for all $u \in T$ we have ${clock_n}'(u) = 0$ for $u \in newly\_en(m, t)$ and ${clock_n}'(u) = clock_n(u)$ otherwise (*action successor*).

Again, firing of a transition takes no time.

Given a set of propositional variables PV, we introduce valuation function $V_n : \Sigma_n \to 2^{PV}$ which assigns the same propositions to the states with the same markings. Similarly as in the dense case, we assume the set PV to be such that each $q \in PV$ corresponds to exactly one $p \in P$, and use the same names for the propositions and the places. The function V_n is then defined by $p \in V_n(\sigma_n)$ iff $p \in m$ for each $\sigma_n = (m, \cdot)$. The structure $M_n(\mathcal{N}) = (T \cup \mathbb{N}, \Sigma_n, {\sigma_n}^0, \to_n, V_n)$ is a *discrete concrete model* of $\mathcal{N}$.

A *discrete* σ_n*-run* of TPN $\mathcal{N}$ is a (maximal) sequence of states: $\sigma_{n0} \stackrel{a_0}{\rightarrow}_n \sigma_{n1} \stackrel{a_1}{\rightarrow}_n \sigma_{n2} \stackrel{a_2}{\rightarrow}_n \ldots$, where $\sigma_{n0} = \sigma_n \in \Sigma_n$ and $a_i \in T \cup \mathbb{N}$ for each $i \geq 0$. A state σ_n is *reachable* in $M_n(\mathcal{N})$ iff there is a ${\sigma_n}^0$-run of $\mathcal{N}$ $\sigma_{n0} \stackrel{a_0}{\rightarrow}_n \sigma_{n1} \stackrel{a_1}{\rightarrow}_n \sigma_{n2} \stackrel{a_2}{\rightarrow}_n \ldots$ such that $\sigma_n = \sigma_{ni}$ for some $i \in \mathbb{N}$.

4 Temporal Logics ACTL* and ECTL*

In our work we deal with verification of properties of time Petri nets expressed in certain sublogics of the standard branching time logic CTL*. Below, we define the logics of our interest.

4.1 Syntax and Sublogics of CTL*

Let $PV = \{\wp_1, \wp_2 \ldots\}$ be a set of propositional variables. The language of CTL* is given as the set of all the state formulas φ_s (interpreted at states of a model), defined using path formulas φ_p (interpreted along paths of a model), by the following grammar:

$$\varphi_s := \wp \mid \neg\varphi_s \mid \varphi_s \wedge \varphi_s \mid \varphi_s \vee \varphi_s \mid \mathrm{A}\varphi_p \mid \mathrm{E}\varphi_p$$
$$\varphi_p := \varphi_s \mid \varphi_p \wedge \varphi_p \mid \varphi_p \vee \varphi_p \mid \mathrm{X}\varphi_p \mid \mathrm{U}(\varphi_p, \varphi_p) \mid \mathrm{R}(\varphi_p, \varphi_p).$$

In the above $\wp \in PV$, A ('for All paths') and E ('there Exists a path') are path quantifiers, whereas U ('Until') and R ('Release') are state operators. Intuitively, the formula $\mathrm{X}\varphi_p$ specifies that φ_p holds in the next state of the path, whereas $\mathrm{U}(\varphi_p, \psi_p)$ expresses that ψ_p eventually occurs and that φ_p holds continuously until then. The operator R is dual to U: the formula $\mathrm{R}(\varphi_p, \psi_p)$ says that either ψ_p holds always or it is released when φ_p eventually occurs. Derived operators are defined as $\mathrm{G}\varphi_p \stackrel{def}{=} \mathrm{R}(false, \varphi_p)$ and $\mathrm{F}\varphi_p \stackrel{def}{=} \mathrm{U}(true, \varphi_p)$, where $true \stackrel{def}{=} \wp \vee \neg\wp$, and $false \stackrel{def}{=} \wp \wedge \neg\wp$ for an arbitrary $\wp \in PV$. Intuitively, the formula $\mathrm{F}\varphi_p$ specifies that φ_p occurs in some state of the path ('Finally'), whereas $\mathrm{G}\varphi_p$ expresses that φ_p holds in all the states of the path ('Globally').

Next, we define some sublogics of CTL*:

ACTL* : the fragment of CTL* in which the state formulas are restricted such that negation can be applied to propositions only, and the existential quantifier E is not allowed,

ECTL* : the fragment of CTL* in which the state formulas are restricted such that negation can be applied to propositions only, and the universal quantifier A is not allowed,

ACTL : the fragment of ACTL* in which the temporal formulas are restricted to positive boolean combinations of $\mathrm{A}(\varphi\mathrm{U}\psi)$, $\mathrm{A}(\varphi\mathrm{R}\psi)$, and $\mathrm{AX}\varphi$ only.

ECTL : the fragment of ECTL* in which the temporal formulas are restricted to positive boolean combinations of $\mathrm{E}(\varphi\mathrm{U}\psi)$, $\mathrm{E}(\varphi\mathrm{R}\psi)$ and $\mathrm{EX}\varphi$ only.

$\mathbf{L}_{-\mathrm{X}}$ denotes the logic $\mathbf{L}$ without the next-step operator X.

4.2 Semantics of CTL*

Let PV be a set of propositions. A *model* for CTL* is a tuple $M = (L, S, s^0, \rightarrow, V)$, where L is a set of labels, S is a set of states, $s^0 \in S$ is the initial state, $\rightarrow \subseteq S \times L \times S$ is a total successor relation[5], and $V : S \longrightarrow 2^{PV}$ is a valuation function. For $s, s' \in S$ the notation $s \rightarrow s'$ means that there is $l \in L$ such that $s \stackrel{l}{\rightarrow} s'$. Moreover, for $s_0 \in S$ a *path* $\pi = (s_0, s_1, \ldots)$ is an infinite sequence of states in S starting at s_0, where $s_i \rightarrow s_{i+1}$ for all $i \geq 0$, $\pi_i = (s_i, s_{i+1}, \ldots)$ is the i-th suffix of π, and $\pi(i) = s_i$.

Given a model M, a state s, and a path π of M, by $M, s \models \varphi$ ($M, \pi \models \varphi$) we mean that φ holds in the state s (along the path π, respectively) of the model M. The model is sometimes omitted if it is clear from the context. The relation $\models$ is defined inductively as follows:

$$
\begin{array}{ll}
M, s \models \wp & \text{iff } \wp \in V(s), \text{ for } \wp \in PV, \\
M, s \models \neg\wp & \text{iff } M, s \not\models \wp, \text{ for } \wp \in PV, \\
M, x \models \varphi \wedge \psi & \text{iff } M, x \models \varphi \text{ and } M, x \models \psi, \text{ for } x \in \{s, \pi\}, \\
M, x \models \varphi \vee \psi & \text{iff } M, x \models \varphi \text{ or } M, x \models \psi, \text{ for } x \in \{s, \pi\}, \\
M, s \models \mathrm{A}\varphi & \text{iff } M, \pi \models \varphi \text{ for each path } \pi \text{ starting at } s, \\
M, s \models \mathrm{E}\varphi & \text{iff } M, \pi \models \varphi \text{ for some path } \pi \text{ starting at } s, \\
M, \pi \models \varphi & \text{iff } M, \pi(0) \models \varphi, \text{ for a state formula } \varphi, \\
M, \pi \models \mathrm{X}\varphi & \text{iff } M, \pi_1 \models \varphi, \\
M, \pi \models \varphi \mathrm{U} \psi & \text{iff } (\exists j \geq 0)\big(M, \pi_j \models \psi \text{ and } (\forall 0 \leq i < j)\ M, \pi_i \models \varphi\big), \\
M, \pi \models \varphi \mathrm{R} \psi & \text{iff } (\forall j \geq 0)\big(M, \pi_j \models \psi \text{ or } (\exists 0 \leq i < j)\ M, \pi_i \models \varphi\big).
\end{array}
$$

Moreover, we assume $M \models \varphi$ iff $M, s^0 \models \varphi$, where s^0 is the initial state of M.

4.3 Equivalence Preserving ACTL* and ECTL*

Let $M = (L, S, s_0, \rightarrow, V)$ and $M' = (L', S', s'_0, \rightarrow', V')$ be two models.

Definition 2 ([2]). *A relation* $\rightsquigarrow_{sim} \subseteq S' \times S$ *is a* simulation *from* M' *to* M *if the following conditions hold:*

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

The model M' simulates M ($M' \rightsquigarrow_{sim} M$) *if there is a simulation from* M' *to* M. *The models* M, M' *are* simulation equivalent *iff* $M \rightsquigarrow^1_{sim} M'$ *and* $M' \rightsquigarrow^2_{sim} M$ *for some simulations* $\rightsquigarrow^1_{sim} \subseteq S \times S'$ *and* $\rightsquigarrow^2_{sim} \subseteq S' \times S$. *Two models* M *and* M' *are called* bisimulation equivalent *if* $M' \rightsquigarrow_{sim} M$ *and* $M(\rightsquigarrow_{sim})^{-1} M'$, *where* $(\rightsquigarrow_{sim})^{-1}$ *is the inverse of* $\rightsquigarrow_{sim}$.

[5] Totality means that $(\forall s \in S)(\exists s' \in S)\ s \rightarrow s'$.

The following theorem holds:

Theorem 1 ([2]). *Let M, M' be two simulation equivalent models, where the range of the valuation functions V, V' is 2^{PV}. Then, $M, s_0 \models \varphi$ iff $M', s'_0 \models \varphi$, for any formula φ over PV such that $\varphi \in$ ACTL$^*\cup$ ECTL*.*

5 Discrete- vs. Dense-Time Verification for ACTL* and ECTL*

It is easy to see that both the models $M_r(\mathcal{N})$ and $M_n(\mathcal{N})$ can be used in ACTL* and ECTL* verification for a net with the semantics a given model corresponds to (as both meet the definition of the model for CTL*). However, it is also not difficult to see that the second model is smaller and less prone to the state explosion problem. The aim of our work is then to show that both the models are equivalent w.r.t. checking ACTL* and ECTL* properties of time Petri nets with the dense-time semantics. In our proof we make use of the approach of Popova presented in the paper [5].

Consider the dense concrete model $M_r(\mathcal{N}) = (T \cup \mathbb{R}_+, \Sigma, \sigma^0, \rightarrow_r, V)$ of a TPN $\mathcal{N}$. A state $\sigma = (m, clock) \in \Sigma$ is called an *integer-state* if $clock(t) \in \mathbb{N}$ for all $t \in T$. A *integer σ-run* of $\mathcal{N}$ is a sequence of states $\sigma_0 \stackrel{a_0}{\rightarrow}_r \sigma_1 \stackrel{a_1}{\rightarrow}_r \sigma_2 \stackrel{a_2}{\rightarrow}_r \ldots$, where $\sigma_0 = \sigma \in \Sigma$ and $a_i \in T \cup \mathbb{N}$ for each $i \geq 0$. Note that all the states of an integer-run which starts at an integer-state are integer-states as well. Thus, it is easy to see that the following holds:

Lemma 1. *For a given time Petri net $\mathcal{N}$ the model $M_r(\mathcal{N})$ reduced to the integer-states and the transition relation between them is equal to $M_n(\mathcal{N})$.*

Given a number $x \in \mathbb{R}_+$, let $\lfloor x \rfloor$ denote the *floor* of x, i.e., the greatest $a \in \mathbb{N}$ such that $a \leq x$, and let $\lceil x \rceil$ denote the *ceiling* of x, i.e. the smallest $a \in \mathbb{N}$ such that $x \leq a$. Moreover, let $fire(\sigma)$ denote a set of transition that are ready to fire in the state $\sigma \in \Sigma$, i.e., $fire(\sigma) = \{t \in en(m) \mid clock(t) \in [Eft(t), Lft(t)]\}$. We define the integer-states to be *neighbour states* of real-valued ones as follows:

Definition 3 (Neighbour states). *Let $\sigma = (m, clock)$ be a state of a TPN $\mathcal{N}$. An integer-state $\sigma' = (m', clock')$ is a* neighbour state of σ *(denoted $\sigma' \sim_n \sigma$) iff*

- $m' = m$,
- *for each $t \in en(m)$, $\lfloor clock(t) \rfloor \leq clock'(t) \leq \lceil clock(t) \rceil$.*

Intuitively, a neighbour state of σ is an integer-state of the same marking, and such that the values of its clocks, for all the enabled transitions, are "in a neighbourhood" of these of σ. However, it is easy to see that these values can be such that they make more transitions ready to fire than the corresponding values in σ do: each transition t which can be fired at a given value of $clock(t)$ can be fired both at $\lfloor clock(t) \rfloor$ and at $\lceil clock(t) \rceil$ since all these three values are either equal if $clock(t)$ is a natural number, or belong to the same (integer-bounded) interval

$[Eft(t), Lft(t)]$ if $clock(t) \notin \mathbb{N}$; on the other hand, a transition t' which is not ready to fire at $clock(t')$ can be firable at $\lceil clock(t') \rceil$ if $\lceil clock(t') \rceil = Eft(t')$. This implies $fire(\sigma) \subseteq fire(\sigma')$.

Let $\pi := \sigma_0 \stackrel{a_0}{\rightarrow}_r \sigma_1 \stackrel{a_1}{\rightarrow}_r \ldots$ be a σ^0-run in $M_r(\mathcal{N})$. By $\pi_{[k]}$, for $k \in \mathbb{N}$, we denote the prefix $\sigma_0 \stackrel{a_0}{\rightarrow}_r \sigma_1 \stackrel{a_1}{\rightarrow}_r \ldots \stackrel{a_{k-1}}{\rightarrow}_r \sigma_k$ of π, and by $\pi(k)$ - the k-th state of π, i.e., σ_k. Moreover, we assign a time δ_i to each step $\sigma_i \stackrel{a_i}{\rightarrow}_r \sigma_{i+1}$ in the run, i.e., define $\delta_i = a_i$ if $a_i \in \mathbb{R}$, and $\delta_i = 0$ otherwise. By $\Delta_G(\sigma_i, \pi)$, for $i \in \mathbb{N}$, we denote the value $\Sigma_{j=0}^{i-1}\delta_i$ (i.e., the time passed along π before reaching σ_i). Moreover, given $k \in \mathbb{N}$ and a $\pi(k)$-run $\rho := \sigma_k \stackrel{b_0}{\rightarrow}_r \beta_1 \stackrel{b_1}{\rightarrow}_r \beta_2 \stackrel{b_2}{\rightarrow}_r \ldots$, by $\pi_{[k]} \cdot \rho$ we denote the run $\sigma_0 \stackrel{a_0}{\rightarrow}_r \sigma_1 \stackrel{a_1}{\rightarrow}_r \ldots \stackrel{a_{k-1}}{\rightarrow}_r \sigma_k \stackrel{b_0}{\rightarrow}_r \beta_1 \stackrel{b_1}{\rightarrow}_r \beta_2 \stackrel{b_2}{\rightarrow}_r \ldots$ (i.e, the run obtained by "joining" $\pi_{[k]}$ and ρ). The above definitions apply to discrete runs in an analogous way. Next, we introduce the following definition (see also Fig. 2):

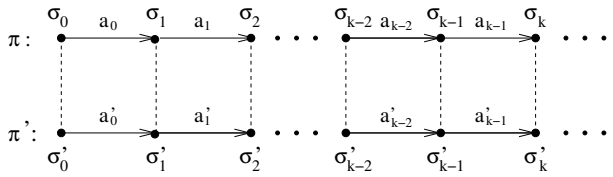

Fig. 2. Neighbour prefix of $\pi_{[k]}$ (denoted $\pi'_{[k]}$). If $a_i \in T$, then $a'_i = a_i$; the states of $\pi_{[k]}$ and $\pi'_{[k]}$ related by $\sim_n$ are linked by dashed lines.

Definition 4 (Neighbour prefix). *Let $\pi := \sigma_0 \stackrel{a_0}{\rightarrow}_r \sigma_1 \stackrel{a_1}{\rightarrow}_r \ldots$ be a run in $M_r(\mathcal{N})$, and let $\pi' := \sigma'_0 \stackrel{a'_0}{\rightarrow}_r \sigma'_1 \stackrel{a'_1}{\rightarrow}_r \ldots$ be an integer run. For $k \in \mathbb{N}$, the prefix $\pi'_{[k]}$ is a* neighbour prefix *of $\pi_{[k]}$ (denoted $\pi'_{[k]} \sim_n \pi_{[k]}$) iff for each $i = 0, \ldots, k$ it holds*

- $\sigma'_i \sim_n \sigma_i$,
- $a_i \in T$ *iff* $a'_i \in T$, *and if* $a_i, a'_i \in T$ *then* $a'_i = a_i$.

Intuitively, a neighbour prefix "visits" neighbour states of these in $\pi_{[k]}$, and the corresponding steps of these prefixes are either both firings of the same transition, or both passages of time (possibly of different lengths).

In order to show that $M_n(\mathcal{N})$ can replace $M_r(\mathcal{N})$ in ACTL*/ECTL* verification we shall prove the following lemma:

Lemma 2. *The models $M_r(\mathcal{N})$ and $M_n(\mathcal{N})$ are simulation equivalent.*

Proof. It is obvious from Lemma 1 that $M_r(\mathcal{N})$ simulates $M_n(\mathcal{N})$, with the relation $\mathcal{R}_1 \subseteq \Sigma \times \Sigma_n$ defined as $\mathcal{R}_1 = \{(\sigma, \sigma') \mid \sigma = \sigma'\}$.

Let $R_r(\mathcal{N})$ and $R_n(\mathcal{N})$ denote respectively the sets of all the dense ${\sigma^0}$-runs (discrete ${\sigma_n}^0$-runs) of the net $\mathcal{N}$. In order to prove that $M_n(\mathcal{N}) \rightsquigarrow_{sim} M_r(\mathcal{N})$ we shall show that the relation $\mathcal{R} \subseteq \Sigma_n \times \Sigma$ given by

$$\mathcal{R} = \{(\sigma', \sigma) \mid \exists \pi \in R_r(\mathcal{N})\ \exists \pi' \in R_n(\mathcal{N})\ \exists k \in \mathbb{N} \text{ s.t.}$$
$$\sigma = \pi(k) \wedge \sigma' = \pi'(k) \wedge \pi'_{[k]} \sim_n \pi_{[k]} \wedge \forall_{j \leq k}\ \Delta_G(\sigma'_j, \pi') = \lfloor \Delta_G(\sigma_j, \pi) \rfloor\}$$

is a simulation from $M_n(\mathcal{N})$ to $M_r(\mathcal{N})$. Intuitively, the states σ, σ' are related by $\mathcal{R}$ if they both are reachable from the initial state of $\mathcal{N}$ in k steps for some natural k, on runs π, π' such that $\pi'_{[k]}$ is a neighbour prefix of $\pi_{[k]}$ and for each $j \leq k$ the total time passed along $\pi'_{[j]}$ is the floor of that passed along $\pi_{[j]}$.

It is obvious that $({\sigma_n}^0, \sigma^0) \in \mathcal{R}$ due to equality of these states. Next, consider σ, σ' such that $(\sigma', \sigma) \in \mathcal{R}$. Assume that the runs "justifying" this relation (for some k) are of the form $\pi := \sigma_0 \xrightarrow{a_0}_r \sigma_1 \xrightarrow{a_1}_r \ldots$ and $\pi' := \sigma'^0_n = \sigma'_0 \xrightarrow{a'_0}_n \sigma'_1 \xrightarrow{a'_1}_n \ldots$ respectively, and that $\sigma_i = (m_i, clock_i)$, $\sigma'_i = (m'_i, clock'_i)$ for each $i \in \mathbb{N}$ (which implies also the notation $\sigma = (m_k, clock_k)$ and $\sigma' = (m'_k, clock'_k)$ used below).

– if $\sigma \xrightarrow{t}_r \gamma$ for a transition $t \in T$ and a state $\gamma = (m_\gamma, clock_\gamma)$, then from $\sigma' \sim_n \sigma$ (and therefore $fire(\sigma) \subseteq fire(\sigma')$) the transition t can be fired at σ' as well, leading to a state $\xi = (m_\xi, clock'_\xi)$. Let ρ be a σ-run of the form $\sigma \xrightarrow{t}_r \gamma \rightarrow_r \ldots$ (i.e., a σ-run whose first step is $\sigma \xrightarrow{t}_r \gamma$), and let ρ' be a σ'-run of the form $\sigma' \xrightarrow{t}_n \xi \rightarrow_n \ldots$ (i.e., a σ'-run whose first step is $\sigma' \xrightarrow{t}_n \xi$; see Fig. 3). We shall show that $(\pi'_{[k]} \cdot \rho')_{[k+1]} \sim_n (\pi_{[k]} \cdot \rho)_{[k+1]}$ and

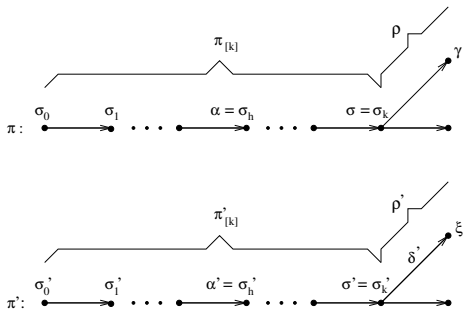

Fig. 3. Relation between π, π', ρ and ρ' in the proof of Lemma 2.

that $\Delta_G(\xi, \pi'_{[k]} \cdot \rho') = \lfloor \Delta_G(\gamma, \pi_{[k]} \cdot \rho) \rfloor$.

- In order to prove $(\pi'_{[k]} \cdot \rho')_{[k+1]} \sim_n (\pi_{[k]} \cdot \rho)_{[k+1]}$ it is sufficient to show that $\xi \sim_n \gamma$. It is obvious that the markings m_γ and m_ξ are equal, and that $newly\_en(m_k, t) = newly\_en(m'_k, t)$. Next, consider $t' \in en(m_\gamma)$. If $t' \notin newly\_en(m_k, t)$ then the value of its clock in γ is the same as in σ (since firing of t does not influence the value of the clock of t'). In turn, if

$t' \in newly\_en(m_k, t)$ then the values of its clock in γ and in ξ are equal to 0. Thus, from the fact that for σ, σ' we have $\lfloor clock_k(t) \rfloor \leq clock'_k(t) \leq \lceil clock_k(t) \rceil$ we have also $\lfloor clock_\gamma(t) \rfloor \leq clock'_\xi(t) \leq \lceil clock_\gamma(t) \rceil$, which implies $\xi \sim_n \gamma$.

- the condition $\Delta_G(\xi, \pi'_{[k]} \cdot \rho') = \lfloor \Delta_G(\gamma, \pi_{[k]} \cdot \rho) \rfloor$ holds in an obvious way ($\Delta_G(\xi, \pi'_{[k]} \cdot \rho') = \Delta_G(\sigma', \pi') = \lfloor \Delta_G(\sigma, \pi) \rfloor = \lfloor \Delta_G(\gamma, \pi_{[k]} \cdot \rho) \rfloor$ as the step consisting in firing a transition is assigned the time 0).

– if $\sigma \xrightarrow{\delta}_r \gamma$ for a time $\delta \in \mathbb{R}_+$ and a state $\gamma = (m_\gamma, clock_\gamma)$, then let ρ be a σ-run $\sigma \xrightarrow{\delta}_r \gamma \dot{\to}_r \ldots$ (i.e., a σ-run of the first step $\sigma \xrightarrow{\delta}_r \gamma$; see Fig. 3), and let $\pi_{[k]} \cdot \rho$ denote the run $\sigma_0 \xrightarrow{a_0}_r \sigma_1 \xrightarrow{a_1}_r \ldots \xrightarrow{a_{k-1}}_r \sigma_k \xrightarrow{\delta}_r \gamma \to_r \ldots$ (i.e, the run obtained by "joining" $\pi_{[k]}$ and ρ). Next, assume

$$\delta' = \lfloor \Delta_G(\gamma, \pi_{[k]} \cdot \rho) \rfloor - \Delta_G(\sigma', \pi')$$

(which is an integer value due to $\Delta_G(\sigma', \pi') \in \mathbb{N}$). We shall show first that the time δ' can pass at σ', leading to a state $\xi = (m_\xi, clock'_\xi)$.

- To show that δ' can pass at σ' notice that

$$\delta = \Delta_G(\gamma, \pi_{[k]} \cdot \rho) - \Delta_G(\sigma, \pi_{[k]} \cdot \rho),$$

and that

$$\Delta_G(\sigma_i, \pi) = \Delta_G(\sigma_i, \pi_{[k]} \cdot \rho) \text{ for each } i = 0, \ldots, k.$$

Moreover, we have that $clock_\gamma(t) = clock_k(t) + \delta \leq Lft(t)$ for each $t \in en(m_k)$.
Consider a transition $t \in en(m'_k)$ (where $en(m'_k) = en(m_k) = en(m_\gamma)$). Let h be an index along $\pi_{[k]}$ pointing to a state (denoted α) at which t became enabled most recently, and let h' be an index along $\pi'_{[k]}$ pointing to a state (denoted α') at which t became enabled most recently. From the fact that $\pi'_{[k]} \sim_n \pi_{[k]}$ we have $h = h'$ (for each $j \leq k-1$ the corresponding j-th steps of $\pi_{[k]}$ and $\pi'_{[k]}$ are either both firings of the same transition or both time passings, which implies that for each $i \leq k$ a transition t becomes enabled in $\pi(i)$ iff it becomes enabled in $\pi'(i)$). From the definitions of $clock$, $clock'$ it is easy to see that

$$clock_k(t) = \Delta_G(\sigma, \pi) - \Delta_G(\alpha, \pi),$$

$$clock_\gamma(t) = \Delta_G(\gamma, \pi_{[k]} \cdot \rho) - \Delta_G(\alpha, \pi)$$

and

$$clock'_k(t) = \Delta_G(\sigma', \pi') - \Delta_G(\alpha', \pi')$$

Moreover, it holds
$clock'_k(t) + \delta' = \Delta_G(\sigma', \pi') - \Delta_G(\alpha', \pi') + \delta' =$
$\Delta_G(\sigma', \pi') - \Delta_G(\alpha', \pi') + \lfloor \Delta_G(\gamma, \pi_{[k]} \cdot \rho) \rfloor - \Delta_G(\sigma', \pi') =$
$\lfloor \Delta_G(\gamma, \pi_{[k]} \cdot \rho) \rfloor - \Delta_G(\alpha', \pi') \overset{def.\ of\ \mathcal{R}\ and\ h=h'}{=} \lfloor \Delta_G(\gamma, \pi_{[k]} \cdot \rho) \rfloor - \lfloor \Delta_G(\alpha, \pi) \rfloor.$

From $clock_\gamma(t) \leq Lft(t)$ we have $\lceil clock_\gamma(t) \rceil \leq Lft(t)$, and from the property $\lfloor a \rfloor - \lfloor b \rfloor \leq \lceil a - b \rceil$ we get
$clock'_k(t) + \delta' = \lfloor \Delta_G(\gamma, \pi_{[k]} \cdot \rho) \rfloor - \lfloor \Delta_G(\alpha, \pi) \rfloor \leq \lceil \Delta_G(\gamma, \pi_{[k]} \cdot \rho) - \Delta_G(\alpha, \pi) \rceil = \lceil clock_\gamma(t) \rceil \leq Lft(t)$;
Thus, we have that $clock'_k(t) + \delta' \leq Lft(t)$ for each $t \in en(m'_k)$, and therefore the time δ' can pass at σ'.

Next, let ρ' be a σ'-run of the form $\sigma' \xrightarrow{\delta'}_n \xi \rightarrow_n \ldots$. We shall show that $(\pi'_{[k]} \cdot \rho')_{[k+1]} \sim_n (\pi_{[k]} \cdot \rho)_{[k+1]}$ and that $\Delta_G(\xi, \pi'_{[k]} \cdot \rho') = \lfloor \Delta_G(\gamma, \pi_{[k]} \cdot \rho) \rfloor$.

- In order to prove $(\pi'_{[k]} \cdot \rho')_{[k+1]} \sim_n (\pi_{[k]} \cdot \rho)_{[k+1]}$ it is sufficient to show that $\xi \sim_n \gamma$. It is obvious that the markings of these states are equal. Consider $t \in en(m)$. We show that $\lfloor clock_\gamma(t) \rfloor \leq clock'_\xi(t) \leq \lceil clock_\gamma(t) \rceil$. Let α, α', h, h' be defined as in the previous part of the proof (see the 8th line of the previous item). Similarly as before, from the definitions of $clock$, $clock'$ we have that

$$clock_\gamma(t) = \Delta_G(\gamma, \pi_{[k]} \cdot \rho) - \Delta_G(\alpha, \pi),$$

and that

$$clock'_\xi(t) = \Delta_G(\sigma', \pi') + \delta' - \Delta_G(\alpha', \pi').$$

  - From the property $\lfloor a - b \rfloor \leq \lfloor a \rfloor - \lfloor b \rfloor$ (for $a, b \in \mathbb{R}_+$ with $a \geq b$) we have
$\lfloor clock_\gamma(t) \rfloor = \lfloor \Delta_G(\gamma, \pi_{[k]} \cdot \rho) - \Delta_G(\alpha, \pi) \rfloor \leq \lfloor \Delta_G(\gamma, \pi_{[k]} \cdot \rho) \rfloor - \lfloor \Delta_G(\alpha, \pi) \rfloor \overset{h=h' \text{and def. of } \mathcal{R}}{=} \lfloor \Delta_G(\gamma, \pi_{[k]} \cdot \rho) - \Delta_G(\sigma', \pi') + \Delta_G(\sigma', \pi') \rfloor - \Delta_G(\alpha', \pi') = \lfloor \Delta_G(\gamma, \pi_{[k]} \cdot \rho) \rfloor - \Delta_G(\sigma', \pi') + \Delta_G(\sigma', \pi') - \Delta_G(\alpha', \pi') = \delta' + \Delta_G(\sigma', \pi') - \Delta_G(\alpha', \pi') = clock'_\xi(t)$.
  - From the property $\lfloor a \rfloor - \lfloor b \rfloor \leq \lceil a - b \rceil$ we have
$clock'_k(t) + \delta' = \lfloor \Delta_G(\gamma, \pi_{[k]} \cdot \rho) \rfloor - \lfloor \Delta_G(\alpha, \pi) \rfloor \leq \lceil \Delta_G(\gamma, \pi_{[k]} \cdot \rho) - \Delta_G(\alpha, \pi) \rceil = \lceil clock_\gamma(t) \rceil$
- Next, we have $\Delta_G(\xi, \pi'_{[k]} \cdot \rho') = \Delta_G(\sigma', \pi') + \delta' = \Delta_G(\sigma', \pi') + \lfloor \Delta_G(\gamma, \pi_{[k]} \cdot \rho) \rfloor - \Delta_G(\sigma', \pi') = \lfloor \Delta_G(\gamma, \pi_{[k]} \cdot \rho) \rfloor$, which ends the proof.

Therefore, we can formulate the following theorem:

Theorem 2. *Let $M_r(\mathcal{N})$ and $M_n(\mathcal{N})$ be respectively a dense and a discrete model for a time Petri net $\mathcal{N}$, and let φ be an* ACTL* *(*ECTL**) formula. The following condition holds:*

$$M_r(\mathcal{N}) \models \varphi \textit{ iff } M_n(\mathcal{N}) \models \varphi.$$

Proof. Follows from Theorem 1 and Lemma 2 in a straightforward way.

It should also be explained that in the case of timed systems (and therefore also TPNs) with the dense-time semantics, logics without the next-step operator are usually used, due to problems with intepreting the "next" step in the case of continuous time. However, Thm. 2 considers more general logics, in case one would interpret the next-step operator over an arbitrary passage of time.

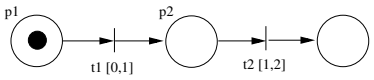

Fig. 4. A net

It should be noticed that the relation $\mathcal{R}$ used in our proof cannot be used to prove bisimulation between the models, i.e., their equivalence w.r.t. the CTL* properties, since the integer run π' "justifying" $\sigma'\mathcal{R}\sigma$ and the dense run π occurring in the relation do not need to "branch" in the same way. Thus, although one can prove that each transition t which can be fired at σ can be fired at σ' as well, the reverse does not hold. To see an example of the above consider the net shown in Fig. 4 and its runs:

- the dense one:
  $\pi := (p_1,(0,0)) \overset{0.5}{\to}_r (p1,(0.5,0)) \overset{t1}{\to}_r (p2,(0,0)) \overset{0.6}{\to}_r (p2,(0,0.6)) \to_r \ldots$
- and the discrete one (denoted π'), built in the way shown in [5] and used in our proof in the definition of $\mathcal{R}$ (i.e., satisfying $\pi'_{[3]} \sim_n \pi_{[3]}$ and $\Delta_G(\sigma'_j,\pi') = \lfloor \Delta_G(\sigma_j,\pi) \rfloor$ for each $j \leq 3$):
  $\pi' := (p_1,(0,0)) \overset{0}{\to}_n (p1,(0,0)) \overset{t1}{\to}_n (p2,(0,0)) \overset{1}{\to}_n (p2,(0,1)) \to_n \ldots.$

It is easy to see that in $\pi(3)$ we have $clock(t2) = 0.6$, which means that $t2$ cannot be fired at this state, while in $\pi'(3)$ we have $clock(t2) = 1$, which means that the transition $t2$ is firable.

6 Experimental Results

In order to show that using discrete-time models instead of the dense ones can be profitable, we performed some tests, using as an example an implementation of SAT-based bounded model checking (BMC) for a subclass of TPNs (i.e., distributed time Petri nets) with the discrete-time semantics and the logic ACTL$_{-\mathrm{X}}$ used in [4], and its modification for the dense-time case prepared for the current paper. BMC is a technique applied mainly to searching for counterexamples for universal properties, using a model truncated up to some specific depth k. The formulas used by the method are then negations of these expressing properties to be tested. So, in our case they are formulas of ECTL$_{-\mathrm{X}}$.

The first system we consider is the Generic Pipeline Paradigm Petri net model (GTPP) shown in Fig. 5. It consists of three parts: Producer producing data ($ProdReady$) or being inactive, Consumer receiving data ($ConsReady$) or being inactive, and a chain of n intermediate Nodes which can be ready for receiving data ($Node_iReady$), processing data ($Node_iProc$), or sending data ($Node_iSend$). The example can be scaled by adding more intermediate nodes. The parameters a, b, c, d, e, f are used to adjust the time properties of Producer, Consumer, and of the intermediate Nodes. The formulas considered are EGEF$ConsReceived$, EG($ProdReady \vee ConsReady$) and EF$Node_1Send$.

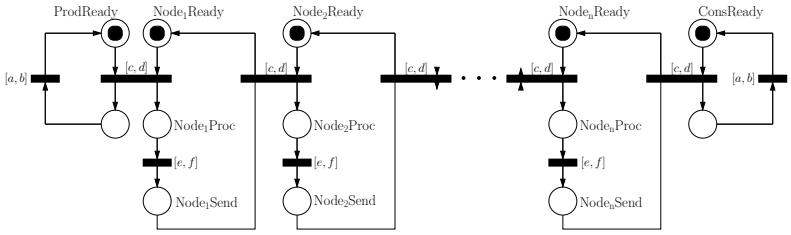

Fig. 5. A net for Generic Timed Pipeline Paradigm

The next system tested is the standard *Fischer's mutual exclusion protocol* (Mutex). The system consists of n time Petri nets, each one modelling a process, plus one additional net used to coordinate the access of the processes to the critical sections. A TPN modelling the system for $n = 2$ is presented in Fig. 6 In this case we have tested the formula $\mathrm{EGEF}(crit_1 \vee \ldots \vee crit_n)$.

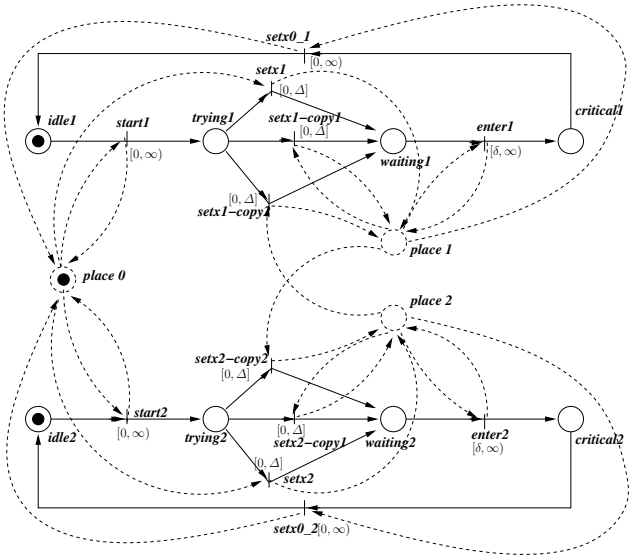

Fig. 6. A net for Fischer's mutual exclusion protocol for $n = 2$

The results are presented in Fig. 7–9 for GTPP, and in Fig. 10 for Mutex. It can be seen that in all the cases we are able to verify systems containing more components (indicated in the column n) than when discrete models are used, and the total time ($bmcT + satT$) and the memory required ($\max(bmcM, satM)$) are usually smaller for the discrete-time case (the columns with "$\mathbb{N}$:"). In some cases the differences are quite substantial, but there are also examples in which the time and the memory used are similar for both the semantics. However, one

n	k	LL	$\mathbb{R}$: bmcT+satT	$\mathbb{R}$: max(bmcM,satM)	$\mathbb{N}$: bmcT+satT	$\mathbb{N}$: max(bmcM,satM)
1	5	7	1.41	12.00	0.20	8.00
2	7	9	9.94	25.00	1.15	12.00
3	9	11	49.45	60.00	3.55	21.00
4	11	13	154.70	146.00	9.94	38.00
5	13	15	310.18	243.00	20.90	61.00
6	15	17	708.66	313.00	41.43	94.00
7	17	19	1934.63	818.00	76.81	145.00
8	19	21	4121.60	1071.00	131.98	215.00
9	21	23	6819.25	1640.00	237.21	314.00
10	23	25	20519.20	3455.00	361.03	377.00
11	25	27	-	-	562.15	552.00

Fig. 7. Comparison of the results for GTPP and the formula EGEF*ConsReceived*

n	k	LL	$\mathbb{R}$: bmcT+satT	$\mathbb{R}$: max(bmcM,satM)	$\mathbb{N}$: bmcT+satT	$\mathbb{N}$: max(bmcM,satM)
1	5	1	0.41	7.00	0.17	7.00
2	7	1	4.14	12.00	0.76	8.00
3	9	1	32.27	29.00	2.20	9.00
4	11	1	63.28	52.00	8.57	12.00
5	13	1	200.14	151.00	21.14	17.00
6	15	1	488.59	165.00	43.18	24.00
7	17	1	870.21	342.00	105.18	38.00
8	19	1	1870.65	415.00	234.00	54.00
9	21	1	3745.33	658.00	763.84	139.00
10	23	1	7097.01	1364.00	1696.58	283.00
11	25	1	-	-	3013.98	306.00

Fig. 8. Comparison of the results: GTPP, the formula EG($ProdReady \vee ConsReady$)

can see that the noticeable differences occur in the cases in which the length of the witness for the formula (k) or the number of paths required to check this formula (LL) grow together with the size of the system, making the verification more expensive.

7 Conclusions and Further Work

We have shown that the result of Popova, stating that integer time steps are sufficient to test reachability of markings in time Petri nets, can be extended to testing ECTL* and ACTL* properties. We have focused on 1-safe TPNs for simplicity of the presentation, but it is easy to see that the result applies also to "general" time Petri nets: neither the definitions of a marking and of enabledness of a transition, nor the way multiple enabledness of transitions is handled do influence the proof.

n	k	LL	$\mathbb{R}$: bmcT+satT	$\mathbb{R}$: max(bmcM,satM)	$\mathbb{N}$: bmcT+satT	$\mathbb{N}$: max(bmcM,satM)
100	2	1	2.02	23.00	1.60	19.00
200	2	1	7.04	76.00	5.74	51.00
300	2	1	15.03	153.00	12.15	102.00
400	2	1	26.30	270.00	18.59	179.00
500	2	1	40.50	412.00	28.47	273.00
600	2	1	58.71	563.00	40.45	397.00
700	2	1	79.71	738.00	54.69	537.00
800	2	1	104.84	992.00	72.06	654.00
900	2	1	133.78	1173.00	90.48	854.00
1000	2	1	169.19	1528.00	114.86	1005.00
1100	2	1	211.16	1772.00	140.22	1168.00
1200	2	1	-	-	168.86	1506.00
1300	2	1	-	-	203.24	1604.00

Fig. 9. Comparison of the results: GTPP, the formula EF$Node1Send$

n	k	LL	$\mathbb{R}$: bmcT+satT	$\mathbb{R}$: max(bmcM,satM)	$\mathbb{N}$: bmcT+satT	$\mathbb{N}$: max(bmcM,satM)
2	4	5	1.04	11.00	0.74	10.00
3	4	5	1.77	13.00	1.10	12.00
4	4	5	1.83	15.00	1.63	14.00
5	4	5	3.18	17.00	2.41	16.00
10	4	5	9.15	40.00	7.20	31.00
20	4	5	26.14	90.00	18.76	86.00
30	4	5	74.70	177.00	58.56	161.00
40	4	5	258.34	330.00	108.34	320.00
50	4	5	265.06	419.00	170.93	358.00
60	4	5	710.11	732.00	442.42	713.00
70	4	5	701.81	1092.00	728.20	1073.00
80	4	5	919.34	1001.00	2288.34	1349.00
90	4	5	780.89	1161.00	934.72	1140.00
100	4	5	4566.16	3181.00	4230.64	4549.00
110	4	5	4260.76	3414.00	4956.38	3237.00
120	4	5	-	-	4217.44	3238.00
130	4	5	-	-	2155.04	2571.00
140	4	5	-	-	5087.76	3603.00

Fig. 10. Comparison of the results: mutex, the formula EGEF($crit_1 \vee \ldots \vee crit_n$)

Our experimental results show that considering the discrete semantics while verifying properties of dense-time nets can be profitable. Due to this, in our further work we are going to check whether discrete-time semantics can be used when testing other classes of properties of the dense-time Petri net systems (e.g., $\mathrm{CTL}^*_{-\mathrm{X}}$).

References

1. U. Goltz, R. Kuiper, and W. Penczek. Propositional temporal logics and equivalences. In *Proc. of the 3rd Int. Conf. on Concurrency Theory (CONCUR'92)*, volume 630 of *LNCS*, pages 222–236. Springer-Verlag, 1992.
2. O. Grumberg and D. E. Long. Model checking and modular verification. In *Proc. of the 2nd Int. Conf. on Concurrency Theory (CONCUR'91)*, volume 527 of *LNCS*, pages 250–265. Springer-Verlag, 1991.
3. A. Janowska, W. Penczek, A. Półrola, and A. Zbrzezny. Towards discrete-time verification of time Petri nets with dense-time semantics. In *Proc. of the Int. Workshop on Concurrency, Specification and Programming (CS&P'11)*, pages 215–228. Bialystok University of Technology, 2011.
4. A. Męski, W. Penczek, A. Półrola, B. Woźna-Szcześniak, and A. Zbrzezny. Bounded model checking approaches for verificaton of distributed time Petri nets. In *Proc. of the Int. Workshop on Petri Nets and Software Engineering (PNSE'11)*, pages 72–91. University of Hamburg, 2011.
5. L. Popova. On time Petri nets. *Elektronische Informationsverarbeitung und Kybernetik*, 27(4):227–244, 1991.
6. L. Popova-Zeugmann. Essential states in time Petri nets. Informatik-Bericht 96, Humboldt University, 1998.
7. L. Popova-Zeugmann. Time Petri nets state space reduction using dynamic programming. *Journal of Control and Cybernetics*, 35(3):721–748, 2006.
8. L. Popova-Zeugmann and D. Schlatter. Analyzing paths in time Petri nets. *Fundamenta Informaticae*, 37(3):311–327, 1999.