SMT-Based Reachability Checking for Bounded Time Petri Nets

Extended Abstract

A. Półrola, P. Cybula, and A. Męski

[1] University of Łódź, FMCS, Banacha 22, 90-238 Łódź, Poland
{polrola,cybula}@math.uni.lodz.pl
[2] Institute of Computer Science, PAS, Jana Kazimierza 5, 01-248 Warsaw, Poland
meski@ipipan.waw.pl

Abstract. Time Petri nets by Merlin and Farber are a powerful modelling formalism. However, symbolic model checking methods for them consider in most cases the nets which are 1-safe, i.e., allow the places to contain at most one token. In our paper we present a preliminary version of the approach aimed at testing reachability for time Petri nets without this restriction. We deal with the class of bounded nets restricted to disallow multiple enabledness of transitions, and present the method of reachability testing based on a translation into a satisfiability modulo theory (SMT).

1 Introduction

The process of design and production of both systems and software – among them, the concurrent ones – involves testing whether the product conforms to its specification. This can be achieved using various kinds of formal methods. However, in order to apply any of these methods, the system to be tested needs to be modelled using an appropriate formalism. One of such formalisms, applicable to modelling systems with time dependencies, are time Petri nets by Merlin and Farber [1]. Numerous subclasses of time Petri nets have been proposed, i.e., 1-safe, bounded, unbounded, distributed nets, and many others.

For concurrent systems, the size of the state-space of the analysed system is a significant factor in the efficiency of the verification. The fact that verification methods need to explore the reachable state-space can lead to the state-space explosion problem. This follows from the fact that the size of the model grows exponentially with the number of the components of the system. Therefore, the development of methods that alleviate this problem is considered an important research subject.

There exist many papers dealing with model checking time Petri nets [2–11]. Most of them describe techniques based on explicitly-represented abstractions of the state spaces. The developed fully symbolic methods include decision diagrams-based ones: reachability verification for Integer Timed Petri Nets [10] as well as LTL and ECTL verification for (1-safe) Distributed Time Petri Nets [6], and SAT-based methods for the distributed nets [6, 8].

This paper presents our first attempt to apply a satisfiability modulo theory (SMT) to verification of (bounded) time Petri nets, i.e., reachability testing for the nets restricted to disallow multiple enabledness of transitions. We use a bounded model checking technique (BMC), i.e., consider models truncated up to a specific depth, and transform the reachability problem into a test of satisfiability of a set of (integer) inequalities. Although the current version of the method was implemented, and we provide some preliminary experimental results, we consider our work to be in progress, and we plan on improving the efficiency of our implementation, as well as to extend it in a way allowing to test more involved reachability-related properties and to handle bounded nets with multiple enabledness of transitions [12].

The rest of the paper is organised as follows: in Sec. 2 we introduce bounded time Petri nets, their concrete state spaces and concrete models. The next Sec. 3 discusses reachability verification using SMT. In Sec. 4 and Sec. 5 we provide some preliminary experimental results and concluding remarks.

2 Time Petri Nets

Let $\mathbb{N}$ ($\mathbb{N}_+$) denote the set of (nonnegative) integers, and $\mathbb{R}$ ($\mathbb{R}_+$) - the set of (nonnegative) reals. We start with a definition of time Petri nets:

Definition 1. *A* time Petri net *(TPN for short) is a tuple* $\mathcal{N} = (P, T, F, cap, w, m^0, Eft, Lft)$*, 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 transition s.t.* $P \cap T = \emptyset$*,*
- $F \subseteq (P \times T) \cup (T \times P)$ *is a flow relation,*
- $cap : P \to \mathbb{N}_+ \cup \{\infty\}$ *is a partial capacity restriction,*
- $w : F \to \mathbb{N}_+$ *is an arc weight function,*
- $m^0 : P \to \mathbb{N}$ *with* $m^0(p) \leq cap(p)$ *for each* $p \in P$ *is an initial marking,*
- $Eft : T \to \mathbb{N}$*,* $Lft : T \to \mathbb{N} \cup \{\infty\}$ *are functions describing the earliest and the latest firing time of a transition, where for each* $t \in T$ *we have* $Eft(t) \leq Lft(t)$*.*

A TPN $\mathcal{N}$ is called *k-bounded*, for some $k \in \mathbb{N}_+$, if $cap(p) \leq k$ for each $p \in P$, and is called *bounded* if there is $k \in \mathbb{N}$ such that $\mathcal{N}$ is k-bounded. The value k is called a *bound* of $\mathcal{N}$. In what follows we consider bounded time Petri nets only.

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. Moreover, we restrict to the nets satisfying the condition $\forall_{p \in P}\ cap(p) < 2 \cdot \min\{w(p, t) \mid t \in T \text{ s.t. } p \in \bullet t\}$ which prevents multiple enabledness of transitions.

We introduce the following notations and definitions:

- a *marking* of $\mathcal{N}$ is a function $m : P \to \mathbb{N}$,
- a transition $t \in T$ is *enabled* at a marking m ($m[t\rangle$ for short) if
  - $\star$ for each $p \in \bullet t$ it holds $w(p,t) \leq m(p)$, and
  - $\star$ for each $p \in t\bullet$ it holds $m(p) - w(p,t) + w(t,p) \leq cap(p)$ if $p \in \bullet t \cap t\bullet$, and $m(p) + w(t,p) \leq cap(p)$ otherwise,
- $en(m) = \{t \in T \mid m[t\rangle\}$ is the set of all the transitions enabled in a marking m of $\mathcal{N}$,
- the marking m' obtained by firing $t \in en(m)$ at m is given by
$$m'(p) = \begin{cases} m(p) & \text{if } (p,t) \notin F \wedge (t,p) \notin F; \\ m(p) - w(p,t) & \text{if } (p,t) \in F \wedge (t,p) \notin F \\ m(p) + w(t,p) & \text{if } (p,t) \notin F \wedge (t,p) \in F \\ m(p) - w(p,t) + w(t,p) & \text{if } (p,t) \in F \wedge (t,p) \in F. \end{cases}$$
The marking is denoted by $m[t\rangle$ as well, if it does not lead to misunderstanding,
- $newly\_en(m,t) = \{u \in T \mid u \in en(m[t\rangle) \ \wedge \ (\exists_{p \in (t\bullet \setminus \bullet t) \cap \bullet u} m(p) < w(p,u) \vee \exists_{p \in \bullet t \cap t\bullet \cap \bullet u} m(p) - w(p,t) < w(p,u) \vee \exists_{p \in u\bullet \cap \bullet t} m(p) + w(u,p) > cap(p))\}$ is a set of transitions which became (newly) enabled by firing t at m.

2.1 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 further behaviour of the net). A *concrete state* σ of a TPN $\mathcal{N}$ is thus a pair $(m, clock)$, where $m : P \to \mathbb{N}$ 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. The set of all the states of $\mathcal{N}$ is denoted by Σ. The initial state of $\mathcal{N}$ is $\sigma^0 = (m^0, clock^0)$, where m^0 is the initial marking of $\mathcal{N}$, and $clock^0(t) = 0$ for each $t \in T$.

For $\delta \in \mathbb{R}_+$, let $clock + \delta$ denote the function given by $(clock + \delta)(t) = clock(t) + \delta$, and let $(m, clock) + \delta$ denote $(m, clock + \delta)$. The states of $\mathcal{N}$ can change due to a passage of time or due to a firing of a transition. The transition relation $\to \subseteq \Sigma \times (T \cup \mathbb{R}_+) \times \Sigma$ of the net $\mathcal{N}$ is thus given by:

- in a state $\sigma = (m, clock)$ a time $\delta \in \mathbb{R}_+$ can pass leading to a new state $\sigma' = (m, clock')$ (denoted $\sigma \xrightarrow{\delta} \sigma'$) iff $(clock + \delta)(t) \leq Lft(t)$ for each $t \in en(m)$ (*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 $\sigma \xrightarrow{t} \sigma'$) if $t \in en(m)$, $Eft(t) \leq clock(t) \leq Lft(t)$, $m' = m[t\rangle$, 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 relation*).

Intuitively, the time successor relation does not change the marking of the net, but increases the clocks of all the transitions, provided no enabled transition becomes disabled by passage of time. Firing of a transition t takes no time (the only change it introduces to the clocks is reseting these related to the newly enabled transitions) and is allowed provided that t is enabled and its clock is not smaller than $Eft(t)$ and not greater than $Lft(t)$. The structure $(\Sigma, \sigma^0, \rightarrow)$ is called a *concrete state space* of $\mathcal{N}$.

A *timed run* of $\mathcal{N}$ starting at a state $\sigma_0 \in \Sigma$ (σ_0-*run*) is a maximal sequence of concrete states, transitions and time passings $\rho = \sigma_0 \overset{a_0}{\rightarrow} \sigma_1 \overset{a_1}{\rightarrow} \ldots$, where $\sigma_i \in \Sigma$, and $a_i \in T \cup \mathbb{R}_+$ for all $i \in \mathbb{N}$. A state $\sigma_\star$ is reachable if there exists a σ^0-run ρ and $i \in \mathbb{N}$ such that $\sigma_\star = \sigma_i$, where σ_i is an element of ρ. 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* $V : \Sigma \rightarrow 2^{PV}$ 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 an (in)equality $I(q)$ of the form $m(p) \sim a$ or $m(p) \oplus m(p') \sim a$, where $p, p' \in P$, $\sim \in \{\leq, <, =, >, \geq\}$, $\oplus \in \{+, -\}$, and $a \in \{0, 1, \ldots, 2k\}$, where k is a bound of $\mathcal{N}$. The function V is defined by $q \in V((m, clock))$ iff $I(q)$ holds for m. The structure $M(\mathcal{N}) = (\Sigma, \sigma^0, \rightarrow, V)$ is called a *concrete model of* $\mathcal{N}$.

3 Testing Reachability

The reachability problem for a time Petri net $\mathcal{N}$ consists in checking, given a property φ, whether $\mathcal{N}$ can ever be in a state where φ holds. The property is expressed in terms of propositional variables. Therefore, the problem can be translated into testing whether the set $Reach_{\mathcal{N}}$ contains a marking of certain features expressed by φ. Checking this can be performed by an explicit exploration of the concrete model for $\mathcal{N}$, but such an approach is usually very inefficient.

If a reachable state satisfying φ exists then proving this can be done using a part of the model only. This enables us to apply the bounded model checking method [13]. The main idea of testing reachability using BMC consists in searching for an *reachability witness* of a bounded length l (i.e., for a path leading from the initial state to a state satisfying φ). One of the possible approaches to this problem consists in generating a logical formula satisfiable iff such a witness exists, and checking its satisfiability using an appropriate solver. The most common choice here is to use a SAT-solver and a propositional formula, but alternatively an SMT-solver (i.e. a solver capable to test satisfiability of a first-order logic over a built-in theory) and the corresponding first-order logic can be used. We apply the second approach.

In order to check whether a state satisfying φ is reachable, we first replace the model $M(\mathcal{N})$ by a model with a restricted set of timed labels, i.e., with $\mathbb{R}_+$ substituted by $[0, c_{\max} + 1]$, where by $c_{\max}$ we mean the greatest finite value of Eft and Lft of the transitions in $\mathcal{N}$. It can be shown that such a model is bisimilar with $M(\mathcal{N})$. Moreover, we restrict to integer time steps only, as they are sufficient to prove reachability [14, 15]. Then, we represent the states of the obtained model using integer variables, encode its transition relation in terms of the logic over integer arithmetics, and encode all the paths of a given length l starting at the initial state of $\mathcal{N}$ as a formula $Path(l)$. We encode also the fact that the property φ holds in the last state of a path as a formula $encode\_prop(\sigma_l, \varphi)$ (we omit technical details of the encodings in the current version of the work), and check satisfiability of the formula $\alpha := Path(l) \wedge encode\_prop(\sigma_l, \varphi)$. The above procedure is started from the length of a potential witness $l = 0$, and repeated iteratively up to (at most) $l = |M(\mathcal{N})|$. The process is stopped when the formula α is satisfiable, as this means that reachability of a state satisfying φ is proven, and therefore no further tests are necessary.

As usually in the case of bounded model checking methods, the above procedure can be inefficient if no state satisfying φ exists, since the depth of the part of the model considered (i.e., l) strongly influences the size of the encoding. Proving unreachability should therefore be done using a different algorithm we do not deal with in the current version of the paper.

3.1 Solving Other Reachability-Related Problems

The language of the propositions used enables to express not only simple properties of the form "a place p contains exactly x tokens", but also more involved ones, corresponding to various features of a marking of the net, e.g., "a place p contains more tokens than a place q", "the difference between the numbers of tokens in p and p' is greater than 5" etc. However, augmenting the net with an additional component enables testing also certain time-related properties.

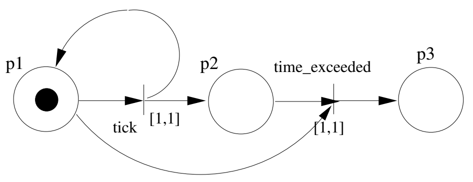

Fig. 1. An additional component

An example of such a component is shown in Fig. 1. If we set $cap(p2)$ and $w(p2, time\_exceeded)$ to a certain value $x \in \mathbb{N}_+$, the number of tokens in the place $p2$ corresponds to the number of time units (not greater than x) passed since the net started, and a token in $p3$ means that the time assumed has been exceeded. So, augmenting the net with this component enables to express in terms of reachability of a marking the properties like "a state satisfying φ is reachable in less / more than x time units". It is also possible to search for minimal / maximal time in which a state satisfying φ is reached, using the method similar to that described in [8].

4 Experimental Results

In this section, we present preliminary experimental results for the described method. The implemented tool consists of a module that generates an SMT input (the translation) for the SMT-solver, and a guiding module that performs the execution of the verification task which calls the generator module and the SMT-solver for the increasing lengths of the paths, until the SMT-solver terminates with a satisfiable formula.

We also provide a comparison with Sift which is a module of the Tina toolbox [16] providing *on-the-fly* verification capabilities. The results we present were generated using the switch `-D`, i.e., reachability was tested on the graph of essential states [14], similarly as in our method.

The translation module is implemented in C++ language, and the guiding module is implemented as a simple UNIX shell script. The SMT-solver used in the experiments was Z3 (version 4.3.2) [17]. The experiments were executed on a Linux 3.9.8 system, equipped with AMD X6 FX-6100 processor, and 8GB of memory. However, for all our experiments we assume the time limit of 2000 seconds, and the memory limit of 2000 MiB.

4.1 Fischer's Mutual Exclusion Benchmark

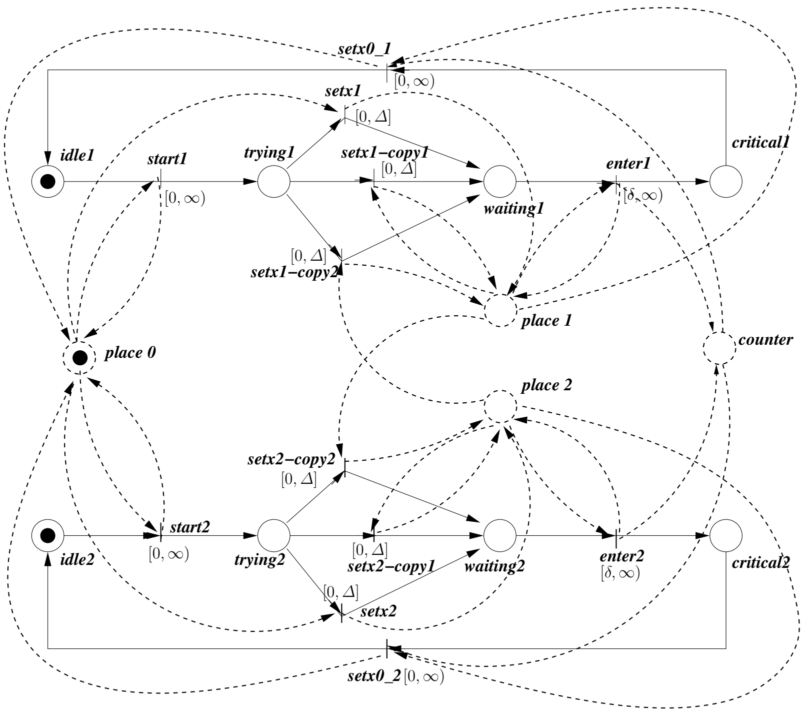

Fig. 2. A net for Fischer's mutual exclusion protocol for $n = 2$

In this benchmark we consider a net for the Fischer's mutual exclusion protocol [18] depicted in Figure 2. In our figures we assume that unless stated (or denoted in the figure) otherwise, the weight of all the arcs is 1, and the capacity restriction is also 1.

The net models n processes with critical sections. When the i-th process enters (leaves) its critical section $critical_i$ (for $i \in \{1, \cdots, n\}$), the number of tokens in the $counter$ place is incremented (decremented). We assume that $cap(counter) = n$. The mutual exclusion property of the protocol depends on the values of the time-delay constants δ and Δ, i.e., the property is preserved iff $\Delta < \delta$.

For this benchmark, we assume $\Delta = 2$ and $\delta = 1$, and we test the reachability of a marking m such that $m(counter) > 1$. The experimental results for this benchmark are presented in Table 1. The time is given in seconds (cpu time), and the memory in MiB. Where the assumed time or memory limit was exceeded, we denote this in the tables with the symbol $\star$.

4.2 Assembly Line Benchmark

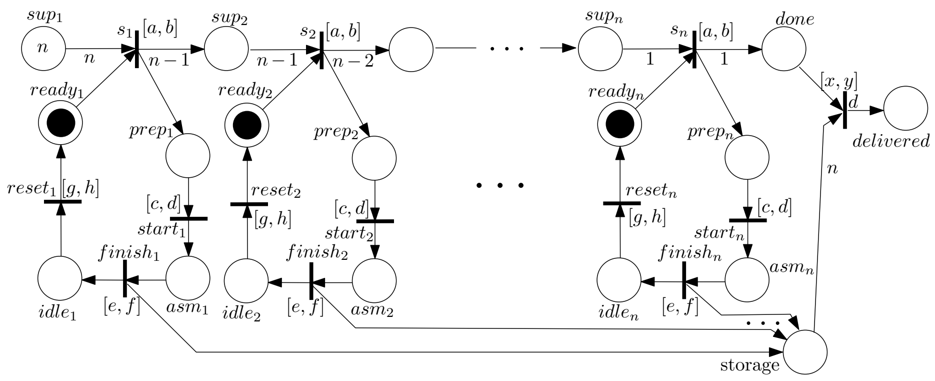

Fig. 3. Assembly line

A net for an abstract assembly line inspired by the generalised transfer chain model from [19] is presented in Figure 3. It consists of n assembly workers, and a supplier that provides resources needed in the assembly process. There are n resource packages which are represented by tokens in places sup_i for $i \in \{1, \cdots, n\}$. When an i-th worker receives the resource package via the s_i transition, it first prepares for the assembly process ($prep_i$), then performs the assembly itself (asm_i), and when finishing ($finish_i$) it also delivers the assembled product to the storage ($storage$). Then, it idles ($idle_i$), and becomes ready again ($ready_i$).

For all $p \in \{sup_1, \cdots, sup_n, storage, delivered\}$ we assume $cap(p) = n$. However, note that it could also be assumed that $cap(sup_i) = n - i + 1$ for $i \in \{1, \cdots, n\}$.

In our benchmark, we also assume the following time constraints in the system: $a = 0$, $b = 5$, $c = 0$, $d = 1$, $e = 0$, $f = 1$, $g = 0$, $h = 1$, $x = 0$, and $y = 100$. Then, we test the reachability of a marking m, where $m(done) > 0$ and $m(storage) > 0$. The length of the witness for this property is $n + 2$, where n is the number of assembly workers. The experimental results for this benchmark are presented in Table 2. Similarly as before, with $\star$ we mark the cases when the assumed time or memory limit was exceeded.

	SMT-BMC		TINA (sift)	
n	**time**	**memory**	**time**	**memory**
2	1.143	12.4	0.01	4.02
3	2.864	12.4	0.01	4.02
4	5.165	12.72	0.01	4.02
5	7.860	16.07	0.01	4.02
6	11.543	22.80	0.01	4.02
7	16.632	27.13	0.01	4.02
8	23.782	33.60	0.01	4.02
9	34.094	40.93	0.03	21.05
10	45.082	52.79	0.05	39.62
11	62.156	60.41	0.06	37.93
12	88.648	73.39	0.09	42.87
13	113.881	82.92	0.12	72.55
14	144.093	102.99	0.12	133.74
15	188.220	119.49	0.28	199.49
16	230.758	132.14	0.39	296.74
17	287.631	147.20	0.57	392.93
18	349.011	168.58	0.68	514.87
19	425.466	194.11	0.89	684.80
20	517.590	212.50	1.16	903.99
21	634.941	262.36	1.64	1202.24
22	775.141	282.44	1.91	1539.24
23	980.982	334.33	2.47	1877.18
24	1097.792	359.29	3.35	2387.43*
25	1311.071	385.24	4.29	2932.49*
26	1508.893	423.76	5.34	3715.05*
27	1721.292	452.26	5.45	3631.68*
28	1986.422	506.51	5.28	3606.80*
29	2242.422*	551.32		

Table 1. Results for Fischer's Protocol

	SMT-BMC		TINA (sift)	
n	**time**	**memory**	**time**	**memory**
2	1.073	15.70	0.01	4.02
3	2.387	27.52	0.01	4.02
4	4.574	43.74	0.01	4.02
5	8.071	60.08	0.01	4.02
6	13.315	85.56	0.01	4.02
7	20.373	106.97	0.02	4.02
8	30.039	142.03	0.03	28.24
9	41.742	172.50	0.05	38.12
10	54.235	202.65	0.13	95.74
11	71.999	235.27	0.25	214.80
12	94.619	295.05	0.56	519.68
13	124.712	336.16	1.28	1202.62
14	159.626	382.76	2.88	2664.55*
15	184.786	428.20	4.33	3552.93*
16	239.593	477.61	5.44	3891.19*
17	314.841	570.93	6.14	3891.19*
18	371.564	632.92		
19	479.219	686.16		
20	581.981	753.00		
21	717.623	818.28		
22	920.044	894.76		
23	1135.285	964.96		
24	1238.729	1115.49		
25	1674.850	1195.32		
26	1860.204	1280.82		
27	2219.660*	1374.26		

Table 2. Results for Assembly Line system

It can be seen from the above results that our implementation has much longer execution times than Tina. However, our method requires less memory than the corresponding Tina execution, and the differences are also significant.

One should also comment on the lack of any comparison with the implementation of the SAT-based reachability verification described in [8]. The paper mentioned provides experimental results for the Fischer's mutual exclusion protocol obtained on a computer equipped with Intel Pentium Dual CPU (2.00 GHz) and 2 GB of main memory, and confirms verifying the system consisting of 40 processes in 2999.5 seconds and using 1153.2 MB of memory. This seems to be a far better result. However, it should be noticed that the method of [8] was developed for distributed TPNs, of a semantics aimed at making verification more efficient than in the standard case (i.e., of the one in which clocks are assigned to the processes of the net instead of to the transitions which enables to reduce their number from $6n$ to $n+1$). The distributed nets are also 1-safe, which reduces further the complexity of their description.

5 Final Remarks

We have presented a preliminary version of a reachability verification method for (bounded) time Petri nets, based on bounded model checking and SMT. In our future work we are going to build upon this method an approach allowing to verify more involved properties. In the preliminary comparison with Tina our method performed efficiently in terms of the memory consumption when testing the reachability property, which is very promising for the mentioned extension of our work.

Acknowledgements. Artur Męski acknowledges the support of the EU, European Social Fund. Project PO KL "Information technologies: Research and their interdisciplinary applications" (UDA-POKL.04.01.01-00-051/10-00).

References

1. Merlin, P., Farber, D.J.: Recoverability of communication protocols – implication of a theoretical study. IEEE Trans. on Communications **24(9)** (1976) 1036–1043
2. Berthomieu, B., Diaz, M.: Modeling and verification of time dependent systems using time Petri nets. IEEE Trans. on Software Eng. **17(3)** (1991) 259–273
3. Berthomieu, B., Vernadat, F.: State class constructions for branching analysis of time Petri nets. In: Proc. of the 9th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'03). Volume 2619 of LNCS., Springer-Verlag (2003) 442–457
4. Boucheneb, H., Hadjidj, R.: CTL* model checking for time Petri nets. Theoretical Computer Science **353(1)** (2006) 208–227
5. Lime, D., Roux, O.H.: Model checking of time Petri nets using the state class timed automaton. Discrete Event Dynamic Systems **16(2)** (2006) 179–205
6. Męski, A., Penczek, W., Półrola, A., Woźna-Szcześniak, B., Zbrzezny, A.: 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), University of Hamburg (2011) 72–91

7. Penczek, W., Półrola, A.: Abstractions and partial order reductions for checking branching properties of time Petri nets. In: Proc. of the 22nd Int. Conf. on Applications and Theory of Petri Nets (ICATPN'01). Volume 2075 of LNCS., Springer-Verlag (2001) 323–342
8. Penczek, W., Półrola, A., Zbrzezny, A.: SAT-based (parametric) reachability for a class of distributed time Petri nets. In: Trans. on Petri Nets and Other Models of Concurrency IV. Volume 6550 of LNCS. Springer-Verlag (2010) 72–97
9. Virbitskaite, I.B., Pokozy, E.A.: A partial order method for the verification of time Petri nets. In: Fundamental of Computation Theory. Volume 1684 of LNCS. Springer-Verlag (1999) 547–558
10. Wan, M., Ciardo, G.: Symbolic reachability analysis of integer timed Petri nets. In: Proc. 35th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM). (2009) 595–608
11. Yoneda, T., Ryuba, H.: CTL model checking of time Petri nets using geometric regions. IEICE Trans. Inf. and Syst. **3** (1998) 1–10
12. Boyer, M., Diaz, M.: Multiple enabledness in Petri nets with time. In: Proc. of the 9th Int. Workshop on Petri Nets and Performance Models (PNPM'01). (2001) 219–228
13. Biere, A., Cimatti, A., Clarke, E., Zhu, Y.: Symbolic model checking without BDDs. In: Proc. of the 5th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'99). Volume 1579 of LNCS., Springer-Verlag (1999) 193–207
14. Popova-Zeugmann, L.: Time Petri nets state space reduction using dynamic programming. Journal of Control and Cybernetics **35(3)** (2006) 721–748
15. Janowska, A., Penczek, W., Półrola, A., Zbrzezny, A.: 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), Bialystok University of Technology (2011) 215–228
16. Berthomieu, B., Ribet, P.O., Vernadat, F.: The tool TINA - construction of abstract state spaces for Petri nets and time Petri nets. International Journal of Production Research **42(14)** (2004)
17. de Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: Proc. of the 14th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'08). Volume 4963 of LNCS., Springer-Verlag (2008) 337–340
18. Abadi, M., Lamport, L.: An old-fashioned recipe for real time. In: REX workshop on Real-Time: Theory in Practice. Volume 600 of LNCS., Springer-Verlag (1991) 1–27
19. Tsinarakis, G., Tsourveloudis, N., Valavanis, K.: Modular Petri net based modeling, analysis, synthesis and performance evaluation of random topology dedicated production systems. Journal of Intelligent Manufacturing **16**(1) (2005) 67–92