Scientific Issues
Jan Długosz University
in Częstochowa
Mathematics XX (2015)
83–92
DOI http://dx.doi.org/10.16926/m.2015.20.09

SMT-BASED REACHABILITY ANALYSIS FOR SIMPLY-TIMED SYSTEMS

AGNIESZKA M. ZBRZEZNY, ANDRZEJ ZBRZEZNY

Abstract

In the paper we present Satisfiability Modulo Theory based (SMT-based) reachability analysis algorithm for Simply-Timed Systems (i.e., Kripke structures where each transition holds a duration, which is an arbitrary natural number) generated by simply-timed automata. The algorithm is based on a SMT-based encoding for Simply-Timed Systems. We have tested the algorithm in question by using the generic simply timed pipeline paradigm model as the benchmark. The performance evaluation of the algorithm is given by means of the running time and the memory used.

1. Introduction

Model checking has been proposed independently by Clarke and Emerson [11], and by Quielle and Sifakis [22] as a method for automatic and algorithmic verification of finite state concurrent systems, and impressive strides have been made on this problem over the past thirty years [10, 12]. Model checking of real-time [2, 3, 14, 4] and multi-agent systems [15, 23] is a very active field for both theoretical research and practical applications.

The practical applicability of model checking in real-time, and multi-agent settings has required the development of sophisticated means of coping with what is known as the state explosion problem. It means that the

• *Agnieszka M. Zbrzezny* — Jan Długosz University in Częstochowa.

Partly supported by National Science Centre under the grant No. 2014/15/N/ST6/05079.

The study is co-funded by the European Union, European Social Fund. Project PO KL "Information technologies: Research and their interdisciplinary applications", Agreement UDA-POKL.04.01.01-00-051/10-00.

• *Andrzej Zbrzezny* — Jan Długosz University in Częstochowa.

number of model states grows exponentially in the size of the system representation. To avoid this problem a number of state reduction techniques and symbolic model checking approaches have been developed.

Verification of real-time systems (i.e., systems in which the total correctness of a computation depends not only on its logical correctness, but also on a certain subset of deadlines in which it is performed) is an actively developing field of research [3, 25, 9, 24, 16, 21]. Popular models of such systems include timed automata [1], time Petri nets [19], and simply-timed systems (STSs) [18], i.e., Kripke models where each transition holds a duration, which can be any integer value (including zero). For those models several approaches based on model checking have been proposed [3, 9, 24, 16, 21, 25].

Simply-timed systems are an extension of transition system where each transition is labelled by a nonnegative STSs allow for transitions that take a long time, allow transitions to have zero duration and the transitions with the zero duration allow for counting specific events only and thus omitting the irrelevant ones from the model checking point of view.

The SMT problem [7] is a generalisation of the SAT problem, where Boolean variables are replaced by predicates from various background theories, such as linear, real, and integer arithmetic. SMT generalises SAT by adding equality reasoning, arithmetic, fixed-size bit-vectors, arrays, quantifiers, and other useful first-order theories. Using SMT to express different problems has important advantages over SAT. If one uses SAT, then, for example, data must be encoded into a Boolean representation: a bit-vector must be represented as just its individual bits, for example. In contrast, an SMT encoding can represent the bit-vector directly, and may be able to reason more efficiently at the bit-vector level of abstraction, without resorting to bit-level reasoning (though this may sometimes be necessary).

In analogy with SAT, SMT procedures are usually referred to as SMT solvers. Present-day SMT research started in the late 1990s with different independent attempts [6, 17, 5, 20, 8] to build more scalable SMT solvers by exploiting advances in SAT technology. Over the last ten years a great deal of interest and research on the foundational and practical aspects of SMT was seen. SMT solvers have been developed in academia (Carnegie Mellon University, DISI-University of Trento, University of Lugano) and industry (Microsoft Research, Fondazione Bruno Kessler) with increasing scope and performance. An SMT solver is a tool for deciding the satisfiability (or dually the validity) of formulae in a number of theories. SMT solvers enable applications such as extended static checking, predicate abstraction, test case generation, and bounded model checking over infinite domains, to mention a few.

SAT can encode operations and relations on bounded integers using bitvector representation, with adders etc. represented as Boolean circuits and other finite data types and structures but cannot do not unbounded types (e.g. reals), or infinite structures (e.g. queues, lists) and even bounded arithmetic can be slow when it is large. There are fast decision procedures for these theories but their basic form works only on conjunctions. General propositional structure requires case analysis and that is what an SMT solver does.

The rest of the paper is organised as follows. We begin in 2 by introducing simply-timed systems. In Section 3 we present our simple translation to SMT. In Section 4 we present reachability algorithm. In Section 5 we discuss our experimental results. In the last section we conclude the paper.

2. Preliminaries

2.1. **Simply-timed systems.** Simply-timed systems [25] are an extension of transition system where each transition is labelled by a nonnegative integer. There are three main reasons why it is interesting to consider STSs instead of standard Kripke models. First, STSs allow for transitions that take a long time, e.g. 100 time units. Such transitions could be simulated in standard Kripke models by inserting 99 intermediate states. But this increases the size of the model, and so it makes the model checking process more difficult. Second, STSs allow transitions to have zero duration. This is very convenient in models where some steps are described indirectly, as a short succession of micro-steps. Third, the transitions with the zero duration allow for counting specific events only and thus omitting the irrelevant ones from the model checking point of view.

Definition 1. *A* simply-timed system *(STS for short), also called a* model, *is a tuple* $\mathcal{M} = (S, \mathcal{T}, Act, s^0, AP, L, d)$*, where* S *is a nonempty finite set of states,* $s^0 \in S$ *is the initial state,* Act *is a nonempty set of actions,* $\mathcal{T} \subseteq S \times Act \times S$ *is a transition relation,* AP *is a set of atomic propositions,* $L : S \to 2^{AP}$ *is a* labelling function *that assigns to each state a set of atomic propositions that are assumed to be true at that state, and* $d : Act \to \mathbb{N}$ *is a duration function.*

Each element $t = (s, \sigma, s') \in \mathcal{T}$ represents a transition from the state s to the state s', where σ is the action of the transition t.

From now on we assume that a STS has no terminal states, i.e. for every $s \in S$ there exist $s' \in S$ such that $s \longrightarrow s'$. Figure 1 displays a simple example of STS. A *path* in $\mathcal{M}$ is an infinite sequence $\pi = s_0 \xrightarrow{\sigma_1} s_1 \xrightarrow{\sigma_2} s_2 \xrightarrow{\sigma_3} \dots$ of transitions. For such a path, and for $m \in \mathbb{N}$, by $\pi(m)$ we denote the m-th state of π by s_m. For $j \leq m \in \mathbb{N}$, $\pi[j..m]$ denotes the

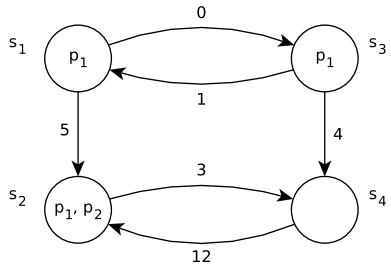

FIGURE 1. A simple STS.

finite sequence $s_j \xrightarrow{\sigma_{j+1}} s_{j+1} \xrightarrow{\sigma_{j+2}} \dots s_m$ with $m-j$ transitions and $m-j+1$ states. The (cumulative) *duration* $D\pi[j..m]$ of such a finite sequence is $d(\sigma_{j+1}) + \ldots + d(\sigma_m)$ (hence 0 when $j = m$). By $\Pi(s)$ we denote the set of all the paths starting at $s \in S$. Moreover we define a $k-path$ as a finite sequence $\pi = (s_0, ..., s_k)$ of states such that $s_j \longrightarrow s_{j+1}$ for each $0 \leq j < k$.

Let $\mathcal{M} = (S, \longrightarrow, Act, s^0, AP, L, d)$ be a model and s be a state. The state s is *reachable* in the model $\mathcal{M}$ if there exists a path $\pi \in \Pi(s^0)$ such that $\pi(k) = s$ for some $k \in \mathbb{N}$.

Since concurrent systems are modelled as a set of communicating processes, to verify them, it is reasonable to model communicating processes by a network of STSs that run in parallel, communicate with each other via shared actions and perform transitions with shared actions synchronously.

3. Translation to SMT

We have implemented a translation to SMT strictly following the translation to SAT (for details see for example [26]). In our translation to SMT states, durations and actions are represented by natural variables. Since $\mathcal{M}$ is a parallel composition of a finite number n of finite transition system, every state of $\mathcal{M}$ can be encoded as a natural number vector of the length n. Thus, each state of $\mathcal{M}$ can be represented by a valuation of a vector (called a *symbolic state*) of different individual variables called *individual state variables*. Moreover, every action of $\mathcal{M}$ can be represented by a valuation of an individual variable. Furthermore, k-paths can be represented as vectors of $k+1$ symbolic states.

3.1. **Bounded Model Checking.**

Now let us recall the BMC method of checking a reachability property represented by a given quantifier-free first-order formula $\varphi(\mathbf{w}, \mathbf{d})$, where $\mathbf{w}$ is a symbolic state and $\mathbf{d}$ is a symbolic

duration. Let $\mathcal{I}(\mathbf{w}_0, \mathbf{d}_0)$ be a quantifier-free first-order formula that represents the initial state and $\mathcal{T}((\mathbf{w}, \mathbf{d}), (\mathbf{w}', \mathbf{d}'))$ be a quantifier-free first order formula that encodes any possible transition. Now let us define the formula $[\mathcal{M}]_k$ that represents the unfolding to the depth k of the transition relation of $\mathcal{M}$:

$$[\mathcal{M}]_k := \bigwedge_{i=0}^{k-1} \mathcal{T}((\mathbf{w}_i, \mathbf{d}_i), (\mathbf{w}_{i+1}, \mathbf{d}_{i+1}))$$

In order to check the reachability property represented by the formula φ one has to check the satisfiability of the following conjunction:

$$[\mathcal{M}]_k^{\varphi} := \mathcal{I}(\mathbf{w}_0, \mathbf{d}_0) \wedge [\mathcal{M}]_k \wedge \varphi(\mathbf{w}_k, \mathbf{d}_k)$$

starting with $k = 0$. If for a given k the formula $[\mathcal{M}]_k^{\varphi}$ is not satisfiable, then k is increased and the resulting formula is to be checked by a SMT-solver again.

The method described relies on the following theorem.

Theorem 1. *Let $\mathcal{M}$ be a model and φ be a quantifier-free first-order formula. Then, there exists a reachable state in the set of states represented by the formula φ if, and only if, the formula $[\mathcal{M}]_k^{\varphi}$ is satisfiable for some $k \in \mathbb{N}$.*

Example 1. We use the generic simply-timed pipeline paradigm (GSPP) [25] as the benchmark. A network of STSs that models GSPP is shown in Fig. 2. We have $n + 2$ automata (n automata representing Nodes, one

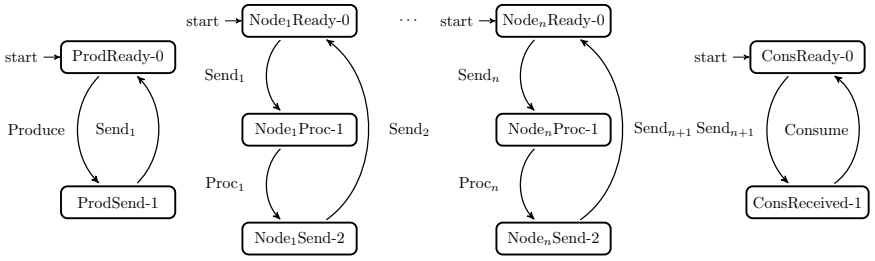

FIGURE 2. A network of STSs that models GSPP.

automaton for Producer, and one automaton for Consumer) that run in parallel and synchronise on actions $Send_i$ ($1 \leqslant i \leqslant n + 1$). Action $Send_i$ ($1 \leqslant i \leqslant n$) means that i-th Node has received data produced by Producer. Action $Send_{n+1}$ means that Consumer has received data produced by Producer. Action $Proc_i$ ($1 \leqslant i \leqslant n$) means that i-th Node processes data. Action *Produce* means that Producer generates data. Action *Consume* means that Consumer consumes data produced by Producer.

We show our translation to SMT for one Node and the following basic durations: $d(Produce) = 2$, $d(Send_i) = 2$, $d(Proc_i) = 4$, $d(Consume) = 2$, on the reachability property that the state $ConsRecieved$ is reachable in time less or equal to (2·coefficient)+(2·coefficient)·$(n+1)$+(4·coefficient)·n, where coefficient = 1.

Let π be a symbolic path of the length k. Stating that $\pi(i)$ is the initial state is encoded as the following formula $I(i)$:

$$w_{i,0} = 0 \wedge w_{i,1} = 0 \wedge w_{i,2} = 0 \wedge d_i = 0,$$

where $w_{i,j}$ denotes the local state of the j-th component at the depth i of the path π, and d_i denotes the symbolic duration at the depth i of the path.

The transition from the state $\pi(i)$ to the state $\pi(i+1)$ of the path π is encoded as the following formula $T(i)$, where $0 \leq i < k$:

$$\begin{aligned}
&(w_{i,0} = 0 \wedge w_{i+1,0} = 1 \wedge w_{i,1} = w_{i+1,1} \wedge w_{i,2} = w_{i+1,2} \wedge d_1 = 2) \vee \\
&(w_{i,0} = 1 \wedge w_{i+1,0} = 0 \wedge w_{i,1} = 0 \wedge w_{i+1,1} = 1 \wedge w_{i,2} = w_{i+1,2} \wedge d_1 = 2) \vee \\
&(w_{i,1} = 2 \wedge w_{i+1,1} = 0 \wedge w_{i,2} = 0 \wedge w_{i+1,2} = 1 \wedge w_{i,0} = w_{i+1,0} \wedge d_1 = 2) \vee \\
&(w_{i,1} = 1 \wedge w_{i+1,1} = 2 \wedge w_{i,0} = w_{i+1,0} \wedge w_{i,2} = w_{i+1,2} \wedge d_1 = 4) \vee \\
&(w_{i,2} = 1 \wedge w_{i+1,2} = 0 \wedge w_{i,0} = w_{i+1,0} \wedge w_{i,1} = w_{i+1,1} \wedge d_1 = 2)
\end{aligned}$$

Now, the symbolic path of the length k is encoded as the following formula:

$$I(0) \wedge \bigwedge_{i=0}^{k-1} T(i)$$

The reachability property for the path of the length k is encoded in the following way:

$$w_{k,2} = 1 \wedge \sum_{i=0}^{k} d_i \leq 10$$

4. Reachability algorithm

Given a simply-timed system and a property p (by a property we mean a set of states), the reachability problem consists in establishing whether a state satisfying p is reachable from the initial state of the system. To solve this problem we propose a method, which combines the well-known forward reachability analysis and Bounded Model Checking (BMC) method for STSs [25]. The forward reachability algorithm searches the state space by moving from one state to its successors in the Breadth First mode, whereas BMC performs a verification on a part of the model exploiting SMT solvers.

To check reachability of a state satisfying the property p, we unfold iteratively the transition relation of a simply-timed system in k steps. Then, the unfolding is encoded by a propositional formula, which characterises the

set of all the feasible paths through the transition relation with the length smaller than or equal k. Next, we translate the property p to a quantifier-free formula. Finally, the formula representing the „unfolded" transition relation and the property p is tested for satisfiability using Z3 [13]. The unfolding of the transition relation can be terminated when either a state satisfying the property has been found or all the states of the STS have been searched.

5. Experimantal Results

We have performed our experimental results on a computer equipped with I7-3770 processor, 32 GB of RAM, and the operating system Linux. We have tested the GSPP problem with the following basic durations: $d(Produce) = 2$, $d(Send_i) = 2$, $d(Proc_i) = 4$, $d(Consume) = 2$, and their multiplications by 1, 1000 and 1000000, on the reachability property stating that the state $ConsRecieved$ is reachable in time that is less or equal to $(2 \cdot \text{coefficient} + (2 \cdot \text{coefficient}) \cdot (n + 1) + (4 \cdot \text{coefficient}) \cdot n$, where coefficient $\in \mathbb{N}$.

In Figure 3 we present a comparison of total time usage and total memory usage of the reachability property for coefficient = 1.

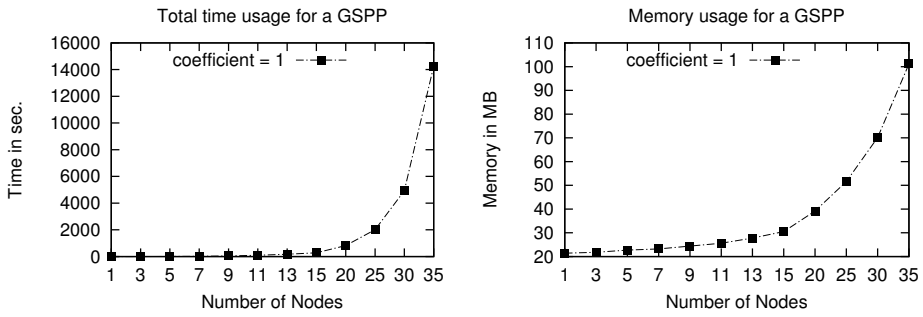

Figure 3. The total time usage and total memory usage of the reachability property for coefficient = 1.

In Figure 4 we present a comparison of total time usage and total memory usage of the reachability property for coefficient = 1000.

In Figure 5 we present a comparison of total time usage and total memory usage of the reachability property for coefficient = 1000000.

6. Conclusions

In this paper we have presented a SMT-based BMC reachability analysis algorithm for Simply-Timed Systems generated by simply-timed automata.

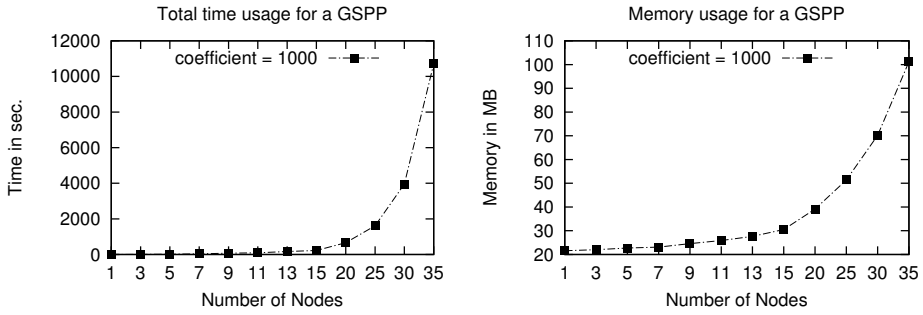

FIGURE 4. The total time usage and total memory usage of the reachability property for coefficient = 1000.

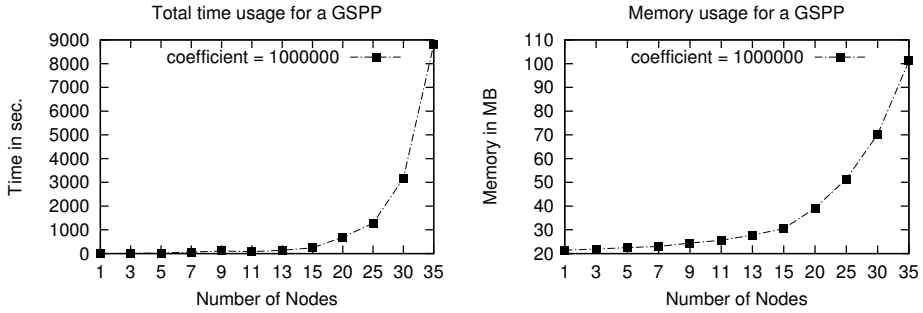

FIGURE 5. The total time usage and total memory usage of the reachability property for coefficient = 1000000.

The experimental results show performance of our SMT-based BMC reachability algorithm. More precisely our SMT-based BMC algorithm is insensitive to scaling up the durations. Time usage decreases with increase of coefficient. In our SMT-BMC reachability analysis algorithm we use the state of the art SMT-solver Z3 [13] (`http://z3.codeplex.com/`).

We should notice that the implementation of the SMT-based BMC reachability analysis algorithm for Simply-Timed Systems we used for performing the experiments is our first implementation that uses SMT solvers. Therefore, we hope to improve the implementation in the near future by taking many advantages of possibilities (of SMT-solvers) that we did not use so far.

References

[1] R. Alur. Timed Automata. In *Proceedings of the 11th International Conference on Computer Aided Verification (CAV'99)*, volume 1633 of *LNCS*, pages 8–22. Springer-Verlag, 1999.

[2] R. Alur, C. Courcoubetis, and D. Dill. Model checking for real-time systems. In *Proceedings of the 5th Symp. on Logic in Computer Science (LICS'90)*, pages 414–425. IEEE Computer Society, 1990.

[3] R. Alur, C. Courcoubetis, and D. Dill. Model checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.

[4] R. Alur and D. Dill. Automata-theoretic verification of real-time systems. In *Formal Methods for Real-Time Computing, Trends in Software Series*, pages 55–82. John Wiley & Sons, 1996.

[5] Alessandro Armando, Claudio Castellini, and Enrico Giunchiglia. SAT-based procedures for temporal reasoning. In *ECP*, pages 97–108, 1999.

[6] Alessandro Armando and Enrico Giunchiglia. Embedding complex decision procedures inside an interactive theorem prover. *Ann. Math. Artif. Intell.*, 8(3-4):475–502, 1993.

[7] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

[8] Randal E. Bryant, Steven M. German, and Miroslav N. Velev. Processor verification using efficient reductions of the logic of uninterpreted functions to propositional logic. *ACM Trans. Comput. Log.*, 2(1):93–134, 2001.

[9] S. Campos and E. Clarke. Analysis and verification of real-time systems using quantitative symbolic algorithms. *International Journal on Software Tools for Technology Transfer*, 2(3):260–269, 1999.

[10] Edmund M. Clarke. The birth of model checking. In *25 Years of Model Checking - History, Achievements, Perspectives*, volume 5000 of *Lecture Notes in Computer Science*, pages 1–26. Springer, 2008.

[11] Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. Automatic verification of finite state concurrent systems using temporal logic specifications: A practical approach. In *Conference Record of the Tenth Annual ACM Symposium on Principles of Programming Languages, Austin, Texas, USA, January 1983*, pages 117–126. ACM Press, 1983.

[12] Edmund M. Clarke and Jeannette M. Wing. Formal methods: State of the art and future directions. *ACM Comput. Surv.*, 28(4):626–643, 1996.

[13] Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *TACAS*, pages 337–340, 2008.

[14] E. A. Emerson and R. Trefler. Parametric quantitative temporal reasoning. In *Proceedings of the 14th Symp. on Logic in Computer Science (LICS'99)*, pages 336–343. IEEE Computer Society, July 1999.

[15] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

[16] C. A. Furia and P. Spoletini. Tomorrow and all our yesterdays: MTL satisfiability over the integers. In *Proceedings of the Theoretical Aspects of Computing (ICTAC'2008)*, volume 5160 of *LNCS*, pages 253–264. Springer-Verlag, 2008.

[17] Fausto Giunchiglia and Roberto Sebastiani. Building decision procedures for modal logics from propositional decision procedure - the case study of modal K. In *CADE*, pages 583–597, 1996.

[18] N. Markey and Ph. Schnoebelen. Symbolic model checking of simply-timed systems. In *Proceedings of the Joint Conferences Formal Modelling and Analysis of Timed Systems (FORMATS'04) and Formal Techniques in RealTime and Fault-Tolerant Systems (FTRTFT'04)*, volume 3253 of *LNCS*, pages 102–117. Springer, 2004.

[19] P. Merlin and D. J. Farber. Recoverability of communication protocols - implication of a theoretical study. *IEEE Transaction on Communications*, 24(9):1036–1043, 1976.

[20] Amir Pnueli, Yoav Rodeh, Ofer Strichman, and Michael Siegel. Deciding equality formulas by small domains instantiations. In *CAV*, pages 455–469, 1999.

[21] M. Pradella, A. Morzenti, and P. San Pietro. A metric encoding for bounded model checking. In *Proceedings of the 2nd World Congress on Formal Methods (FM 2009)*, volume 5850 of *LNCS*, pages 741–756. Springer-Verlag, 2009.

[22] J. P. Quielle and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *Proceedings of the 5th International Symp. on Programming*, volume 131 of *LNCS*, pages 337–351. Springer-Verlag, 1981.

[23] Wiebe van der Hoek and Michael Wooldridge. Model checking knowledge and time. In *Model Checking of Software, 9th International SPIN Workshop, Grenoble, France, April 11-13, 2002, Proceedings*, volume 2318 of *Lecture Notes in Computer Science*, pages 95–111. Springer, 2002.

[24] B. Woźna-Szcześniak, A. M. Zbrzezny, and A. Zbrzezny. The BMC method for the existential part of RTCTLK and interleaved interpreted systems. In *Proceedings of the 15th Portuguese Conference on Artificial Intelligence (EPIA'2011)*, volume 7026 of *LNAI*, pages 551–565. Springer-Verlag, 2011.

[25] Bożena Woźna-Szcześniak, Agnieszka Zbrzezny, and Andrzej Zbrzezny. SAT-based bounded model checking for RTECTL and simply-timed systems. In *EPEW*, pages 337–349, 2013.

[26] A. Zbrzezny. A new translation from ECTL* to SAT. *Fundamenta Informaticae*, 120(3-4):377–397, 2012.

Received: October 2015

Agnieszka M. Zbrzezny
Jan Długosz University in Częstochowa
Institute of Mathematics and Computer Science
al. Armii Krajowej 13/15, 42-200 Częstochowa, Poland
E-mail address: agnieszka.zbrzezny@ajd.czest.pl

Andrzej Zbrzezny
Jan Długosz University in Częstochowa
Institute of Mathematics and Computer Science
al. Armii Krajowej 13/15, 42-200 Częstochowa, Poland
E-mail address: a.zbrzezny@ajd.czest.pl