Checking RTECTL properties of STSs via SMT-based Bounded Model Checking

Agnieszka M. Zbrzezny, Andrzej Zbrzezny

MCS, Jan Długosz University in Częstochowa, Poland

***Abstract* —We present an SMT-based bounded model checking (BMC) method for Simply-Timed Systems (STSs) and for the existential fragment of the Real-time Computation Tree Logic. We implemented the SMT-based BMC algorithm and compared it with the SAT-based BMC method for the same systems and the same property language on several benchmarks for STSs. For the SAT-based BMC we used the PicoSAT solver and for the SMT-based BMC we used the Z3 solver. The experimental results show that the SMT-based BMC performs quite well and is, in fact, sometimes significantly faster than the tested SAT-based BMC.**

***Keywords* — RTECTL, SMT-based bounded model checking, STS**

I. Introduction

Verification of soft real-time systems is an actively developing field of research [2,9,10]. Popular models of such systems include, among others, timed automata [1], and simply-timed systems (STSs) [5], i.e., Kripke models where each transition holds a duration, which can be any integer value (including zero).

The fundamental thought behind bounded model checking (BMC) is, given a system, a property, and an integer bound $k \geq 0$, to define a formula such that the formula is satisfiable if and only if the system has a counterexample (of the length at most k) violating the property. The bound is incremented until a satisfiable formula is discovered or a completeness threshold is reached without discovering any satisfiable formulae. The SMT problem [3] 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.

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.

The original contribution of this paper consists in defining a SMT-based BMC method for the existential fragment of RTCTL (RTECTL) interpreted over simply-timed systems (STSs) generated by simply-timed automata with discrete data (STADDs). We implemented our SMT-based BMC algorithm and we compared it with the SAT-based BMC method for RTCTL and STSs. For a constructive evaluation of our SMT-based BMC method we have used two scalable benchmarks: a modified *bridge-crossing problem* [8] and a modified *generic pipeline paradigm* [7].

The rest of the paper is organised as follows. We begin in Section II by introducing simply-timed automata with discrete data, simply-timed systems, and we present the syntax and semantics of RTECTL over simply-timed systems. In Section III we present our SMT-based BMC method for RTECTL and simply-timed systems. In Section IV we discuss our experimental results. In the last section we conclude the paper.

II. Preliminaries

In this section we first define simply-timed automata with discrete data and simply-timed systems, and next we introduce syntax and semantics of RTECTL. The formalism of STADD was introduced in [12] and formalism of STS in [10].

A. Simply-timed automata with discrete data and simply-timed systems

Let $\mathbb{Z}$ be the set of integer numbers, $\bar{Z}$ a finite set of integer variables $\{z_1,\ldots,z_2\}$, $c \in Z$, $z \in \bar{Z}$, and $\oplus \in \{+,-,*,mod,\div\}$. The set $Expr(\bar{Z})$ of all the *arithmetic expressions* over $\bar{Z}$ is defined by the following grammar: $ae ::= c \mid z \mid ae \oplus ae \mid -ae \mid (ae)$. Next, for $ae \in Expr(\bar{Z})$ and $\sim \in \{=,\neq,<,\leq,\geq,>\}$, the set $BoE(\bar{Z})$ of all the Boolean expressions over $\bar{Z}$ is defined by the following grammar:

$$\beta ::= true \mid ae \sim ae \mid \beta \wedge \beta \mid \beta \vee \beta \mid \neg\beta \mid (\beta).$$

For $z \in \bar{Z}$, $ae \in Expr(\bar{Z})$, ϵ denoting the empty sequence, the set $SimAss(\bar{Z})$ of all the simultanoues assignments over $\bar{Z}$ is defined as $\alpha ::= \epsilon \mid z_{i_1},\ldots,z_{i_m} := ae_{i_1},\ldots,ae_{i_m}$, where $i_j \in \{1,\ldots,n\}$ and any $z \in \bar{Z}$ appears on the left-hand side of $:=$ at most once.

A *variables valuation* is a total mapping $v : \bar{Z} \to \mathbb{Z}$. We extend this mapping to expressions of $Expr(\bar{Z})$ in the usual way. Moreover, we assume that a domain of values for each variable is finite. Satisfiability of a Boolean expression $\beta \in BoE(\bar{Z})$ by a valuation v, denoted $v \models \beta$, is defined inductively as follows: $v \models true$, $v \models ae_1 \sim ae_2$ iff $v(ae_1) \sim v(ae_2)$, $v \models \beta_1 \wedge \beta_2$ iff $v \models \beta_1$ and $v \models \beta_2$, $v \models \beta_1 \vee \beta_2$ iff $v \models \beta_1$ or $v \models \beta_2$, $v \models \neg\beta$ iff $v \not\models \beta$, $v \models (\beta)$ iff $(v \models \beta)$. Given a variables valuation v and an instruction $\alpha \in Ins(\mathbb{Z})$, we denote by $v(\alpha)$ a valuation v' such that: if $\alpha = \epsilon$, then $v' = v$; if $\alpha = (z := expr)$, then for all $z' \in \mathbb{Z}$ it holds $v'(z') = v(expr)$ if $z' = z$, and $v'(z') = v(z')$ otherwise; if $\alpha = \alpha_1;\alpha_2$, then $v' = (v(\alpha_1))(\alpha_2)$.

DOI: 10.9781/ijimai.2015.354

Definition 1. *Let* PV *be a set of atomic propositions and* $\mathbb{N}=\{0,1,2,\ldots\}$. A simply-timed automaton with discrete data (STADD) *is a tuple* $A=(\Sigma,L,l^0,Z,E,d,V_A)$, *where* Σ *is a finite set of actions,* L *is a finite set of locations,* l^0 *is an initial location,* $\bar{Z}$ *is a finite set of integer variables,* $E\subseteq L\times\Sigma\times BoE(\bar{Z})\times SimAss(\bar{Z})\times L$ *is a transition relation,* $d:\Sigma\to\mathbb{N}$ *is a duration function, and* $V_A:L\to 2^{PV}$ *is a valuation function that assigns to each location a set of propositional variables that are assumed to be true at that location.*

The semantics of the STADD is defined by associating to it a simply-timed system as defined below.

Definition 2. *Let* PV *be a set of atomic propositions,* $v^0:\bar{Z}\to\mathbb{Z}$ *an initial variables valuation, and* $A=(\Sigma,L,l^0,\bar{Z},E,d,V_A)$ *a simply-timed automaton with discrete data.* A simply-timed system *(or* a model*) for* A *is a tuple* $M=(\Sigma,S,\iota,T,d,V)$, *where* Σ *is a finite set of actions of* A, $S=L\times\mathbb{Z}^{|\bar{Z}|}$ *is a set of states,* $d:\Sigma\to N$ *is the duration function of* A, $V:S\to 2^{PV}$ *is a valuation function defined as* $V((l,v))=V_A(l)$, *and* $\iota=(l^0,v^0)\in S$ *is the initial state,* $T\subseteq S\times\Sigma\times S$ *is the smallest simply-timed transition relation defined in the following way: for* $\sigma\in\Sigma$, $(l,v)\xrightarrow{\sigma}(l',v')$ *iff there exists a transition* $(l,\sigma,\beta,\alpha,l')\in E$ *such that* $v\models\beta$, $v'=v(\alpha)$. *We assume that the relation* T *is total, i.e., for any* $s\in S$ *there exists* $s'\in S$ *and an action* $\sigma\in\Sigma$ *such that* $(s,\sigma,s')\in T$ *(or* $s\xrightarrow{\sigma}s'$*).*

A *path* in M is an infinite sequence $\pi=s_0\xrightarrow{\sigma_1}s_1\xrightarrow{\sigma_2}s_2\xrightarrow{\sigma_2}\ldots$ of transitions. For such a path, and for $m\in\mathbb{N}$, by $\pi(m)$ we denote the m-th state s_m. For $j\le m\in\mathbb{N}$, $\pi[j..m]$ denotes the finite sequence $s_j\xrightarrow{\sigma_{j+1}}s_{j+1}\xrightarrow{\sigma_{j+2}}\ldots s_m$ 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 paths starting at $s\in S$.

B. RTECTL: an existential fragment of a soft real-time temporal logic.

In the syntax of RTECTL we assume the following: $p\in PV$ is an atomic proposition, and I is an interval in $\mathbb{N}=\{0,1,2,\ldots\}$ of the form: $[a,b)$ or $[a,\infty)$, for $a,b\in N$ and $a\neq b$. The RTECTL formulae are defined by the following grammar:

$$\varphi::=true\,|\,false\,|\,p\,|\,\neg p\,|\,\varphi\wedge\varphi\,|\,\varphi\vee\varphi\,|\,EX\varphi\,|\,E(\varphi U_I\varphi)\,|\,E(\varphi R_I\varphi).$$

Intuitively, we have an existential path quantifier E, and the symbols X, U_I, and R_I that are the temporal operators for "neXt time", "bounded until", and "bounded release", respectively. The formula $E(\alpha U_I\beta)$ means that it is possible to reach a state satisfying β via a finite path whose cumulative duration is in I, and always earlier α holds. The formula $E(\alpha R_I\beta)$ means that either it is possible to reach a state satisfying α and β via a finite path whose cumulative duration is in I, and always earlier β holds, or there is a path along which β holds at all states with cumulative duration being in I. The formulae for the "bounded eventually", and "bounded always" are defined as standard: $EF_I\varphi\stackrel{\text{def}}{=}E(true\,U_I\varphi)$, $EG_I\varphi\stackrel{\text{def}}{=}E(false\,R_I\varphi)$.

An RTECTL formula φ is true in the model M (in symbols $M\models\varphi$) iff $M,\iota\models\varphi$ (i.e., φ is true at the initial state of the model M). For every $s\in S$ the relation $\models$ is defined inductively as follows:

- $M,s\models\alpha\wedge\beta$ iff $M,s\models\alpha$ and $M,s\models\beta$
- $M,s\models\alpha\vee\beta$ iff $M,s\models\alpha$ or $M,s\models\beta$
- $M,s\models EX\alpha$ iff $(\exists\pi\in\Pi(s))(M,\pi(1)\models\alpha)$
- $M,s\models E(\alpha U_I\beta)$ iff $(\exists\pi\in\Pi(s))(\exists m\ge 0)$ $(D\pi[0..m]\in I$ and $M,\pi(m)\models\beta$ and $(\forall\, j<m)M,\pi(j)\models\alpha)$
- $M,s\models E(\alpha R_I\beta)$ iff $(\exists\pi\in\Pi(s))(\exists m\ge 0)$ $(D\pi[0..m]\in I$ and $M,\pi(m)\models\alpha$ and $(\forall\, j\le m)M,\pi(j)\models\beta)$ or $(\forall\, m\ge 0)D\pi[0..m]\in I$ implies $M,\pi(m)\models\beta$.

III. SMT-based bounded model checking

In this section we define the SMT-based BMC method for the existential fragment of RTCTL (RTECTL) [9]. Similarly to SAT-based BMC, the SMT-based BMC is based on the notion of the bounded semantics for RTECTL ([10]) in which one inductively defines for every $s\in S$ the relation $\models_k$. Let M be a model, $k\ge 0$ a bound, φ an RTECTL formula, and let $M,s\models_k\varphi$ denotes that φ is k-true at the state s of M. The formula φ is k-true in M (in symbols $M\models_k\varphi$) iff $M,\iota\models_k\varphi$ (i.e., φ is k-true at the initial state of the model M).

A. Bounded Semantics for RTECTL

Let M be a model, $k\ge 0$ a bound, φ an RTECTL formula, and $M,s\models_k\varphi$ denote that φ is k-true at the state s of M.

The formula φ is k-true in M (in symbols $M\models_k\varphi$) iff $M,\iota\models_k\varphi$ (i.e., φ is k-true at the initial state of the model M).

For every $s\in S$, the relation $\models_k$ (the bounded semantics) is defined inductively as follows:

- $M,s\models_k EX\alpha$ iff $k>0$ and $(\exists\pi\in\Pi_k(s))(M,\pi(1)\models_k\alpha)$
- $M,s\models_k E(\alpha U_I\beta)$ iff $(\exists\pi\in\Pi_k(s))(\exists 0\le m\le k)$ $(D\pi[0..m]\in I$ and $M,\pi(m)\models_k\beta$ and $(\forall\, 0\le j<m)M,\pi(j)\models_k\alpha)$
- $M,s\models_k E(\alpha R_I\beta)$ iff $(\exists\pi\in\Pi_k(s))(\exists 0\le m\le k)$ $(D\pi[0..m]\in I$ and $M,\pi(m)\models\alpha$ and $(\forall\, 0\le j\le m)M,\pi(j)\models_k\beta)$ or $(D\pi[0..k]\ge\text{right}(I)$ and $(\forall\, 0\le j\le k)$ $D\pi[0..j]\in I$ implies$)$ or $(\forall\, m\ge 0)D\pi[0..m]\in I$ implies $M,\pi(m)\models\beta$

Theorem 1 ([10]). *Let* M *be a model and an RTECTL formula. Then, the following equivalence holds:* $M \models \varphi$ *iff there exists* $k \geq 0$ *such that* $M \models_k \varphi$.

The bounded model checking problem asks whether there exists $k \in N$ such that $M \models_k \varphi$. The following theorem states that for a given model and an RTECTL formula there exists a bound k such that the model checking problem ($M \models_k \varphi$) can be reduced to the BMC problem ($M \models_k \varphi$). The theorem can be proven by induction on the length of the formula φ.

B. Translation to SMT

The translation to SMT is based on the bounded semantics. Let M M be a simply-timed model, φ an RTECTL formula, and k a bound. The presented SMT encoding of the BMC problem for RTECTL and for STS is based on the SAT encoding of the same problem [10,11], and it relies on defining the quantifier-free first-order formula

$$[M,\varphi]_k := [M^{\varphi,\iota}]_k \wedge [\varphi]_{M,k}$$

that is satisfiable if and only if $M \models_k \varphi$ holds.

The definition of the formula $[M^{\varphi,\iota}]_k$assumes that states of the model M are encoded in a symbolic way. Such a symbolic encoding is possible, since the set of states of M is finite. In particular, each state s can be represented by a vector w (called a *symbolic state*) of different individual variables ranging over the natural numbers (called *individual state variables*).

The formula $[M^{\varphi,\iota}]_k$ encodes a rooted tree of k -paths of the model M. The number of branches of the tree depends on the value of the auxiliary function $f_\iota : RTECTL \to \mathbb{N}$ defined in [10].

Given the above, the j -th symbolic k -path is defined as the following sequence $((d_{0,j}, w_{0,j}), \ldots, (d_{k,j}, w_{k,j}))$, where $w_{i,j}$ are symbolic states and $d_{i,j}$ are *symbolic durations*, for $0 \leq i \leq k$ and $0 \leq j \leq f_k(\varphi)$. The *symbolic duration* $d_{i,j}$ is a individual variable ranging over natural numbers.

Let w and w' (resp., d and d') be two different symbolic states (resp., durations). We assume definitions of the following auxiliary quantifier-free first-order formulae:

- $I_\iota(w)$ - encodes the initial state of the model M,
- $T((d,w),(d',w'))$ - encodes the transition relation of M,
- $p(w)$ - encodes the set of states of M in which $p \in PV$ holds,
- $B_k^I(\pi_n)$ encodes that the duration time represented by the sequence $d_{1,n}, \ldots, d_{k,n}$ of symbolic durations is less than right(I),
- $D_j^I(\pi_n)$ - encodes that the duration time represented by the sequence $d_{1,n}, \ldots, d_{j,n}$ of symbolic durations belongs to the interval I,
- $D_{k,l,m}^I(\pi_n)$ for $l \leq m$ - encodes that the duration time represented by the sequences $d_{1,n}, \ldots, d_{k,n}$ and $d_{l+1,n}, \ldots, d_{m,n}$ of symbolic durations belongs to the interval I.

The formula $[M^{\varphi,\iota}]_k$ encoding the unfolding of the transition relation of the model M $f_k(\varphi)$-times to the depth k is defined as follows:

$$[M^{\varphi,\iota}]_k = I_i(w_{0,0}) \wedge \bigwedge_{j=0}^{f_k(\varphi)-1} \bigwedge_{i=0}^{k-1} T((d_{i,j}, w_{i,j}), (d_{i+1,j}, w_{i+1,j}))$$

For every RTECTL formula φ the function f_k determines how many symbolic k -paths are needed for translating the formula φ. Given a formula φ and a set A of k -paths such that $|A| = f_k(\varphi)$, we divide the set A into subsets needed for translating the subformulae of φ. To accomplish this goal we need the auxiliary functions $g_s(A)$, $h_n^U(A,e)$ and $h_n^R(A,e)$ that were defined in [11].

Let φ be an RTECTL formula, M a model, and $k \in \mathbb{N}$ a bound. The quantifier-free first-order formula $[\varphi]_{M,k} := [\varphi]_k^{[0,0,F_k(\varphi)]}$, where $F_k(\varphi) = \{j \in N \mid 0 \leq j < f_k(\varphi)\}$, encodes the bounded semantics for RTECTL, and It is defined inductively as shown below. Namely, let $0 \leq n < f_k(\varphi)$, $m \leq k$, $n' = min(A)$, $h_k^U = h_k^U(A, f_k(\beta))$, $h_k^R = h_k^R(A, f_k(\alpha))$, then:

- $[EX\,\alpha]_k^{m,n,A} := w_{m,n} = w_{0,n'} \wedge [\alpha]_k^{[1,n',g_s(A)]}$ if $k > 0$; *false*, *otherwise*,
- $[E(\alpha U_I \beta)]_k^{m,n,A} := w_{m,n} = w_{0,n'} \wedge \bigvee_{i=0}^{k} \left([\beta]_k^{[i,n',h_k^U(k)]} \wedge D_i^I(\pi_{n'}) \wedge \bigwedge_{j=0}^{i-1} [\alpha]_k^{[j,n',h_k^R(j)]} \right)$,
- $[E(\alpha R_I \beta)]_k^{m,n,A} := w_{m,n} = w_{0,n'} \wedge \bigvee_{i=0}^{k} \left([\alpha]_k^{[i,n',h_k^R(k+1)]} \wedge D_i^I(\pi_{n'}) \wedge \bigwedge_{j=0}^{i} [\beta]_k^{[j,n',h_k^R(j)]} \right) \vee \left(\neg B_k^I(\pi_{n'}) \wedge \bigwedge_{j=0}^{k} \left(D_j^I(\pi_{n'}) \rightarrow [\beta]_k^{[j,n',h_k^R(j)]} \right) \right) \vee \left(B_k^I(\pi_{n'}) \wedge \bigwedge_{j=0}^{k} \left(D_j^I(\pi_{n'}) \rightarrow [\beta]_k^{[j,n',h_k^R(j)]} \right) \wedge \bigvee_{l=0}^{k-1} \left[w_{k,n'} = w_{l,n'} \right] \wedge \bigwedge_{l=0}^{k-1} \left(D_{k;l,j+1}^I(\pi_{n'}) \rightarrow [\beta]_k^{[j,n',h_k^R(j)]} \right) \right) \right)$.

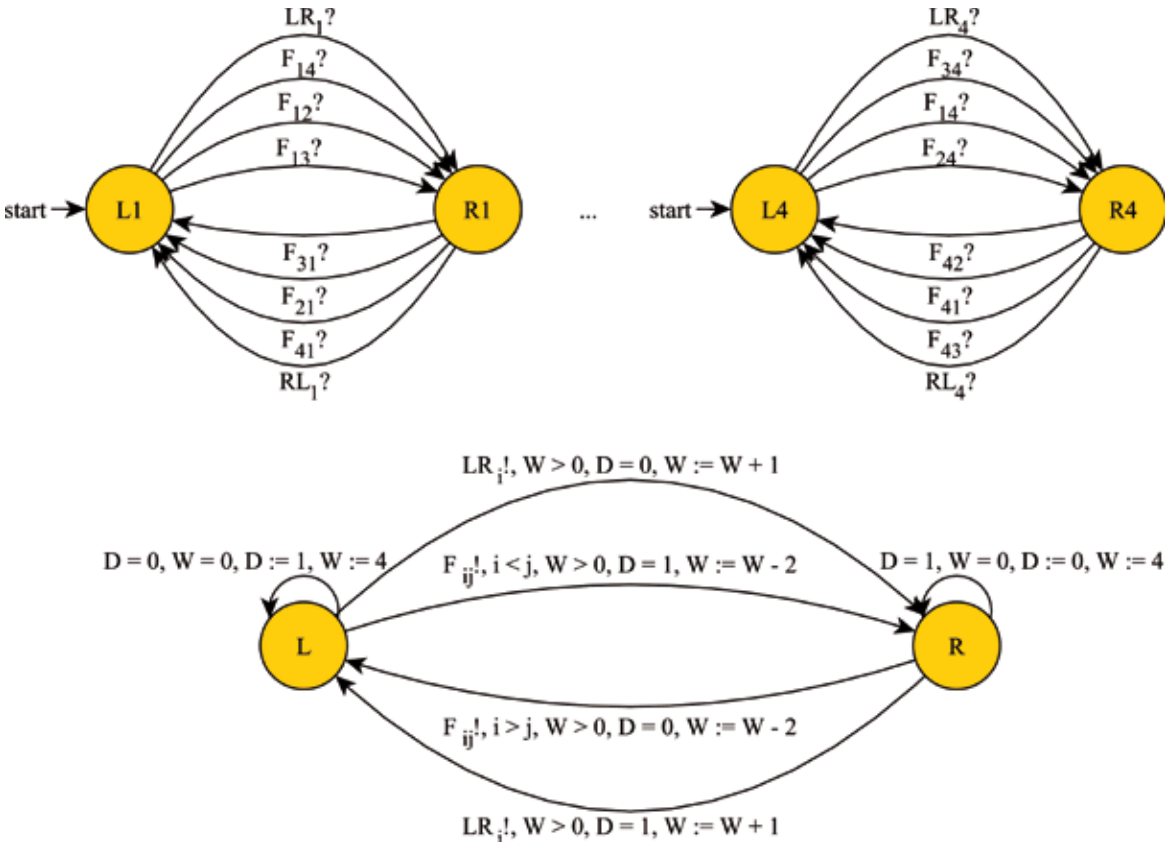


Figure 1: A network of STADD automata that models BCP for 4 persons. The variableD indicates the crossing direction: D = 1 (D = 0) means that all the persons crossthe bridge from its left side to its right side, (from its right side to its left side). Thevariable W denotes the number of persons waiting on the left (right) side of the bridge,if D = 1 (D = 0).

IV. Experimental results

Our SAT-based and SMT-based BMC algorithms were implemented as standalone programs written in the programming language C++. For the SAT-based BMC module we used the state of the art SAT-solver PicoSAT [4], and for our SMT-based BMC module we used the state of the art SMT-solver Z3 [6].

In this section we experimentally evaluate the performance of our SMT-based BMC encoding for RTECTL over the STS se- mantics. We compare our experimental results with the SAT-based BMC [10], the only existing method that is suitable with respect to the input formalism and checked properties.

We have conducted the experiments using two benchmarks: the generic simply-time pipeline paradigm (GSPP) STS model [10] and the bridge crossing problem (BCP) STS model [10]. We would like to point out that both benchmarks are very useful and scalable examples. Further, we specify each property for the considered benchmarks in the existential form, and for every specification given, there exists a witness.

We have computed our experimental results on a computer equipped with I7-3770 processor, 32 GB of RAM, and the operating system Arch Linux with the kernel 3.15.3. We set the CPU time limit to 3600 seconds. Moreover, in order to compare our SMT-based BMC with the SAT-based BMC, we have asked the authors of [10] to provide us the binary version of their implementation of the SAT-based BMC method. We have obtained the requested binaries. Furthermore, our SMT-based BMC algorithm is implemented as standalone program written in the programming language C++.

For the SAT-based BMC module we used the state of the art SAT-solver PicoSAT (http://fmv.jku.at/picosat/) [4], and for our SMT-based BMC module we used the state of the art SMT- solver Z3 [6] (http://z3.codeplex.com/).

A. The bridge-crossing problem

The bridge-crossing problem (BCP) [8] is a famous mathematical puzzle. To generate experimental results we have tested BCP system defined in [10]. We have five automata that run in parallel and synchronised on actions LR_i, RL_i, and F_{ij} for $i \neq j$ and $i, j \in \{1,...4\}$. The action LR_i (respectively, RL_i) means that the i-th person goes from the left side of the bridge to its right side (respectively, from the right side of the bridge to its left side) bringing back the lamp. The F_{ij} action with $i<j$ (respectively, F_{ij} with $i>j$) means that the persons i and j cross the bridge together from its left side to its right side (respectively, from its right side to its left side). Four automata (those with states named as Li and Ri, for $1 \leq i \leq 4$) represent persons, and one represents a lamp that keeps track of the position of the lamp, and ensures that at most two persons cross in one move. Let Min denote the minimum time required to cross the bridge, $N_p \geq 2$ be the number of persons, and $\text{right} = (2 \cdot N_p - 3) \cdot (t_1 + (N_p - 1) \cdot 3)$. We have tested BCP for $N_p \geq 4$ persons, $t_i - t_1 + (i-1) \cdot 3$ with $1 \leq i \leq N - p$ and $t_1 \geq 10$, on the following RTECTL formulae:

- $\varphi_{1\,BCP} = EF_{|Min, Min+1|}\left(\wedge_{i=1}^{N_p} Ri\right)$,
- $\varphi_{2\,BCP} = EG_{|0,\text{right}|}\left(\vee_{i=1}^{N_p} \neg Ri\right)$;

the formulae are true in the model for BCP.

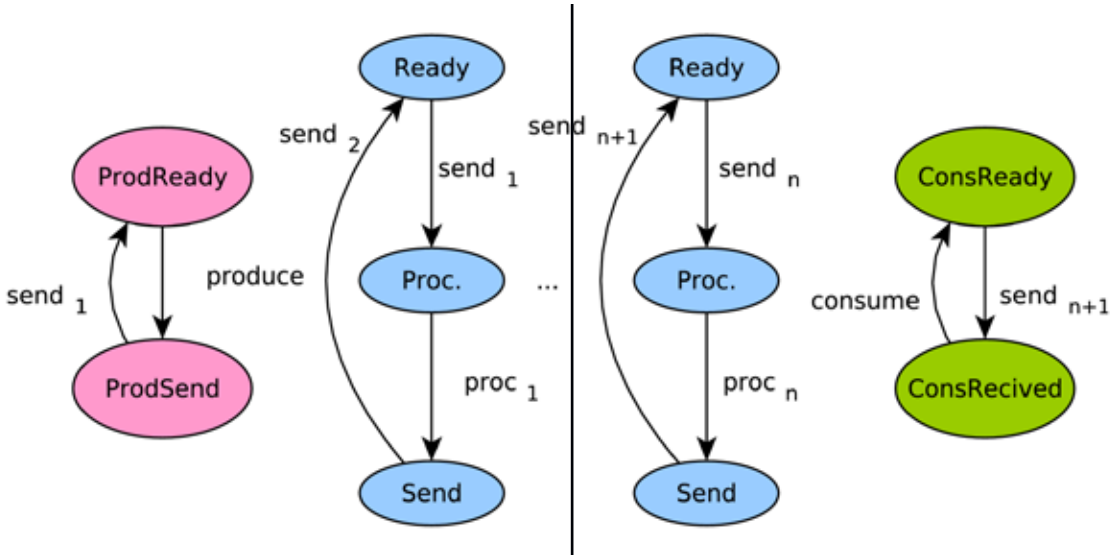


Figure 2: A network of STADD automata that models GSPP.

B. Generic Simply-timed Pipeline Paradigm

We adapted the benchmark scenario of *a generic pipeline paradigm* [7], and we called it the *generic simply-timed pipeline paradigm* (GSPP). The model of GSPP involves Producer producing data, Consumer receiving data, and a chain of n intermediate Nodes that transmit data produced by Producer to Consumer. Producer, Nodes, and Consumer have different producing, sending, processing, and consuming times. A STADD automata model of GSPP is shown in Fig 2. We have $n+2$ automata (n automata representing Nodes, one automaton for Producer, and one automaton for Consumer) that run in parallel and synchronise on actions $Send_i$ ($1 \le i \le n+1$). Action $Send_i$ $1 \le i \le n$ means that i -th Node has received data produced by Producer. Action $Proc_i$ $1 \le i \le n$ means that Consumer has received data produced by Producer.

Action $Proc_i$ $1 \le i \le n$ means that i -th Node processes data. Action $Produce$ means that Producer generates data. Action $Consume$ means that Consumer consumes data produced by Producer.

Let $1 \le i \le n$. 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 50, 100, 150, etc., on the following RTECTL formulae:

- $\varphi_{1GSPP} = EF_{[Min, Min+1]} ConsReceived$,
- $\varphi_{2GSPP} = EG_{[0,\infty)}(\neg ProdSend \vee EF_{[0, Min-d(Produce)+1]} ConsReceived)$,
- $\varphi_{3GSPP} = EG_{[0,\infty)}(\neg ProdSend \vee EG_{[0, Min-d(Produce)]} ConsReady)$,

where Min denotes the minimum time required to receive by Consumer the data produced by Producer.

Note that the φ_{2GSPP} and φ_{3GSPP} are properties, respectively, of the type the existential *bounded-response* and existential *bounded-invariance*. All the above formulae are true in the model for GSPP.

C. Performance evaluation

The evaluation of both the BMC algorithms is given by means of the running time and the memory used. In most cases, the experimental results show that the SMT-based BMC method is significantly faster than the SAT-based BMC method.

1) GSPP

From Fig. 3-8 and Tables 1-3 we can notice that for the GSPP system and all considered formulae the SMT-based BMC is faster than the SAT-based BMC, however, the SAT-based BMC consumes less memory. Moreover, the SMT-based method is able to verify more nodes for all the tested formulae. In particular, in the time limit set for the benchmarks, the SMT-based BMC is able to verify the formula φ_{1GSPP} for 54 nodes while the SAT-based BMC can handle 40 nodes, for the formula φ_{2GSPP} respectively 25 nodes and 21 nodes. For φ_{3GSPP} the SMT-based BMC is still more efficient - it is able to verify 20 nodes, whereas the SAT-based BMC verifies only 17 nodes for $t=1$ and 19 nodes for $t=1000000$.

TABLE 1: SMT-BMC: EXPERIMENTAL RESULTS FOR GSPP AND φ_{1GSPP} SCALING UP THE NUMBER OF NODES AND BASIC DURATION

n	Sec.	MB	Whitness length
1	0.1	12.5	5
5	0.1	13.5	13
10	0.7	16.6	23
15	3.0	21.8	33
20	8.4	29.7	43
25	21.4	40.3	53
30	56.3	50.7	63
35	139.0	70.6	73
40	323.0	86.3	83
45	577.9	109.8	93
50	1615.1	145.0	103
53	3220.1	153.5	109
54	3957.4	180.8	111

TABLE 2: SMT-BMC: EXPERIMENTAL RESULTS FOR GSPP AND φ_{2GSPP} SCALING UP THE NUMBER OF NODES AND BASIC DURATION

n	Sec.	MB	Whitness length
1	0.6	4.4	6
2	1.9	7.2	8
3	3.4	11.2	10
4	6.1	15.9	12
5	10.9	22.2	14
6	24.6	29.9	16
7	31.8	38.9	18
8	50.0	50.4	20
9	79.1	63.4	22
10	98.7	78.7	24
12	170.1	120.3	28
15	615.5	198.2	34
20	2304.1	423.5	44
22	2462.1	537.3	48
25	7203.7	777.5	54

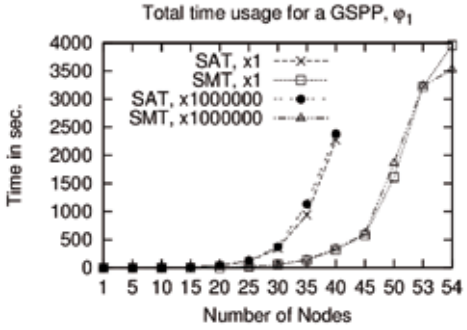
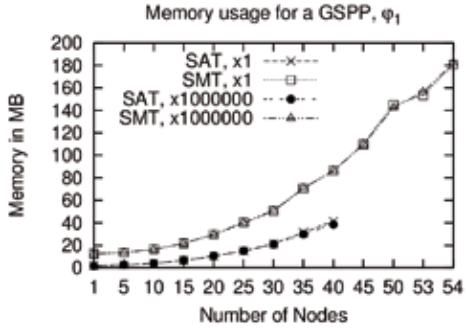


Figure 3: φ_{1GSPP} SAT/SMT-BMC: GSPP scaling up both the number of nodes and durations



4: φ_{1GSPP} SAT/SMT-BMC: GSPP scaling up both the number of nodes and durations

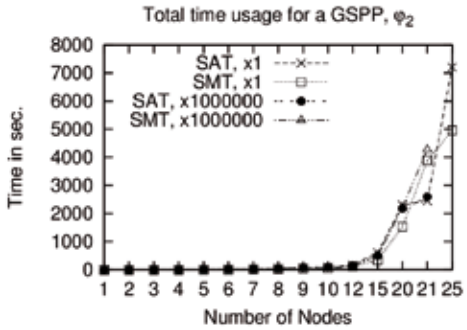


Figure 5:SAT/SMT-BMC: GSPP scaling up both the number of nodes and durations

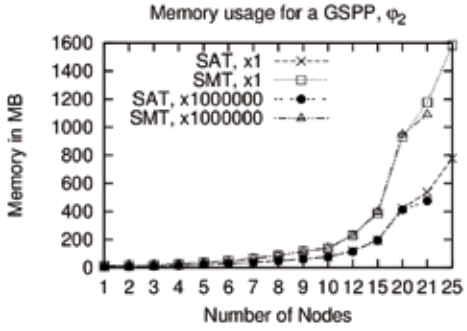


Figure 6: φ_{2GSPP} SAT/SMT-BMC: GSPP scaling up both the number of nodes and durations

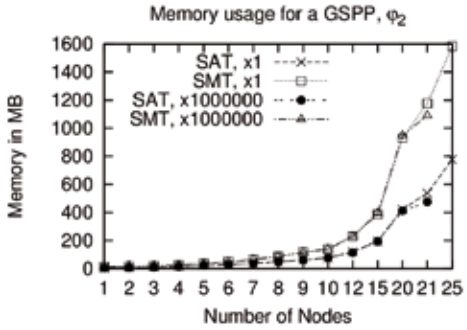


Figure 7: φ_{3GSPP} SAT/SMT-BMC: GSPP scaling up both the number of nodes and durations

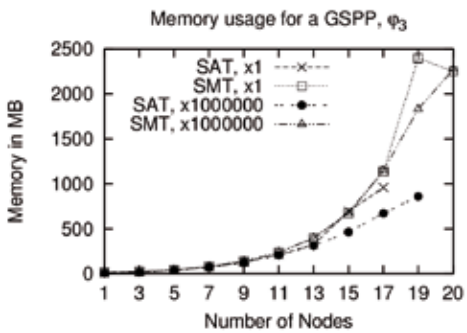


Figure 8: φ_{3GSPP} SAT/SMT-BMC: GSPP scaling up both the number of nodes and durations

TABLE 3: SMT-BMC: EXPERIMENTAL RESULTS FOR GSPP AND φ_{3GSPP} SCALING UP THE NUMBER OF NODES AND BASIC DURATION

n	Sec.	MB	Whitness length
1	0.2	14.4	5
2	0.7	17.7	7
3	1.4	23.6	9
4	2.7	31.8	11
5	4.8	43.6	13
6	8.4	62.6	15
7	16.9	81.6	17
8	25.8	103.3	19
9	37.7	138.1	21
10	78.0	200.0	23
11	90.8	232.3	25
12	143.0	301.1	27
13	203.3	393.9	29
14	407.9	503.8	31
15	402.0	670.1	33
16	675.1	1014.5	35
17	762.1	1136.4	37
18	1663.9	1901.3	39
19	2235.8	2397.2	41
20	3641.5	2252.4	43

2) BCP

As one can see from the line charts for the BCP system (Figures 9-12, Tables 4-5), in the case of this benchmark the SMT-based BMC and SAT-based BMC are complementary. In the case of the formula SMT-based BMC is able to verify system with 10 persons while the SAT-based BMC can handle 11 persons. For the SMT-based BMC is more efficient - it is able to verify 31 persons, whereas the SAT-based BMC verifies only 27 nodes for and 29 nodes for , but the SAT-based BMC consumes less memory.

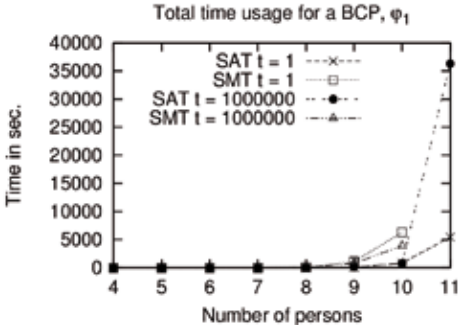


Figure 9: φ_{1BCP} SAT/SMT BMC: BCP scaling up both the number of persons and durations

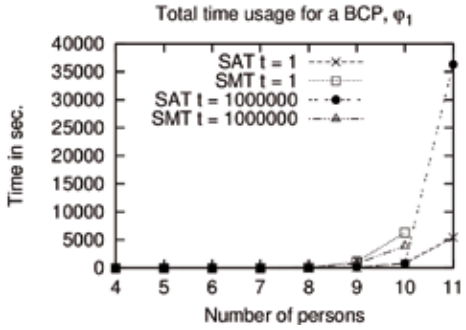


Figure 10: φ_{1BCP} SAT/SMT BMC: BCP scaling up both the number of persons and durations

TABLE 4: SMT-BMC: EXPERIMENTAL RESULTS FOR BCP AND φ_{1BCP} SCALING UP THE NUMBER OF NODES AND BASIC DURATION

n	Sec.	MB	Whitness length
4	0.0	13.6	6
5	0.1	14.4	8
6	0.8	16.4	10
7	4.8	21.3	12
8	67.5	44.3	14
9	1094.3	158.8	16
10	6298.2	163.6	18

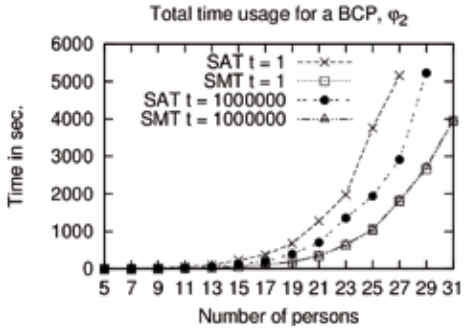


Figure 11: φ_{2BCP} SAT/SMT BMC: BCP scaling up both the number of persons and durations

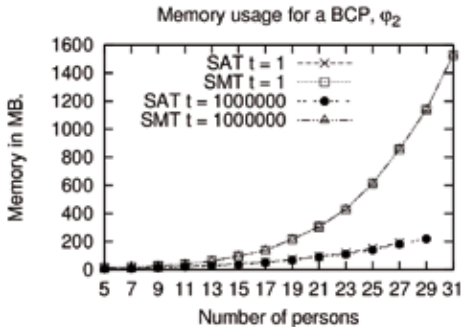


Figure 12: φ_{2BCP} SAT/SMT BMC: BCP scaling up both the number of persons and durations

TABLE 5: SMT-BMC: EXPERIMENTAL RESULTS FOR BCP AND φ_{2BCP} SCALING UP THE NUMBER OF NODES AND BASIC DURATION

n	Sec.	MB	Whitness length
4	0.1	13.7	6
5	0.2	14.6	8
6	0.5	16.4	10
7	1.2	19.3	12
8	2.1	22.4	14
9	3.7	27.3	16
10	5.9	33.1	18
11	9.9	39.7	20
12	14.9	49.0	22
13	25.9	61.7	24
14	34.1	73.8	26
15	54.4	96.5	28
16	71.7	114.3	30
17	105.3	138.4	32
18	141.7	172.2	34
19	182.1	216.8	36
20	265.5	251.8	38
21	340.1	305.9	40
22	436.0	367.3	42
23	630.3	428.9	44
24	816.6	521.3	46
25	1037.9	614.6	48
26	1309.7	734.7	50
27	1804.1	857.4	52
28	2385.1	986.7	54
29	2666.8	1139.2	56
30	3527.2	1326.0	58
31	3946.9	1528.0	60

V. Conclusion

We have proposed an SMT-based BMC verification method for model checking RTECTL properties interpreted over the simply-time systems that are generated for simply-timed automata with discrete data. We have provided a preliminary experimental results showing that our method is worth interest.

References

[1] R. Alur. Timed Automata. In *Proceedings of CAV'99*, volume 1633 of *LNCS*, pages 8-22. Springer-Verlag, 1999.

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

[3] Clark Barrett, Roberto Sebastiani, Sanjit Seshia, and Cesare Tinelli. Satisfiability modulo theories. In Armin Biere, Marijn J. H. Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 26, pages 825-885. IOS Press, 2009.

[4] A. Biere. PicoSAT essentials. *Journal on Satisfiability, Boolean Modeling and Computation (JSAT)*, 4:75-97, 2008.

[5] N. Markey and Ph. Schnoebelen. Symbolic model checking of simply-timed systems. In *Proceedings of FORMATS'04 and FTRTFT'04*, volume 3253 of *LNCS*, pages 102-117. Springer, 2004.

[6] L. De Moura and N. Bjørner. Z3: an efficient SMT solver. In *Proceedings of TACAS'2008*, volume 4963 of *LNCS*, pages 337-340. Springer-Verlag, 2008.

[7] D. Peled. All from one, one for all: On model checking using representatives. In *Proceedings of CAV'93*, volume 697 of *LNCS*, pages 409-423. Springer-Verlag, 1993.

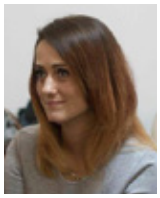
[8] Elizabeth Early Cook Saul X. Levmore. *Super Strategies for Puzzles and Games*. Doubleday, Garden City, N.Y., 1981.

[9] 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 EPIA'2011*, volume 7026 of *LNAI*, pages 551-565. Springer-Verlag, 2011.

[10] B. Woźna-Szcześniak, A. M. Zbrzezny, and A. Zbrzezny. SAT-based bounded model checking for RTECTL and simply-timed systems. In *Proceedings of EPEW 2013*, volume 8168 of *LNCS*, pages 337-349. Springer-Verlag, 2013.

[11] A. Zbrzezny. Improving the translation from ECTL to SAT. *Fundamenta Informaticae*, 85(1-4):513-531, 2008.

[12] A. Zbrzezny and A. Pólrola. SAT-based reachability checking for timed automata with discrete data. *Fundamenta Informaticae*, 79(3-4):579-593, 2007.



Agnieszka M. Zbrzezny received the M.S. in Computer Science from Jan Długosz University in Częstocjowa, now she is finishing the doctorate in the Institute of Computer Science Polish Acadamy of Sciences. Her research interest include model checking of multi-agent system and real-time systems.

Andrzej Zbrzezny received M.S. deegree in computer science from Jagielonian University in Cracow in 1981, PhD deegre in logic from Wrocław University in 1991, and habilitation in computer science from Polish Academy of Sciences in Warsaw in 2014. Now he is an associate professor at Jan Dlugosz University in Częstochowa. His research interest include model checking of concurrent systems, multi-agent system and real-time systems.