Bounded Model Checking for the Existential Fragment of TCTL and Diagonal Timed Automata

Bożena Woźna[1] and Andrzej Zbrzezny[2]

[1] Department of Computer Science,
University College London
Gower Street, London WC1E 6BT, UK
email: B.Wozna@cs.ucl.ac.uk

[2] Institute of Mathematics and Computer Science,
Jan Długosz University of Częstochowa
Al. Armii Krajowej 13/15, 42-200 Częstochowa, Poland
email: a.zbrzezny@ajd.czest.pl

Technical report: RN/05/19
August 2005

Abstract. Bounded Model Checking (BMC) is one of the well known SAT based symbolic model checking techniques. It consists in searching for a counterexample of a particular length, and generating a propositional formula that is satisfiable iff such a counterexample exists. The BMC method is feasible for the various classes of temporal logic; in particular it is feasible for TECTL (the existential fragment of Time Computation Tree Logic) and Diagonal-free Timed Automata. The main contribution of the paper is to show that the concept of Bounded Model Checking can be extended to deal with TECTL properties of Diagonal Timed Automata. We have implemented our new BMC algorithm, and we present preliminary experimental results, which demonstrate the efficiency of the method.

1 Introduction

Model checking is a verification technique that was originally developed for (untimed) temporal logics [8]. Its main idea is to represent a finite state system, often deriving from a hardware or software design, as a labelled transition system (model), represent a specification (property) by a modal formula, and check automatically whether the formula holds in the model.

Over the past few years, the interest in automated verification has been moved towards concurrent real-time systems, and now verification of such systems is an active area of research. Various classes of models for real time systems have been proposed in the literature, but Timed Automata [2] and Timed Petri Nets [20] are the best known ones. The properties to be verified are usually expressed either in a standard temporal logic like LTL [8] and CTL [14], or in their timed versions like MITL [3] and TCTL [1].

The practical applicability of the model checking method is strongly restricted by the so-called *state explosion problem*. Therefore, different reduction techniques were proposed to minimise models. The major methods include application of partial order

reductions [4, 9, 17, 21, 22, 30], symmetry reductions [15], abstraction techniques [10, 11], BDD-based symbolic storage methods [7, 18], and SAT-related algorithms [5, 19, 12, 28, 24–26, 32, 31].

Bounded model checking (BMC) is one of the SAT-based (satisfiability checking) methods, and it was introduced as a technique complementary to the BDD-based symbolic model checking for LTL [5]. The main idea of BMC is to search for an execution (or a set of executions) of the system of some length k that constitutes a counterexample for a tested property. If no counterexample of length k can be found, then k is increased by one until it reaches the size of the model. The efficiency of this method is based upon the observation that if a system is faulty, then often examining only a (small) fragment of its state space is sufficient for finding an error. Obviously, when testing large models and complex formulae the efficiency of the BMC method is dependent on the speed of the chosen SAT solver, with which the test is carried out. As SAT checkers have progressively become more effective, the efficiency of BMC has improved, an observation experimentally demonstrated in, among others, [5, 25, 26].

The main contribution of the paper consists in showing that the concept of Bounded Model Checking can be extended to deal with TECTL (the existential fragment of TCTL) properties of Diagonal Timed Automata [29]. In particular we show that the discretisation method [33] can be applied to model check an arbitrary TCTL formula over Diagonal Timed Automata. Moreover, we have implemented the new BMC algorithm, and we provide some preliminary experimental results that seem to be promising.

The rest of the paper is organised as follows. In Section 2 we present briefly the basic definitions and notations used through the paper. In Section 3 we present the bounded model checking for TECTL, and provide some preliminary experimental results for a modified Fischer mutual exclusion protocol. The last section contains a discussion of related work and final remarks.

2 Preliminaries

2.1 Diagonal timed automata

Let X be a finite set of variables, called *clocks*, and $\mathbb{N} = \{0, 1, \ldots\}$ a set of natural numbers. The set of *clock constraints* over X, denoted by $C(X)$, is defined by the following grammar:

$$\varphi \ ::= \ true \mid x \sim c \mid x - y \sim c \mid \varphi \wedge \varphi$$

where $x,\ y \in X, c \in \mathbb{N}$ and $\sim \in \{<, \leq, =, >, \geq\}$.

A *clock valuation* v of X is a total function from X into the set of nonnegative real numbers $\mathbb{R}$. $\mathbb{R}^X$ denotes the set of all clock valuations of X. For a clock constraint $\varphi \in C(X)$, $[\![\varphi]\!]$ denotes the set of all clock valuations of X that satisfy φ. The clock valuation that assigns the value 0 to all clocks is denoted by v^0. For $v \in \mathbb{R}^X$ and $\delta \in \mathbb{R}$, $v + \delta$ is the clock valuation of X that assigns the value $v(x) + \delta$ to each clock x. For $v \in \mathbb{R}^X$ and $Y \subseteq X$, $v[Y]$ denotes the clock valuation of X that assigns the value 0 to each clock in Y and leaves the values of the other clocks unchanged.

Definition 1 (Diagonal Timed Automaton). *Let $\mathcal{PV}$ be a set of propositional variables. A* diagonal timed automaton *$\mathcal{A}$ is a tuple $(\Sigma, L, l^0, \mathcal{V}, X, Inv, R)$, where Σ is a*

nonempty finite set of actions, L is a nonempty finite set of locations, $l^0 \in L$ is an initial location, $\mathcal{V} : L \longrightarrow 2^{\mathcal{PV}}$ is a function assigning to each location a set of atomic propositions true in that location, X is a finite set of clocks, $Inv : L \longrightarrow C(X)$ is a state invariant function, and $R \subseteq L \times \Sigma \times C(X) \times 2^X \times L$ is a transition relation.

Hereafter by *Timed Automata* we will mean Diagonal Timed Automata.

An element $(l, \sigma, \varphi, Y, l') \in R$ represents a transition from the location l to the location l' labelled with the action σ. The invariant condition allows the automaton to stay at the location l as long only as the constraint $Inv(l)$ is satisfied. The guard φ has to be satisfied to enable the transition. The transition resets all clocks in the set Y to the value 0.

2.2 Transition systems for timed automata

The *semantics of a timed automaton* $\mathcal{A}$ is defined by associating with it a labelled transition system $\mathcal{G}(\mathcal{A}) = (\Sigma \cup \mathbb{R}, Q, q^0, \widetilde{\mathcal{V}}, \longrightarrow)$, where $\Sigma \cup \mathbb{R}$ is the set of labels, $Q = L \times \mathbb{R}^X$ is the set of states, $q^0 = (l^0, v^0)$ is the initial state, $\widetilde{\mathcal{V}} : Q \longrightarrow 2^{AP}$ is a function such that $\widetilde{\mathcal{V}}((l, v)) = \mathcal{V}(l)$, and $\longrightarrow \subseteq Q \times (\Sigma \cup \mathbb{R}) \times Q$ is a transition relation of $\mathcal{G}(\mathcal{A})$ defined by:

- Time transitions: $(l, v) \xrightarrow{\delta} (l, v + \delta)$ iff $(\forall 0 \leq \delta' \leq \delta)\ v + \delta' \in [\![Inv(l)]\!]$
- Action transitions: $(l, v) \xrightarrow{\sigma} (l', v')$ iff $(\exists \varphi \in C(X))(\exists Y \subseteq X)$ such that $v' = v[Y]$, $(l, \sigma, \varphi, Y, l') \in R$, $v \in [\![\varphi]\!]$, and $v' \in [\![Inv(l')]\!]$.

Hereafter, the labelled transition system $\mathcal{G}(\mathcal{A})$ is called a *model*.

For $(l, v) \in Q$, let $(l, v) + \delta$ denote $(l, v + \delta)$. A q_0-*run* ρ of $\mathcal{A}$ is a sequence of states: $q_0 \xrightarrow{\delta_0} q_0 + \delta_0 \xrightarrow{\sigma_0} q_1 \xrightarrow{\delta_1} q_1 + \delta_1 \xrightarrow{\sigma_1} q_2 \xrightarrow{\delta_2} \ldots$, where $q_i \in Q$, $\sigma_i \in \Sigma$ and $\delta_i \in \mathbb{R} \setminus \{0\}$ for each $i \in \mathbb{N}$. A run ρ is said to be *progressive* iff $\Sigma_{i \in \mathbb{N}} \delta_i$ is unbounded. $\mathcal{A}$ is *progressive* iff all its runs are progressive. Hereafter, we consider only progressive timed automata. The reason of such restriction comes from the fact that the translation described in Section 2.7 works for progressive timed automata only. Note that progressiveness can be checked as in [29].

Lemma 1. *Let $\varphi \in C(X)$, $v \in \mathbb{R}^X$, and $\delta \in \mathbb{R}$. If $v \in [\![\varphi]\!]$ and $v + \delta \in [\![\varphi]\!]$, then for each $(0 \leq \delta' \leq \delta)$ $v + \delta' \in [\![\varphi]\!]$.*

Proof. Straightforward by induction on clock constraints.

As the above lemma shows, for the considered set of clock constraints $C(X)$, in the semantics of timed automata the condition of a time transition $(l, v) \xrightarrow{\delta} (l, v + \delta)$ can be replaced by the following: $v \in [\![\varphi]\!]$ and $v + \delta \in [\![\varphi]\!]$.

2.3 Time Computation Tree Logic

In this section, we formally present a syntax and semantics of Time Computation Tree Logic (TCTL) [1], which is a branching time temporal logic with bounded operators, and interpreted over a dense domain.

Syntax. Let $\mathcal{PV}$ be a set of propositional variables containing the symbol $\top$ (denoting the constant "true"), and I an interval in $\mathbb{R}$ with integer bounds of the form $[n, n']$, $[n, n')$, $(n, n']$, (n, n'), (n, ∞), and $[n, \infty)$, for $n, n' \in \mathbb{N}$. For $p \in \mathcal{PV}$, the *set of* TCTL *formulae* $\mathcal{F}(TCTL)$ is defined by the following grammar:

$$\varphi := p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathrm{E}(\varphi \mathrm{U}_I \varphi) \mid \mathrm{E}(\varphi \mathrm{R}_I \varphi) \mid \mathrm{A}(\varphi \mathrm{U}_I \varphi) \mid \mathrm{A}(\varphi \mathrm{R}_I \varphi)$$

The other basic operators are defined as usual: $\bot \stackrel{def}{=} \neg\top$, $\alpha \rightarrow \beta \stackrel{def}{=} \neg\alpha \vee \beta$, $\alpha \leftrightarrow \beta \stackrel{def}{=} (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$, $\mathrm{EG}_I\varphi \stackrel{def}{=} \mathrm{E}(\bot \mathrm{R}_I \varphi)$, $\mathrm{AG}_I\varphi \stackrel{def}{=} \mathrm{A}(\bot \mathrm{R}_I \varphi)$, $\mathrm{EF}_I\varphi \stackrel{def}{=} \mathrm{E}(\top \mathrm{U}_I \varphi)$, $\mathrm{AF}_I\varphi \stackrel{def}{=} \mathrm{A}(\top \mathrm{U}_I \varphi)$.

TECTL is a fragment of TCTL such that the temporal formulae are restricted to the Boolean combination of $\mathrm{E}(\alpha \mathrm{U}_I \beta)$ and $\mathrm{E}(\alpha \mathrm{R}_I \beta)$ only. Similarly, TACTL is a fragment of TCTL such that the temporal formulae are restricted to the Boolean combination of $\mathrm{A}(\alpha \mathrm{U}_I \beta)$ and $\mathrm{A}(\alpha \mathrm{R}_I \beta)$ only.

Semantics. Let $\mathcal{A} = (\Sigma, L, l^0, \mathcal{V}, X, Inv, R)$ be a timed automaton, and $\mathcal{G}(\mathcal{A}) = (\Sigma \cup \mathbb{R}, Q, q^0, \widetilde{\mathcal{V}}, \longrightarrow)$ a model for $\mathcal{A}$. Moreover, let $\rho = q_0 \xrightarrow{\delta_0} q_0 + \delta_0 \xrightarrow{a_0} q_1 \xrightarrow{\delta_1} q_1 + \delta_1 \xrightarrow{a_1} q_2 \xrightarrow{\delta_2} \ldots$ be a run of $\mathcal{A}$ such that $\delta_i \in \mathbb{R} \setminus \{0\}$ for $i \in \mathbb{N}$, and let $f_{\mathcal{A}}(q_0)$ denote the set of all such q_0-runs of $\mathcal{A}$. In order to give a semantics for TCTL, we introduce the notation of a *dense path* π_ρ *corresponding to run* ρ. *A dense path* π_ρ *corresponding to* ρ is a mapping from $\mathbb{R}$ to a set of states Q[1], such that $\pi_\rho(r) = q_i + \delta$ for $r = \Sigma_{j=0}^{i} \delta_j + \delta$ with $i \in \mathbb{N}$ and $0 \leq \delta < \delta_i$.

Definition 2 (Satisfaction). *Let* $\mathcal{G}(\mathcal{A}) = (\Sigma \cup \mathbb{R}, Q, q^0, \widetilde{\mathcal{V}}, \longrightarrow)$ *be a* model*, q a state, and α a* TCTL *formula. The satisfaction relation* $\models$*, which indicates truth of a formula in model* $\mathcal{G}(\mathcal{A})$ *at state q, is defined inductively as follows:*
$q \models p$ *iff* $p \in \widetilde{\mathcal{V}}(q)$, $q \models \varphi \vee \psi$ *iff* $q \models \varphi$ *or* $q \models \psi$,
$q \models \neg p$ *iff* $p \notin \widetilde{\mathcal{V}}(q)$, $q \models \varphi \wedge \psi$ *iff* $q \models \varphi$ *and* $q \models \psi$,
$q \models \mathrm{E}(\varphi \mathrm{U}_I \psi)$ *iff* $(\exists\, \rho \in f_{\mathcal{A}}(q))(\exists r \in I)[\pi_\rho(r) \models \psi$ *and* $(\forall r' < r)\ \pi_\rho(r') \models \varphi]$,
$q \models \mathrm{E}(\varphi \mathrm{R}_I \psi)$ *iff* $(\exists\, \rho \in f_{\mathcal{A}}(q))(\forall r \in I)[\pi_\rho(r) \models \psi$ *or* $(\exists r' < r)\ \pi_\rho(r') \models \varphi]$,
$q \models \mathrm{A}(\varphi \mathrm{U}_I \psi)$ *iff* $(\forall\, \rho \in f_{\mathcal{A}}(q))(\exists r \in I)[\pi_\rho(r) \models \psi$ *and* $(\forall r' < r)\ \pi_\rho(r') \models \varphi]$,
$q \models \mathrm{A}(\varphi \mathrm{R}_I \psi)$ *iff* $(\forall\, \rho \in f_{\mathcal{A}}(q))(\forall r \in I)[\pi_\rho(r) \models \psi$ *or* $(\exists r' < r)\ \pi_\rho(r') \models \varphi]$.

A TCTL formula φ is *satisfiable* iff there exists a model $\mathcal{G}(\mathcal{A}) = (\Sigma \cup \mathbb{R}, Q, q^0, \widetilde{\mathcal{V}}, \longrightarrow)$ and a state q of $\mathcal{G}(\mathcal{A})$, such that $\mathcal{G}(\mathcal{A}), q \models \varphi$. A TCTL formula φ is *valid in* $\mathcal{G}(\mathcal{A})$ (denoted by $\mathcal{G}(\mathcal{A}) \models \varphi$) iff $\mathcal{G}(\mathcal{A}), q^0 \models \varphi$, i.e. φ is true at the initial state of the model $\mathcal{G}(\mathcal{A})$; checking validity for given $\mathcal{G}(\mathcal{A})$ and φ is called the *model checking problem*.

2.4 Bounded transition systems for timed automata

Let c be a nonnegative integer constant. By $\mathcal{G}(\mathcal{A}, c) = (\Sigma \cup [0, c], Q, q^0, \widetilde{\mathcal{V}}, \longrightarrow)$ we denote the transition system for a timed automaton $\mathcal{A}$ which differs from the system

[1] This can be done because of the assumption that $\delta_i \in \mathbb{R} \setminus \{0\}$.

$\mathcal{G}(\mathcal{A})$ in the set of labels only. We shall call it *the bounded transition system*, or shortly *bounded model*.

It can be checked that the following lemma holds.

Lemma 2. *Let $\mathcal{A}$ be a timed automaton, φ a* TCTL *formula, and c_{max} the largest constant appearing in φ and in the invariants and the guards of $\mathcal{A}$. Moreover, let $\mathcal{G}(\mathcal{A})$ be a model for $\mathcal{A}$, $\mathcal{G}(\mathcal{A}, c_{max}+1)$ a bounded model for $\mathcal{A}$, and $q \in Q$ a state. Then, $\mathcal{G}(\mathcal{A}), q \models \varphi$ iff $\mathcal{G}(\mathcal{A}, c_{max}+1), q \models \varphi$*

In view of Lemma 2 our further considerations can be restricted to transition systems with bounded sets of time labels.

2.5 Weak region equivalence

For any $t \in \mathbb{R}$, $\langle t \rangle$ denotes the fractional part of t and $\lfloor t \rfloor$ denotes its integral part; note that $t = \lfloor t \rfloor + \langle t \rangle$. Then, the equivalence relation $\cong$, called the *weak region equivalence*, is defined over the set of all clock valuations for X. For two clock valuations u and v in $\mathbb{R}^X$, $u \cong v$ iff the following conditions hold:

E1. $\lfloor u(x) \rfloor = \lfloor v(x) \rfloor$, for all $x \in X$,
E2. $\langle u(x) \rangle = 0$ iff $\langle v(x) \rangle = 0$, for all $x \in X$,
E3. $\langle u(x) \rangle < \langle u(y) \rangle$ iff $\langle v(x) \rangle < \langle v(y) \rangle$, for all $x, y \in X$.

Notice, that the condition E3 implies the following condition:

E4. $\langle u(x) \rangle = \langle u(y) \rangle$ iff $\langle v(x) \rangle = \langle v(y) \rangle$, for all $x, y \in X$

Lemma 3 ([33]). *Let X be a set of clocks, and $u, v \in \mathbb{R}^X$ be clock valuations such that $u \cong v$. Then for any clock constraint $\varphi \in C(X)$, $u \in [\![\varphi]\!]$ iff $v \in [\![\varphi]\!]$.*

Lemma 4 ([33]). *Let X be a set of clocks, and $u, v \in \mathbb{R}^X$ be clock valuations such that for any clock constraint $\varphi \in C(X)$, $u \in [\![\varphi]\!]$ iff $v \in [\![\varphi]\!]$. Then $u \cong v$.*

Lemma 5. *Let X be a set of clocks, and $u, v \in \mathbb{R}^X$ be clock valuations such that $u \cong v$. Then for any $Y \subseteq X$, $u[Y] \cong v[Y]$.*

Proof. Straightforward from the definitions of the involved notions.

2.6 Discretisation

Let $\mathbb{Q}$ be a set of rational numbers, and $\mathcal{G}(\mathcal{A}, c)$ a bounded model for a timed automaton $\mathcal{A} = (\Sigma, L, l^0, \mathcal{V}, X, Inv, R)$ with n clocks. For every $m \in \mathbb{N}$, we define $D_m = \{d \in \mathbb{Q} \mid (\exists k \in \mathbb{N})\ d \cdot 2^m = k\}$ and $E_m = \{e \in \mathbb{Q} \mid (\exists k \in \mathbb{N})\ e \cdot 2^m = k \wedge e \leq c\}$. Then, we choose $D = \bigcup_{m=0}^{\infty} D_m$ as the set of the discretised clock values, and $E = \bigcup_{m=1}^{\infty} E_m$ as the set of labels.

Lemma 6 ([33]). *For every $v \in \mathbb{R}^X$ there exist $u \in D^X$ such that $u \cong v$.*

Lemma 7 ([33]). *Let $v \in \mathbb{R}^X$ be a clock valuation, $\delta \in [0, c]$, and $m \in \mathbb{N}$. Then for each $u \in D_m^X$ such that $u \cong v$ there exists $\delta' \in E_{m+1}$ such that $v + \delta \cong u + \delta'$. Moreover, $u + \delta' \in D_{m+1}^X$.*

We can now define a discretised bounded model for $\mathcal{G}(\mathcal{A}, c)$. Namely, a *discretised bounded model* for $\mathcal{G}(\mathcal{A}, c)$ is the tuple $\mathcal{D}(\mathcal{A}, c) = (\Sigma \cup E, L \times D^X, (l^0, v^0), \widetilde{\mathcal{V}}, \longrightarrow)$, where the transition relation $\longrightarrow$ is defined as follows:

- Time transitions: $(l, v) \xrightarrow{\delta} (l, v + \delta)$ iff $v \in [\![Inv(l)]\!]$ and $v + \delta \in [\![Inv(l)]\!]$.
- Action transitions: $(l, v) \xrightarrow{\sigma} (l', v')$ iff $(l, v) \xrightarrow{\sigma} (l', v')$ in $\mathcal{G}(\mathcal{A}, c)$

It is easy to check that the following lemma holds:

Lemma 8. *Let $\mathcal{A}$ be a timed automaton, φ a* TCTL *formula, and c_{max} the largest constant appearing in φ and in the invariants and the guards of $\mathcal{A}$. Moreover, let $\mathcal{G}(\mathcal{A}, c_{max} + 1)$ be a bounded model for $\mathcal{A}$, $\mathcal{D}(\mathcal{A}, c_{max} + 1)$ its discretised version, and $q \in Q$ a state. Then, $\mathcal{G}(\mathcal{A}, c_{max} + 1), q \models \varphi$ iff $\mathcal{D}(\mathcal{A}, c_{max} + 1), q \models \varphi$.*

2.7 Translation from TCTL to CTL$_y$

It is well known that the model checking problem for TCTL can be translated into the model checking problem for a fair version of CTL [1]; however, since we have assumed that we deal with progressive timed automata only, we can define that translation to the slightly different logic CTL$_y$, as presented below. The main idea of that translation consists in encoding all the time intervals appearing in the TCTL formula under consideration by propositional variables.

Formally, let $\mathcal{A} = (\Sigma, L, l^0, \mathcal{V}, X, Inv, R)$ be a timed automaton, and φ a TCTL formula. First, a new timed automaton $\mathcal{A}_\varphi = (\Sigma', L, l^0, \mathcal{V}', X', Inv, R')$ is constructed by extending $\mathcal{A}$ with: (1) a new clock y that corresponds to all the time intervals $\{I_1, \ldots, I_r\}$ appearing in φ, i.e. $X' = X \cup \{y\}$[2]; (2) an action σ_y, i.e. $\Sigma' = \Sigma \cup \{\sigma_y\}$; and (3) transitions that are used to reset the new clock y. Namely, $R' = R \cup \{(l, \sigma_y, true, \{y\}, l) \mid l \in L\}$. These transitions are used to start the runs over which subformulae of φ are checked. Then, the set of propositional variables $\mathcal{PV}$ is extended to the set $\mathcal{PV}'$ that is defined as follows: $\mathcal{PV}' = \mathcal{PV} \cup \mathcal{PV}_\varphi \cup \{p_b\}$, where p_b is a proposition variable representing the fact that a state is boundary, i.e. the value of at least one clock is equal to zero, and $\mathcal{PV}_\varphi = \{p_{y \in I_i} \mid I_i \text{ is a time interval in } \varphi\}$. Next, the discretised bounded model $\mathcal{D}(\mathcal{A}_\varphi, c_{max} + 1) = (\Sigma \cup E, L \times D^X, (l^0, v^0), \widetilde{\mathcal{V}}, \longrightarrow)$ for $\mathcal{A}_\varphi$ and c_{max} (i.e. the largest constant appearing in φ and in the invariants and the guards of $\mathcal{A}$) is constructed, where $\widetilde{\mathcal{V}} : L \times D^X \to 2^{\mathcal{PV}'}$, and $\mathcal{D}(\mathcal{A}_\varphi, c_{max} + 1), (l, v) \models p_{y \in I_i}$ iff $v(y) \in I_i$, and $\mathcal{D}(\mathcal{A}_\varphi, c_{max} + 1), (l, v) \models p_b$ iff there exists $x \in \mathrm{X}'$ such that $v(x) = 0$. Finally, the TCTL formula φ is translated into a CTL$_y$ formula $\psi = \mathrm{cr}(\varphi)$ in such a way that the model checking of φ over the discretised bounded model for $\mathcal{A}$ can be reduced to the model checking of ψ over the discretised bounded model for $\mathcal{A}_\varphi$.

In order to translate a TCTL formula φ into the corresponding CTL formula ψ we need to modify the CTL language into CTL$_y$ by reinterpreting the next-time operator, denoted now by X$_y$. This language is interpreted over discretised bounded model for $\mathcal{A}_\varphi$. The modality X$_y$ is interpreted only over the new transitions that reset the new clock y, whereas the other operators are interpreted over all other old transitions. Formally, for $p \in \mathcal{PV}'$, the set of CTL$_y$ formulae $\mathcal{F}(CTL_y)$ is defined by the grammar:

[2] One clock is sufficient to perform the bounded model checking algorithm that is proposed in the next section; however, other model checking methods can require one clock per one time interval appearing in the TCTL formula under consideration.

$$\alpha := p \mid \neg p \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \mathrm{X}_y\alpha \mid \mathrm{E}(\alpha \mathrm{U} \alpha) \mid \mathrm{E}(\alpha \mathrm{R} \alpha) \mid \mathrm{A}(\alpha \mathrm{U} \alpha) \mid \mathrm{A}(\alpha \mathrm{R} \alpha)$$

The satisfaction relation $\models$ for CTL_y formulae is defined as the corresponding satisfaction relation for CTL formulae [8]. It only differs in the operator X_y, which is defined as follows: $\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1), (l, v) \models \mathrm{X}_y\alpha$ iff $\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1), (l, v[\{y\}]) \models \alpha$.
The TCTL formula φ is translated inductively into the CTL_y formula $\mathrm{cr}(\varphi)$ as follows:

- $\mathrm{cr}(p) = p$ for $p \in \mathcal{PV}'$,
- $cr(\neg p) = \neg cr(p)$ for $p \in \mathcal{PV}'$,
- $\mathrm{cr}(\alpha \vee \beta) = \mathrm{cr}(\alpha) \vee \mathrm{cr}(\beta)$,
- $\mathrm{cr}(\alpha \wedge \beta) = \mathrm{cr}(\alpha) \wedge \mathrm{cr}(\beta)$,
- $\mathrm{cr}(\mathrm{E}(\alpha \mathrm{U}_{I_i} \beta)) = \mathrm{X}_y(\mathrm{E}(\mathrm{cr}(\alpha) \mathrm{U} (\mathrm{cr}(\beta) \wedge p_{y \in I_i} \wedge (p_b \vee \mathrm{cr}(\alpha)))))$,
- $\mathrm{cr}(\mathrm{E}(\alpha \mathrm{R}_{I_i} \beta)) = \mathrm{X}_y(\mathrm{E}(\mathrm{cr}(\alpha) \mathrm{R} (\neg p_{y \in I_i} \vee (\mathrm{cr}(\beta) \wedge (p_b \vee \mathrm{cr}(\alpha))))))$,
- $\mathrm{cr}(\mathrm{A}(\alpha \mathrm{U}_{I_i} \beta)) = \mathrm{X}_y(\mathrm{A}(\mathrm{cr}(\alpha) \mathrm{U} (\mathrm{cr}(\beta) \wedge p_{y \in I_i} \wedge (p_b \vee \mathrm{cr}(\alpha)))))$,
- $\mathrm{cr}(\mathrm{A}(\alpha \mathrm{R}_{I_i} \beta)) = \mathrm{X}_y(\mathrm{A}(\mathrm{cr}(\alpha) \mathrm{R} (\neg p_{y \in I_i} \vee (\mathrm{cr}(\beta) \wedge (p_b \vee \mathrm{cr}(\alpha))))))$.

The following lemma shows that validity of the TCTL formula φ over the model for $\mathcal{A}$ is equivalent to the validity of the corresponding CTL_y formula $\mathrm{cr}(\varphi)$ over the discretised bounded model for $\mathcal{A}_\varphi$ with the extended valuation function.

Lemma 9. *$\mathcal{G}(\mathcal{A}) \models \varphi$ iff $\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1) \models cr(\varphi)$, for each* TCTL *formula φ.*

Proof. The proof follows directly from Lemma on Correctness of the labelling algorithm of [1], Theorem 3.1 of [33], and Lemma 2.

3 Bounded Model Checking for TECTL

Let $\mathcal{A}$ be a time automaton, φ a TECTL formula, and c_{max} the largest constant appearing in φ, in the invariants and in the guards. In order to perform bounded model checking for TECTL we proceed by extending the technique employed for TCTL and diagonal-free automata [26, 23]. Namely, we first translate the model checking problem from TECTL into that problem for ECTL_y as in Section 2.7, and then we define BMC for ECTL_y.

3.1 BMC for ECTL_y.

Consider an ECTL_y formula $\psi = \mathrm{cr}(\varphi)$, where φ is a TECTL formula, a discretised bounded model $\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1) = (\Sigma \cup E, L \times D^X, (l^0, v^0), \widetilde{\mathcal{V}}, \longrightarrow)$, and a bound $k \in \mathbb{N}_+$. The main idea of BMC for ECTL_y consists in translating the model checking problem of an ECTL_y formula into the problem of satisfiability of a propositional formula $[\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1), \psi]_k = [\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1)^{\psi,(l^0,v^0)}]_k \wedge [\psi]_k^{0,0}$. The translation is based on $k-$bounded semantics for ECTL_y, which is defined as follows.

Let us denote by $\rightarrow_{\mathcal{A}}$ the part of $\longrightarrow$, where transitions are labelled with elements of $\Sigma \cup E$, and by $\rightarrow_y$ the transitions that reset the clock y. Then, a *path* π in $\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1)$ is a sequence $(s_0, s_1, \ldots)$ of states such that $s_i \rightarrow_{\mathcal{A}} s_{i+1}$ for each $i \in \mathbb{N}$. A path of length k is called *$k-$path*, and the set of all the k-paths starting at s in $\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1)$ is denoted by $\Pi_k(s)$. Furthermore, let α, β be ECTL_y subformulae of ψ, $k \in \mathbb{N}_+$ be a bound. Then, $(\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1), k)$, $s \models \alpha$ denotes that α is true at the state s of $\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1)$ with the bound k. $(\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1), k)$ is omitted if it is clear from the context. The satisfaction relation $\models$ is defined inductively as follows:

$s \models p$ iff $p \in \widetilde{\mathcal{V}}(s)$, $s \models \alpha \vee \beta$ iff $s \models \alpha$ or $s \models \beta$,
$s \models \neg p$ iff $p \notin \widetilde{\mathcal{V}}(s)$, $s \models \alpha \wedge \beta$ iff $s \models \alpha$ and $s \models \beta$,
$s \models \mathrm{X}_y\alpha$ iff $\exists s' \in L \times D^X$ $(s \rightarrow_y s'$ and $s' \models \alpha)$,
$s \models \mathrm{E}(\alpha \mathrm{U} \beta)$ iff $(\exists \pi \in \Pi_k(s))(\exists 0 \leq j \leq k)(\pi(j) \models \beta$ and $(\forall 0 \leq i < j)\pi(i) \models \alpha)$,
$s \models \mathrm{E}(\alpha \mathrm{R} \beta)$ iff $(\exists \pi \in \Pi_k(s))(\exists_{0 \leq j \leq k})(\pi(j) \models \alpha$ and $(\forall 0 \leq i \leq j)\ \pi(i) \models \beta)$
or $(\forall 0 \leq j \leq k)\ \pi(j) \models \beta$ and $(\exists 0 \leq i \leq k)(\pi(k) \rightarrow_{\mathcal{A}} \pi(i))$.

The first conjunct of $[\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1), \psi]_k$ represents all the possible submodels of $\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1)$ that consist of $f_k(\psi)$ $k-$paths of $\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1)$. The function f_k gives a bound for the number of k-paths in the submodel M_k of $\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1)$ such that the validity of ψ in M_k (i.e. validity in $\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1)$ with the bound k) is equivalent to the validity of ψ in $\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1)$. The function $f_k : \mathcal{F}(ECTL_y) \rightarrow \mathbb{N}$ is defined by:

- $f_k(p) = f_k(\neg p) = 0$, where $p \in \mathcal{PV}$,
- $f_k(\mathrm{X}_y\alpha) = f_k(\alpha)$,
- $f_k(\alpha \vee \beta) = max\{f_k(\alpha), f_k(\beta)\}$,
- $f_k(\alpha \wedge \beta) = f_k(\alpha) + f_k(\beta)$,
- $f_k(\mathrm{E}(\alpha \mathrm{U} \beta)) = k \cdot f_k(\alpha) + f_k(\beta) + 1$,
- $f_k(\mathrm{E}(\alpha \mathrm{R} \beta)) = k \cdot f_k(\beta) + f_k(\alpha) + 1$,

The second conjunct of $[\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1), \psi]_k$ encodes a number of constraints that must be satisfied on the submodel M_k of $\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1)$, which consists of all the k-paths of $\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1)$, for ψ to be satisfied. Once this translation is defined, checking satisfiability of an ECTL$_y$ formula can be done by means of a SAT-checker.

Let us assume that each state s of the discretised bounded model $\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1)$ is encoded by a bit-vector whose length, say b, depends on the number of locations, the number of clocks, the discretisation step, and c_{max}. So, each state s of $\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1)$ can be represented by a vector $w = (w[1], \ldots, w[b])$ (called a *global state variable*), where each $w[i]$ is a propositional variable for $i = 1, \ldots, b$. A finite sequence $(w_0, \ldots, w_k)$ of global state variables is called a *symbolic k-path*[3]. Moreover, we assume familiarity with basic BMC contributions; namely the definitions of propositional formulae $I_s(w)$, $p(w)$, $H(w, w')$, $\mathcal{R}(w, w')$, and $R_y(w, w')$ as defined in [26, 23].

The propositional formula $[\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1), \psi]_k$ is defined over a global state variable $w_{n,m}$, for $0 \leq m \leq k$ and $1 \leq n \leq f_k(\psi) + r$[4]; note that the index n denotes the number of a symbolic path, whereas the index m the position at that path. The formal definition of its first conjunct is the following:

$$[\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1)^{\psi,(l^0,v^0)}]_k := I_{(l^0,v^0)}(w_{0,f_k(\psi)+1}) \wedge \bigwedge_{n=1}^{f_k(\psi)} \bigwedge_{m=0}^{k-1} \mathcal{R}(w_{m,n}, w_{m+1,n})$$

The second conjunct of $[\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1), \psi]_k$, i.e. the formula $[\psi]_k^{[0,0]}$ is defined as follows:

$[p]_k^{[m,n]} := p(w_{m,n})$, $[\alpha \wedge \beta]_k^{[m,n]} := [\alpha]_k^{[m,n]} \wedge [\beta]_k^{[m,n]}$,
$[\neg p]_k^{[m,n]} := \neg p(w_{m,n})$, $[\alpha \vee \beta]_k^{[m,n]} := [\alpha]_k^{[m,n]} \vee [\beta]_k^{[m,n]}$,

[3] In general we shall need to consider not just one but a number of symbolic k-paths. This number depends on the formula ψ under investigation, and it is returned as the value $f_k(\psi)$ of the function f_k.

[4] Recall that r is the number of the non-trivial intervals in φ, where $\psi = \mathrm{cr}(\varphi)$.

$$[\mathrm{E}(\alpha \mathrm{U} \beta)]_k^{[m,n]} := \bigvee_{i=1}^{f_k(\psi)} (H(w_{m,n}, w_{0,i}) \wedge \bigvee_{j=0}^{k} ([\beta]_k^{[j,i]} \wedge \bigwedge_{l=0}^{j-1} [\alpha]_k^{[l,i]})),$$
$$[\mathrm{E}(\alpha \mathrm{R} \beta)]_k^{[m,n]} := \bigvee_{i=1}^{f_k(\psi)} (H(w_{m,n}, w_{0,i}) \wedge (\bigvee_{j=0}^{k} ([\alpha]_k^{[j,i]} \wedge \bigwedge_{l=0}^{j} [\beta]_k^{[l,i]}) \vee \bigwedge_{j=0}^{k} [\beta]_k^{[j,i]} \wedge \bigvee_{l=0}^{k} \mathcal{R}(w_{k,i}, w_{l,i}))),$$
$$[\mathrm{X}_y \alpha]_k^{[m,n]} := \bigvee_{j=1}^{r} (R_y(w_{m,n}, w_{0,f_k(\psi)+j}) \wedge [\alpha]_k^{[0,f_k(\psi)+j]}).$$

Theorem 1. *Let $\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1)$ be a discretised bounded model, and ψ an* ECTL$_y$ *formula. Then, $\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1) \models \psi$ iff there exists $k \in \mathbb{N}_+$ such that $[\psi]_k^{0,0} \wedge [\mathcal{D}(\mathcal{A}_\varphi, c_{max}+1)^{\psi,(l^0,v^0)}]_k$ is satisfiable.*

3.2 Experimental results

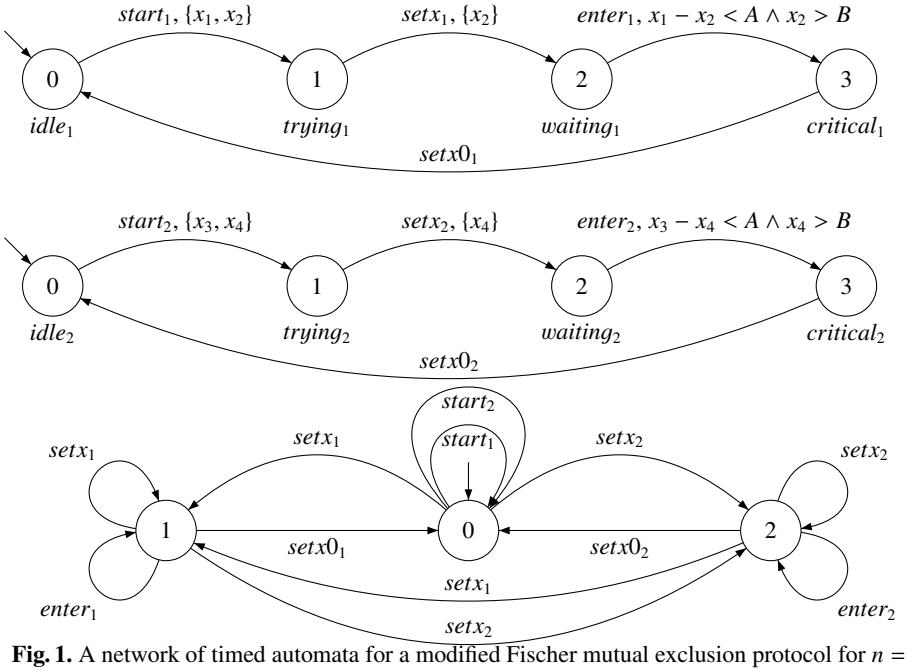

Fig. 1. A network of timed automata for a modified Fischer mutual exclusion protocol for $n = 2$

The new BMC algorithm presented in the paper has been implemented in the programming language C++, and some preliminary experiments has been performed. We have done this on the computer equipped with the processor AMD Athlon XP 1800 (1544 MHz), 768 MB main memory, and the operating system Linux.

As a real time system to be model checked we have taken a modified *Fischer mutual-exclusion protocol* (MUTEX) [33] that provides the network of timed automata with diagonal constraints. That protocol behaves in the same way as the Fischer mutual-exclusion protocol described in [29], and it is modelled by a net that is shown in Figure 1 and consists of n timed automata, each one modelling a process, together with one timed automaton modelling the global variable X, used to coordinate the processes'

access to the protocol. The global system is obtained as the parallel composition of the components.

The preservation of mutual exclusion depends on the time-delay constants A and B, i.e. on their relative values; in particular, the following holds: "*The Fischer protocol ensures mutual exclusion iff* $B < A$". So, we decided to test that protocol for satisfying of the following utility property: "*if* $A \leq B$*, then the mutual exclusion property is violated*". This property is given by the following TECTL formula:

$$\psi = \mathrm{EF}_{[A,\mathrm{inf}]}(\bigvee_{1 \leq i \neq j \leq n} (c_i \wedge c_j))$$

where the propositions c_i and c_j state that the process i (respectively j) is in its "critical section".

It is easy to see that for each $k > 0$ the value of the function f_k for the formula ψ is equal to 1. It follows that the counterexample, if exists, can be found on one symbolic $k-$path.

In Table 1 we give experimental results for the "violated" mutual exclusion property on one symbolic $k-$path, and in Table 2 we show $12-$path that is a witness for the property and it was automatically generated by our tool.

	BMC				**BerkMin**	
NoP	**variables**	**clauses**	**sec**	**MB**	**sec**	**MB**
2	21741	57704	1.1	4.1	0.4	22.5
10	96351	274136	5.8	19.0	3.4	25.3
20	201612	580772	14.3	41.4	8.7	34.3
50	597297	1740520	53.7	123.7	55.6	100.9
100	1524990	4478132	176.4	324.6	308.5	232.2
150	2789219	8225352	379.0	372.4	1447.8	257.2
200	4385319	12968252	682.2	809.9	4659.7	634.5

Table 1. Mutual exclusion violated. $k = 12$, $A = 2$ and $B = 1$.

4 Final Remarks

Our paper extends and improves the results of [26] and [23], where a general BMC approach for the existential fragment of TCTL and diagonal-free automata was described. The idea of BMC is taken from the paper [5]. Timed Automata have been defined and investigated in many papers [1, 29], but we adopt the definition given in [29]. Model checking for TCTL was considered by several authors using different approaches: over clock region models [1], on-the-fly [6], space-efficient [16], over minimal models [29, 13], and using SAT-methods [26, 27]. Our approach is closely related to [2] and [29], from which we draw the idea of translating of the model checking problem for TCTL to the model checking problem for *fair*-CTL.

The paper presents preliminary experimental results only, but they show that the proposed verification method is quite efficient and worth exploring. Since the literature for the formal verification of diagonal timed automata, does not provide any other TCTL model checking method that works on the fragments of models under consideration, we cannot compare our results with others.

	locations			clock valuations				
depth	$P1$	$P2$	Var	x_1	x_2	x_3	x_4	y
0	0	0	0	$0\frac{0}{256}$	$0\frac{0}{256}$	$0\frac{0}{256}$	$0\frac{0}{256}$	$0\frac{0}{256}$
1	0	0	0	$3\frac{0}{256}$	$3\frac{0}{256}$	$3\frac{0}{256}$	$3\frac{0}{256}$	$3\frac{0}{256}$
2	1	0	0	$0\frac{0}{256}$	$0\frac{0}{256}$	$3\frac{0}{256}$	$3\frac{0}{256}$	$3\frac{0}{256}$
3	1	0	0	$2\frac{0}{256}$	$2\frac{0}{256}$	$5\frac{0}{256}$	$5\frac{0}{256}$	$5\frac{0}{256}$
4	1	1	0	$2\frac{0}{256}$	$2\frac{0}{256}$	$0\frac{0}{256}$	$0\frac{0}{256}$	$5\frac{0}{256}$
5	1	1	0	$2\frac{9}{256}$	$2\frac{9}{256}$	$0\frac{9}{256}$	$0\frac{9}{256}$	$5\frac{9}{256}$
6	2	1	1	$2\frac{18}{256}$	$0\frac{0}{256}$	$0\frac{18}{256}$	$0\frac{18}{256}$	$5\frac{18}{256}$
7	2	1	1	$4\frac{25}{256}$	$2\frac{7}{256}$	$2\frac{25}{256}$	$2\frac{25}{256}$	$7\frac{25}{256}$
8	3	1	1	$4\frac{50}{256}$	$2\frac{14}{256}$	$2\frac{50}{256}$	$2\frac{50}{256}$	$7\frac{50}{256}$
9	3	1	1	$4\frac{53}{256}$	$2\frac{17}{256}$	$2\frac{53}{256}$	$2\frac{53}{256}$	$7\frac{53}{256}$
10	3	2	2	$4\frac{106}{256}$	$2\frac{34}{256}$	$2\frac{106}{256}$	$0\frac{0}{256}$	$7\frac{106}{256}$
11	3	2	2	$7\frac{121}{256}$	$5\frac{49}{256}$	$5\frac{121}{256}$	$3\frac{15}{256}$	$10\frac{121}{256}$
12	3	3	2	$7\frac{242}{256}$	$5\frac{98}{256}$	$5\frac{242}{256}$	$3\frac{30}{256}$	$10\frac{242}{256}$

Table 2. Witness

References

1. R. Alur, C. Courcoubetis, and D. Dill. Model checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.
2. R. Alur and D. Dill. A theory of Timed Automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
3. R. Alur, T. Feder, and T. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, 1996.
4. J. Bengtsson, B. Jonsson, J. Lilius, and W. Yi. Partial order reductions for timed systems. In *Proc. of CONCUR'98*, volume 1466 of *LNCS*, pages 485–500. Springer-Verlag, 1998.
5. A. Biere, A. Cimatti, E. Clarke, M.Fujita, and Y. Zhu. Symbolic model checking using SAT procedures instead of BDDs. In *Proc. of DAC'99*, pages 317–320, 1999.
6. A. Bouajjani, S. Tripakis, and S. Yovine. On-the-fly symbolic model checking for real-time systems. In *Proc. of RTSS'97*, pages 232–243. IEEE Computer Society, 1997.
7. R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transaction on Computers*, 35(8):677–691, 1986.
8. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts, 1999.
9. D. Dams, R. Gerth, B. Knaack, and R. Kuiper. Partial-order reduction techniques for real-time model checking. In *Proc. of FMICS'98*, pages 157 – 169, 1998.
10. D. Dams, O. Grumberg, and R. Gerth. Abstract interpretation of reactive systems: Abstractions preserving ACTL*, ECTL* and CTL*. In *Proc. of PROCOMET'94*. Elsevier Science Publishers, 1994.
11. C. Daws and S. Tripakis. Model checking of real-time reachability properties using abstractions. In *Proc. of TACAS'98*, volume 1384 of *LNCS*, pages 313–329. Springer-Verlag, 1998.
12. L. de Moura, H. Rueß, and M. Sorea. Lazy theorem proving for bounded model checking over infinite domains. In *Proc. of CADE-18*, volume 2392 of *LNCS*, pages 438–455. Springer-Verlag, 2002.

13. P. Dembiński, W. Penczek, and A. Półrola. Automated verification of infinite state concurrent systems: an improvement in model generation. In *Proc. of PPAM'01*, volume 2328 of *LNCS*, pages 247–255. Springer-Verlag, 2002.
14. E. A. Emerson. Temporal and modal logic. *Handbook of Theoretical Computer Science*, chapter 16, pages 996 – 1071. Elsevier Science Publishers, 1990.
15. E. A. Emerson and A. P. Sistla. Symmetry and model checking. *Formal Methods in System Design*, 9:105–131, 1995.
16. O. Kupferman, T. A. Henzinger, and M. Y. Vardi. A space-efficient on-the-fly algorithm for real-time model checking. In *Proc. of CONCUR'96*, volume 1119 of *LNCS*, pages 514–529. Springer-Verlag, 1996.
17. J. Lilius. Efficient state space search for Time Petri Nets. In *Proc. of MFCS'98*, volume 18 of *ENTCS*. Elsevier Science Publishers, 1999.
18. K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.
19. K. L. McMillan. Applying SAT methods in unbounded symbolic model checking. In *Proc. of CAV'02*, volume 2404 of *LNCS*, pages 250–264. Springer-Verlag, 2002.
20. P. Merlin and D. J. Farber. Recoverability of communication protocols – implication of a theoretical study. *IEEE Trans. on Communications*, 24(9)(9):1036–1043, 1976.
21. F. Pagani. Partial orders and verification of real-time systems. In *Proc. of FTRTFT'96*, volume 1135 of *LNCS*, pages 327–346. Springer-Verlag, 1996.
22. D. Peled. Partial order reduction: Linear and branching temporal logics and process algebras. In *Proc. of POMIV'96*, volume 29 of *ACM/AMS DIMACS Series*, pages 79–88. Amer. Math. Soc., 1996.
23. W. Penczek and A. Półrola. Specification and model checking of temporal properties in time Petri nets and timed automata. In *Proc. of ATPN'04*, volume 3099 of *LNCS*, pages 37–76. Springer-Verlag, 2004.
24. W. Penczek, A. Półrola, B. Woźna, and A. Zbrzezny. Bounded model checking for reachability testing in time Petri nets. In *Proc. of CS&P'04*, volume 170 of *Informatik-Berichte*, pages 124–135. Humboldt University, 2004.
25. W. Penczek, B. Woźna, and A. Zbrzezny. Bounded model checking for the universal fragment of CTL. *Fundamenta Informaticae*, 51(1-2):135–156, 2002.
26. W. Penczek, B. Woźna, and A. Zbrzezny. Towards bounded model checking for the universal fragment of TCTL. In *Proc. of FTRTFT'02*, volume 2469 of *LNCS*, pages 265–288. Springer-Verlag, 2002.
27. S. Seshia and R. Bryant. Unbounded, fully symbolic model checking of timed automata using boolean methods. In *Proc. of CAV'03*, volume 2725 of *LNCS*, pages 154–166. Springer-Verlag, 2003.
28. M. Sorea. Bounded model checking for Timed Automata. In *Proc. of MTCS'02*, volume 68(5) of *ENTCS*. Elsevier Science Publishers, 2002.
29. S. Tripakis and S. Yovine. Analysis of timed systems using time-abstracting bisimulations. *Formal Methods in System Design*, 18(1):25–68, 2001.
30. P. Wolper and P. Godefroid. Partial order methods for temporal verification. In *Proc. of CONCUR'93*, volume 715 of *LNCS*, pages 233–246. Springer-Verlag, 1993.
31. B. Woźna and A. Zbrzezny. Checking ACTL* properties of discrete timed automata via bounded model checking. In *Proc. of FORMATS'03*, volume 2791 of *LNCS*, pages 18–33. Springer-Verlag, 2004.
32. B. Woźna, A. Zbrzezny, and W. Penczek. Checking reachability properties for Timed Automata via SAT. *Fundamenta Informaticae*, 55(2):223–241, 2003.
33. A. Zbrzezny. Sat-based reachability checking for timed automata with diagonal constraints. *Fundamenta Informaticae*, 67(1-3):303–322, 2005.