Simple Bounded MTL Model Checking for Discrete Timed Automata (Extended abstract) ⋆

Agnieszka M. Zbrzezny and Andrzej Zbrzezny

IMCS, Jan Długosz University. Al. Armii Krajowej 13/15, 42-200 Częstochowa, Poland.
{agnieszka.zbrzezny,a.zbrzezny}@ajd.czest.pl

Abstract. We present a new translation of Metric Temporal Logic to the Linear Temporal Logic with a new set of the atomic propositions. We investigate a SAT-based bounded model checking method for Metric Temporal Logic that is interpreted over linear discrete infinite time models generated by discrete timed automata. We show how to implement the bounded model checking technique for Linear Temporal Logic with a new set of the atomic propositions and discrete timed automata, and as a case study we apply the technique in the analysis of the Timed Generic Pipeline Paradigm modelled by a network of discrete timed automata. We also present a comparison of the two translations of Metric Temporal Logic on common instances that can be scaled up to for performance evaluation. The theoretical description is supported by the experimental results that demonstrate the efficiency of the method.

1 Introduction

Bounded model checking [2, 3, 5] (BMC) is one of the symbolic model checking technique designed for finding witnesses for existential properties or counterexamples for universal properties. Its main idea is to consider a model reduced to a specific depth. The method works by mapping a bounded model checking problem to the satisfiability problem (SAT). For metric temporal logic (MTL) [4] and discrete time automata [1] the BMC method can by described as follows: given a model $\mathcal{M}$ for a discrete timed automaton , an MTL formula φ, and a bound k, a model checker creates a propositional formula $[\mathcal{M}, \varphi]_k$ that is satisfiable if and only if the formula φ is true in the model $\mathcal{M}$.

The novelty of our paper lies in :

1. defining a translation of the existential model checking problem for MTL to the existential model checking problem for linear temporal logic with additional propositional variables q_I. This logic is denoted by LTL$_q$;
2. defining bounded sematics for LTL$_q$ and defining the BMC algorithm;
3. implementing the new method.

The translation from MTL to LTL$_q$ requires neither new clocks nor new transitions, whereas the translation to HLTL [7] requires as many new clocks as there are intervals in a given formula. It also requires an exponential number of resetting transitions.

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

Moreover, our BMC method needs only one path, whereas the BMC method from [7] needs a number of paths depending on a given formula φ. Thus, one may expect that our method is much more effective since intuition is that an encoding which results in fewer variables and clauses is usually easier to solve.

Finally, we evaluate the BMC method experimentally by means of a timed generic pipeline paradigm (TGPP), which we model by a network of discrete timed automata and compare with the corresponding method [7].

The rest of the paper is structured as follows. In Section 2 we briefly recall the basic notion used through the paper. In Section 3 we define the translation to LTL_q. In Section 4 we define the BMC method for LTL_q. In Section 5 we discuss our experimental results. In Section 6 we conclude the paper.

2 Preliminaries

2.1 Discrete Timed Automata

Let $\mathbb{N}$ be a set of natural numbers. We assume a finite set $\mathbb{X} = \{x_0, \ldots, x_{n-1}\}$ of variables, called *clocks*. Each clock is a variable ranging over a set of non-negative natural numbers.

A *clock valuation* is a total function $v : \mathbb{X} \mapsto \mathbb{N}$ that assigns to each clock x a non-negative integer value $v(x)$. The set of all the clock valuations is denoted by $\mathbb{N}^n$. For $X \subseteq \mathbb{X}$, the valuation $v' = v[X := 0]$ is defined as: $\forall x \in X,\ v'(x) = 0$ and $\forall x \in \mathbb{X} \setminus X,\ v'(x) = v(x)$. For $\delta \in \mathbb{N}$, $v + \delta$ denotes the valuation v'' such that $\forall x \in \mathbb{X}, v''(x) = v(x) + \delta$. Let $x \in \mathbb{X}$, $c \in \mathbb{N}$, and $\sim \in \{<, \leqslant, =, \geqslant, >\}$. The set $\mathcal{C}(\mathbb{X})$ of *clock constraints* over the set of clocks $\mathbb{X}$ is defined by the following grammar:

$$\mathfrak{cc} := x \sim c \mid \mathfrak{cc} \wedge \mathfrak{cc}.$$

Let v be a clock valuation, and $\mathfrak{cc} \in \mathcal{C}(\mathbb{X})$. A clock valuation v satisfies a clock constraint $\mathfrak{cc}$, written as $v \models \mathfrak{cc}$, iff $\mathfrak{cc}$ evaluates to true using the clock values given by the valuation v.

Definition 1. *A* discrete timed automaton $\mathcal{A}$ *is a tuple* $(Act, Loc, \ell^0, T, \mathbb{X}, Inv, AP, V)$*, where* Act *is a finite set of actions,* Loc *is a finite set of locations,* $\ell^0 \in Loc$ *is an initial location,* $T \subseteq Loc \times Act \times \mathcal{C}(\mathbb{X}) \times 2^{\mathbb{X}} \times Loc$ *is a transition relation,* $\mathbb{X}$ *is a finite set of clocks,* $Inv : Loc \mapsto \mathcal{C}(\mathbb{X})$ *is a state invariant function,* AP *is a set of atomic proposition, and* $V : Loc \mapsto 2^{AP}$ *is a valuation function assigning to each location a set of atomic propositions true in this location.*

Each element $t \in T$ *is denoted by* $\ell \xrightarrow{\sigma, \mathfrak{cc}, X} \ell'$*, and it represents a transition from location* ℓ *to location* ℓ' *on the input action* σ*.* $X \subseteq \mathbb{X}$ *is the set of the clocks to be reset with this transition, and* $\mathfrak{cc} \in \mathcal{C}(\mathbb{X})$ *is the enabling condition for* t*.*

The semantics of the discrete timed automaton is defined by associating a transition system with it, which we call a *concrete model*.

Definition 2. *Let* $\mathcal{A} = (Act, Loc, \ell^0, T, \mathbb{X}, Inv, AP, V)$ *be a discrete timed automaton, and* v^0 *a clock valuation such that* $\forall x \in \mathbb{X},\ v^0(x) = 0$*. A* concrete model *for* $\mathcal{A}$ *is*

a tuple $\mathcal{M}_{\mathcal{A}} = (Q, q^0, \longrightarrow, \mathcal{V})$*, where* $Q = Loc \times \mathbb{N}^n$ *is a set of the concrete states,* $q^0 = (\ell^0, v^0)$ *is the initial state,* $\longrightarrow \subseteq Q \times Q$ *is a total binary relation on* Q *defined by action and time transitions as follows. For* $\sigma \in Act$ *and* $\delta \in \mathbb{N}$*,*

1. *Action transition:* $(\ell, v) \xrightarrow{\sigma} (\ell', v')$ *iff there is a transition* $\ell \xrightarrow{\sigma, \mathfrak{cc}, X} \ell' \in T$ *such that* $v \models \mathfrak{cc} \wedge Inv(\ell)$ *and* $v' = v[X := 0]$ *and* $v' \models Inv(\ell')$*,*
2. *Time transition:* $(\ell, v) \xrightarrow{\delta} (\ell, v + \delta)$ *iff* $v \models Inv(\ell)$ *and* $v + \delta \models Inv(\ell)$*.*

$\mathcal{V} : Q \mapsto 2^{AP}$ *is a valuation function such that* $\mathcal{V}((\ell, v)) = V(\ell)$ *for all* $(\ell, v) \in Q$*.*

A *run* ρ of $\mathcal{A}$ is an infinite sequence of concrete states: $q_0 \xrightarrow{\delta_0, \sigma_0} q_1 \xrightarrow{\delta_1, \sigma_1} q_2 \xrightarrow{\delta_2, \sigma_2} \dots$ such that $q_i \in Q$, $\sigma_i \in Act$, and $\delta_i \in \mathbb{N}_+$ for each $i \in \mathbb{N}$. Notice that our runs are *strongly monotonic*. This is because the definition of the run does not permit two consecutive actions to be performed one after the other, i.e., between each two actions some time must pass.

2.2 Metric Temporal Logic (MTL)

Let $p \in AP$, and I be an interval in $\mathbb{N}$ of the form: $[a, b)$ or $[a, \infty)$, for $a, b \in \mathbb{N}$ and $a \neq b$. The MTL in release positive normal form is defined by the following grammar:

$$\alpha := \mathbf{true} \mid \mathbf{false} \mid p \mid \neg p \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \alpha \mathbf{U}_I \alpha \mid \mathbf{G}_I \alpha.$$

Intuitively, $\mathbf{U}_I$ and $\mathbf{G}_I$ are the operators for *bounded until* and for *bounded always*. The formula $\alpha \mathbf{U}_I \beta$ is true in a computation if β is true in the interval I at least in one state and always earlier α holds. The formula $\mathbf{G}_I \alpha$ is true in a computation α is true at all states of the computation that are in the interval I. The derived basic modality is defined as follows: $\mathbf{F}_I \alpha \stackrel{def}{=} \mathbf{true} \mathbf{U}_I \alpha$ (*bounded eventually*).

Let $\mathcal{A}$ be a discrete timed automaton, $\mathcal{M}_{\mathcal{A}} = (Q, q^0, \longrightarrow, \mathcal{V})$ a concrete model for $\mathcal{A}$, $\rho : q_0 \xrightarrow{\delta_0, \sigma_0} q_1 \xrightarrow{\delta_1, \sigma_1} q_2 \xrightarrow{\delta_2, \sigma_2} \dots$ a run of $\mathcal{A}$, and α, β formulae of MTL. In order to define the satisfiability relation for MTL, we need to define the notion of a *discrete path* λ_ρ *corresponding to run* ρ [6]. This can be done in a unique way because of the assumption that $\delta_i \in \mathbb{N}_+$. First, define the sequence $\Delta_0 = [b_0, b_1), \Delta_1 = [b_1, b_2), \Delta_2 = [b_2, b_3), \dots$ of pairwise disjoint intervals, where: $b_0 = 0$, and $b_i = b_{i-1} + \delta_{i-1}$ if $i > 0$. Now, for each $t \in \mathbb{N}$, let $idx_\rho(t)$ denote the unique index i such that $t \in \Delta_i$. A *discrete path* (or *path*) λ_ρ *corresponding to* ρ is a mapping $\lambda_\rho : \mathbb{N} \mapsto Q$ such that $\lambda_\rho(t) = (\ell_i, v_i + t - b_i)$, where $i = idx_\rho(t)$. Given $t \in \mathbb{N}$, the suffix λ_ρ^t of a path λ_ρ at time t is a path defined as: $\forall i \in \mathbb{N}$, $\lambda_\rho^t(i) = \lambda_\rho(t + i)$.

In order to improve readability, in the following definition we write $\lambda_\rho^t \models_{\mathrm{MTL}} \varphi$ instead of $\mathcal{M}_\varphi, \lambda_\rho^t \models_{\mathrm{MTL}} \varphi$, for any MTL formula φ.

Definition 3. *The* satisfiability *relation* $\models_{\mathrm{MTL}}$*, which indicates truth of an* MTL *formula in the concrete model* $\mathcal{M}_{\mathcal{A}}$ *along a path* λ_ρ *at time* $t \in \mathbb{N}$*, is defined inductively as follows:*

- $\lambda_\rho^t \models_{\mathrm{MTL}} \mathbf{true}$*,* $\lambda_\rho^t \not\models_{\mathrm{MTL}} \mathbf{false}$*,*
- $\lambda_\rho^t \models_{\mathrm{MTL}} p$ *iff* $p \in \mathcal{V}(\lambda_\rho(t))$*,* $\lambda_\rho^t \models_{\mathrm{MTL}} \neg p$ *iff* $p \notin \mathcal{V}(\lambda_\rho(t))$*,*

- $\lambda_\rho^t \models_{\mathrm{MTL}} \alpha \wedge \beta$ *iff* $\lambda_\rho^t \models_{\mathrm{MTL}} \alpha$ *and* $\lambda_\rho^t \models_{\mathrm{MTL}} \beta$,
- $\lambda_\rho^t \models_{\mathrm{MTL}} \alpha \vee \beta$ *iff* $\lambda_\rho^t \models_{\mathrm{MTL}} \alpha$ *or* $\lambda_\rho^t \models_{\mathrm{MTL}} \beta$,
- $\lambda_\rho^t \models_{\mathrm{MTL}} \alpha \mathbf{U}_I \beta$ *iff* $(\exists t' \in I)(\lambda_\rho^{t+t'} \models_{\mathrm{MTL}} \beta$ *and* $(\forall 0 \leqslant t'' < t')\lambda_\rho^{t+t''} \models_{\mathrm{MTL}} \alpha)$,
- $\lambda_\rho^t \models_{\mathrm{MTL}} \mathbf{G}_I \beta$ *iff* $(\forall t' \in I)(\lambda_\rho^{t+t'} \models_{\mathrm{MTL}} \beta)$.

As $\lambda_\rho^0 = \lambda_\rho$, we shall write $\mathcal{M}_\mathcal{A}, \lambda_\rho \models_{\mathrm{MTL}} \varphi$ for $\mathcal{M}_\mathcal{A}, \lambda_\rho^0 \models_{\mathrm{MTL}} \varphi$. An MTL formula φ is *existentially valid* in the model $\mathcal{M}_\mathcal{A}$, denoted $\mathcal{M}_\mathcal{A} \models_{\mathrm{MTL}} \mathbf{E}\varphi$, if, and only if $\mathcal{M}_\mathcal{A}, \lambda_\rho \models_{\mathrm{MTL}} \varphi$ for some path λ_ρ starting in the initial state of $\mathcal{M}_\mathcal{A}$. Determining whether an MTL formula φ is existentially valid in a given model is called an *existential model checking problem*.

3 Translation from MTL to LTL$_q$

3.1 Abstract model

Let φ be an MTL formula and $\mathcal{A} = (Act, Loc, \ell^0, T, \mathbb{X}, Inv, AP, V)$ be a discrete timed automaton with $\mathbb{X} = \{x_0, \ldots, x_{n-1}\}$. For each $j \in \{0, \ldots, n-1\}$, let c_j^{max} be the largest constant appearing in intervals of φ and in any enabling condition involving the clock x_j and used in the state invariants and guards of $\mathcal{A}$. For two clock valuations v and v' in $\mathbb{N}^n$, we say that $v \simeq v'$ iff for each $0 \leqslant j < n$ either $v(x_j) > c_j^{max}$ and $v'(x_j) > c_j^{max}$ or $v(x) \leqslant c_j^{max}$ and $v'(x) \leqslant c_j^{max}$ and $v(x) = v'(x)$.

It is well known, that the relation $\simeq$ is an equivalence relation, what gives rise to construct an finite abstract model. To this end we define the set of possible values of the clock x_j in the abstract model as $\mathbb{D}_j = \{0, \ldots, c_j^{max}+1\}$ for $0 \leqslant j < n$. Moreover, for two clock valuations v and v' in $\mathbb{D}_0 \times \ldots \times \mathbb{D}_{n-1}$, we say that v' is the *time successor* of v (denoted $succ(v)$) as follows: for each $0 \leqslant j < n$,

$$succ(v)(x_j) = \begin{cases} v(x_j) + 1, \text{ if } v(x_j) \leqslant c_j^{max}, \\ c_j^{max} + 1, \text{ if } v(x_j) = c_j^{max} + 1. \end{cases}$$

Definition 4. *Let $\mathcal{A} = (Act, Loc, \ell^0, T, \mathbb{X}, Inv, AP, V)$ be a discrete timed automaton, and φ an MTL formula build over the set AP of atomic propositions. An* abstract model *for the automaton $\mathcal{A}$ and the formula φ is a tuple $\mathcal{M}_\varphi = (\widehat{S}, s^0, \hookrightarrow, \widehat{\mathcal{V}})$, where $\widehat{S} = L \times (\mathbb{D}_0 \times \ldots \times \mathbb{D}_{n-1})$ is the set of abstract states, $s^0 = (\ell^0, \{0\}^{n+1})$ is the initial state, $\widehat{\mathcal{V}} : \widehat{S} \to 2^{AP}$ is a valuation function such that for all $p \in AP$, $p \in \widehat{\mathcal{V}}((\ell, v))$ iff $p \in V(\ell)$, and $\hookrightarrow \subseteq S \times Act' \times S$, where $Act' = Act \cup \{\tau\}$, is a transition relation defined by the time and action transitions:*

- *Time transition: $(\ell, v) \overset{\tau}{\hookrightarrow} (\ell, v')$ iff $v \models Inv(\ell)$, $v' = succ(v)$, and $v' \models Inv(\ell)$,*
- *Action transition: for any $\sigma \in Act$, $(\ell, v) \overset{\sigma}{\hookrightarrow} (\ell', v')$ iff there exists a transition $\ell \overset{\sigma, \mathfrak{cc}, X}{\longrightarrow} \ell' \in T$ such that $v \models \mathfrak{cc} \wedge Inv(\ell)$, $v' = v[X := 0]$, and $v' \models Inv(\ell')$.*

Definition 5. *A* path *in $\mathcal{M}_\varphi$ is a sequence $\pi = (s_0, s_1, \ldots)$ of states such that for each $j \in \mathbb{N}$, either $(s_j \overset{\tau}{\hookrightarrow} s_{j+1})$ or $(s_j \overset{\sigma}{\hookrightarrow} s_{j+1})$ for some $\sigma \in Act$, and every action transition is preceded by at least one time transition.*

The above definition of the path ensures that the first transition is the time one, and that between each two action transitions at least one time transition appears.

For a path π, $\pi(j)$ denotes the j-th state s_j of π, $\pi[..j] = (\pi(0), \ldots, \pi(j))$ denotes the j-th prefix of π ending with $\pi(j)$, and $\pi^j = (s_j, s_{j+1}, \ldots)$ denotes the j-th suffix of π starting with $\pi(j)$.

Given a path π one can define a function $\zeta_\pi : \mathbb{N} \mapsto \mathbb{N}$ such that for each $j \geqslant 0$, $\zeta_\pi(j)$ is equal to the number of time transitions on the prefix $\pi[..j]$. Let us note that for each $j \geqslant 0$, $\zeta_\pi(j)$ gives the value of the global time in the j-th state of the path π.

3.2 The logic LTL$_q$

Let $\mathcal{I}$ be the set of all intervals in $\mathbb{N}$. Let $AP_{\mathcal{I}} = \{q_I \mid I \in \mathcal{I}\}$. The LTL$_q$ formulae in the negation normal form are defined by the following grammar:

$$\psi ::= \mathbf{true} \mid \mathbf{false} \mid p \mid \neg p \mid q_I \mid \neg q_I \mid \psi \wedge \psi \mid \psi \vee \psi \mid \psi \mathbf{U} \psi \mid \mathbf{G}\psi,$$

where $p \in AP$ and $q_I \in AP_{\mathcal{I}}$. The temporal modalities $\mathbf{U}$ and $\mathbf{G}$ are, respectively, named as the *until* and the *always*. The derived basic temporal modality for *eventually* is defined in the standard way: $\mathbf{F}\psi \stackrel{def}{=} \mathbf{true}\mathbf{U}\psi$.

In order to improve readability, in the following definition we write $\langle \pi, m\rangle \models_k \psi$ instead of $\mathcal{M}_\varphi, \langle \pi, m\rangle \models_k \psi$, for any LTL$_q$ formula ψ.

Definition 6. *The* satisfiability *relation $\models^d$, which indicates truth of an LTL$_q$ formula in the abstract model $\mathcal{M}_\varphi$ along the path π with the starting point m and at the depth $d \geqslant m$, is defined inductively as follows:*

- $\langle \pi, m\rangle \models^d \mathbf{true}$, $\quad \langle \pi, m\rangle \not\models^d \mathbf{false}$,
- $\langle \pi, m\rangle \models^d p$ *iff* $p \in \mathcal{V}(\pi(d))$, $\quad \langle \pi, m\rangle \models^d \neg p$ *iff* $p \notin \mathcal{V}(\pi(d))$,
- $\langle \pi, m\rangle \models^d q_I$ *iff* $\zeta_\pi(d) - \zeta_\pi(m) \in I$,
- $\langle \pi, m\rangle \models^d \neg q_I$ *iff* $\zeta_\pi(d) - \zeta_\pi(m) \notin I$,
- $\langle \pi, m\rangle \models^d \alpha \wedge \beta$ *iff* $\langle \pi, m\rangle \models^d \alpha$ *and* $\langle \pi, m\rangle \models^d \beta$,
- $\langle \pi, m\rangle \models^d \alpha \vee \beta$ *iff* $\langle \pi, m\rangle \models^d \alpha$ *or* $\langle \pi, m\rangle \models^d \beta$,
- $\langle \pi, m\rangle \models^d \alpha \mathbf{U} \beta$ *iff* $(\exists j \geqslant d)(\langle \pi, d\rangle \models^j \beta$ *and* $(\forall d \leqslant i < j)\, \langle \pi, d\rangle \models^i \alpha)$,
- $\langle \pi, m\rangle \models^d \mathbf{G}\beta$ *iff* $(\forall j \geqslant d)\, \langle \pi, d\rangle \models^j \beta$.

An LTL$_q$ formula ψ *existentially holds* in the model $\mathcal{M}_\varphi$, written $\mathcal{M}_\varphi \models \mathbf{E}\psi$, if, and only if $\mathcal{M}_\varphi, \langle \pi, 0\rangle \models^0 \psi$ for some path π starting at the initial state. The *existential model checking problem* asks whether $\mathcal{M}_\varphi \models \mathbf{E}\psi$.

3.3 Translation

Let $p \in AP$, α and β be formulae of MTL. We define the translation from MTL into LTL$_q$ as a function $\mathrm{tr} : \mathrm{MTL} \to \mathrm{LTL}_q$ in the following way:

- $\mathrm{tr}(\mathbf{true}) = \mathbf{true}$, $\mathrm{tr}(\mathbf{false}) = \mathbf{false}$, $\mathrm{tr}(p) = p$, $\mathrm{tr}(\neg p) = \neg p$,
- $\mathrm{tr}(\alpha \wedge \beta) = \mathrm{tr}(\alpha) \wedge \mathrm{tr}(\beta)$, $\mathrm{tr}(\alpha \vee \beta) = \mathrm{tr}(\alpha) \vee \mathrm{tr}(\beta)$,
- $\mathrm{tr}(\alpha \mathbf{U}_I \beta) = \mathrm{tr}(\alpha) \mathbf{U} \mathrm{tr}(q_I \wedge \beta)$, $\mathrm{tr}(\mathbf{G}_I \beta) = \mathbf{G}(\neg q_I \vee \mathrm{tr}(\beta))$

Observe that the translation of literals as well as logical connectives is straightforward. The translation of the $\mathbf{U}_I$ operator ensures that β holds somewhere in the interval I (this is expressed by the requirement $q_I \wedge \mathrm{tr}(\beta)$), and α holds always before β. Similarly, the translation of the $\mathbf{G}_I$ operator ensures that β always holds in the interval I (this is expressed by the requirement $\neg q_I \vee \mathrm{tr}(\beta)$).

Theorem 1. *Let $\mathcal{A}$ be a discrete timed automaton, $\mathcal{M}_{\mathcal{A}}$ the concrete model for $\mathcal{A}$, φ an MTL formula, and $\mathcal{M}_{\varphi}$ the abstract model for the automaton $\mathcal{A}$ and the formula φ. Then, $\mathcal{M}_{\mathcal{A}} \models \mathbf{E}\varphi$ if, and only if $\mathcal{M}_{\varphi} \models \mathbf{E}\mathrm{tr}(\varphi)$.*

4 Bounded model checking

In this section we define a *bounded semantics* for LTL_q in order to translate the *existential model checking problem* for LTL_q into the satisfiability problem.

4.1 Bounded semantics

To define the bounded semantics one needs to represent infinite paths in a model in a special way. To this aim, we recall the notions of *k-paths* and *loops* [8].

Definition 7. *Let $\mathcal{M}_{\varphi}$ be a model, $k \in \mathbb{N}$, and $0 \leqslant l \leqslant k$. A* k-path *is a pair (π, l), also denoted by π_l, where π is a finite sequence $\pi = (s_0, \ldots, s_k)$ of states such that for each $0 \leqslant j < k$, either $(s_j \overset{\tau}{\hookrightarrow} s_{j+1})$ or $(s_j \overset{\sigma}{\hookrightarrow} s_{j+1})$ for some $\sigma \in Act$, and every action transition is preceded by at least one time transition. A k-path π_l is a* loop*, written $\circlearrowright(\pi_l)$ for short, if $l < k$ and $\pi(k) = \pi(l)$.*

If a k-path π_l is a loop it represents the infinite path of the form uv^{ω}, where $u = (\pi(0), \ldots, \pi(l))$ and $v = (\pi(l+1), \ldots, \pi(k))$. We denote this unique path by $\widetilde{\pi}_l$. Note that for each $j \in \mathbb{N}$, $\widetilde{\pi}_l^{\,l+j} = \widetilde{\pi}_l^{\,k+j}$.

In the definition of bounded semantics for variables from $AP_{\mathcal{I}}$ one needs to use only a finite prefix of the sequence $(\zeta_{\widetilde{\pi}_l}(0), \zeta_{\widetilde{\pi}_l}(1), \ldots)$. Namely, for a k-path π_l that is not a loop the prefix of the length k is needed, and for a k-path π_l that is a loop the prefix of the length $k + k - l$ is needed.

In order to improve readability, in the following definition we write $\langle \pi_l, m \rangle \models_k \psi$ instead of $\mathcal{M}_{\varphi}, \langle \pi_l, m \rangle \models_k \psi$, for any LTL_q formula ψ.

Definition 8 (Bounded semantics). *Let $\mathcal{M}_{\varphi}$ be the abstract model, π_l be a k-path in $\mathcal{M}_{\varphi}$, and $0 \leqslant m, d \leqslant k$. The relation $\models_k^d$ is defined inductively as follows:*

- $\langle \pi_l, m \rangle \models_k^d \mathbf{true}$, $\quad \langle \pi_l, m \rangle \not\models_k^d \mathbf{false}$,
- $\langle \pi_l, m \rangle \models_k^d p$ *iff* $p \in \mathcal{V}(\pi_l(d))$, $\quad \langle \pi_l, m \rangle \models_k^d \neg p$ *iff* $p \notin \mathcal{V}(\pi_l(d))$,
- $\langle \pi_l, m \rangle \models_k^d q_I$ *iff* $\begin{cases} \zeta_{\widetilde{\pi}_l}(d) - \zeta_{\widetilde{\pi}_l}(m) \in I, & \text{if } \pi_l \text{ is not a loop}, \\ \zeta_{\widetilde{\pi}_l}(d) - \zeta_{\widetilde{\pi}_l}(m) \in I, & \text{if } \pi_l \text{ is a loop and } d \geqslant m, \\ \zeta_{\widetilde{\pi}_l}(d + k - l) - \zeta_{\widetilde{\pi}_l}(m) \in I, & \text{if } \pi_l \text{ is a loop and } d < m, \end{cases}$
- $\langle \pi_l, m \rangle \models_k^d \neg q_I$ *iff* $\langle \pi_l, m \rangle \not\models_k^d q_I$

– $\langle \pi_l, m\rangle \models_k^d \alpha \wedge \beta$ *iff* $\langle \pi_l, m\rangle \models_k^d \alpha$ *and* $\langle \pi_l, m\rangle \models_k^d \beta$,
– $\langle \pi_l, m\rangle \models_k^d \alpha \vee \beta$ *iff* $\langle \pi_l, m\rangle \models_k^d \alpha$ *or* $\langle \pi_l, m\rangle \models_k^d \beta$,
– $\langle \pi_l, m\rangle \models_k^d \alpha \mathbf{U} \beta$ *iff* $(\exists_{d \leqslant j \leqslant k})\big(\langle \pi_l, d\rangle \models_k^j \beta$ *and* $(\forall_{d \leqslant i < j})\, \langle \pi_l, d\rangle \models_k^j \alpha\big)$
or $\big(\circlearrowright(\pi_l)$ *and* $(\exists_{l<j<d})\, \langle \pi_l, d\rangle \models_k^j \beta$ *and* $(\forall_{l<i<k})\ \langle \pi_l, d\rangle \models_k^j \alpha$
and $(\forall_{d \leqslant i \leqslant k})\ \langle \pi_l, d\rangle \models_k^j \alpha\big)$,
– $\langle \pi_l, m\rangle \models_k^d \mathbf{G}\beta$ *iff* $\circlearrowright(\pi_l)$ *and* $(\forall_{j \leqslant k}) j \geqslant \min(d, l)$ *implies* $\langle \pi_l, d\rangle \models_k^j \beta$.

An LTL$_q$ formula ψ *existentially k-holds* in the model $\mathcal{M}_\varphi$, written $\mathcal{M}_\varphi \models_k \mathbf{E}\psi$, if, and only if $\mathcal{M}_\varphi, \langle \pi, 0\rangle \models_k^0 \psi$ for some path π starting at the initial state.

Theorem 2. *Let $\mathcal{A}$ be a discrete timed automaton, φ an MTL formula, and $\mathcal{M}_\varphi$ the abstract model for the automaton $\mathcal{A}$ and the formula φ. Moreover, let $\psi = \mathrm{tr}(\varphi)$. Then, $\mathcal{M}_\varphi \models \mathbf{E}\psi$ if, and only if there exists $k \geqslant 0$ such that $\mathcal{M}_\varphi \models_k \mathbf{E}\psi$.*

4.2 Translation to SAT

The last step of our method is the standard one (see [8, 7]). It consists in encoding the transition relation of $\mathcal{M}_\varphi$ and the LTL formula $\mathrm{tr}(\varphi)$. The only novelty lies in encoding of the finite prefix of the sequence $(\zeta_{\widetilde{\pi}_l}(0), \zeta_{\widetilde{\pi}_l}(1), \ldots)$. The translation to SAT results in the propositional formula $[\mathcal{M}_\varphi, \mathrm{tr}(\varphi)]_k$ with the property expressed in the following theorem.

Theorem 3. *Let $\mathcal{M}_\varphi$ be an abstract model. Then, for every $k \in \mathbb{N}$, $\mathcal{M}_\varphi \models_k^d \mathbf{E}\mathrm{tr}(\varphi)$ if, and only if, the propositional formula $[\mathcal{M}_\varphi, \mathrm{tr}(\varphi)]_k$ is satisfiable.*

5 Experimental results

In this section we experimentally evaluate the performance of our new translation. We have conducted the experiments using the slightly modified timed generic pipeline paradigm (TGPP) [7].

5.1 Timed Generic Pipeline Paradigm

The Timed Generic Pipeline Paradigm (TGPP) discrete timed automata model shown in Figure 1 consists of Producer producing data within the certain time interval $([a, b])$ or being inactive, Consumer receiving data within the certain time interval $([c, d])$ or being inactive within the certain time interval $([g, h])$, and a chain of n intermediate Nodes which can be ready for receiving data within the certain time interval $([c, d])$, processing data within the certain time interval $([e, f])$ or sending data. We assume that $a = c = e = g = 1$ and $b = d = f = h = 2 \cdot n + 2$, where n represents number of nodes in the TGPP.

To compare our experimental results with [7], we have tested the TGPP discrete timed automata model, scaled in the number of intermediate nodes on the following MTL formulae that existentially hold in the model of TGPP (n is the number of nodes):

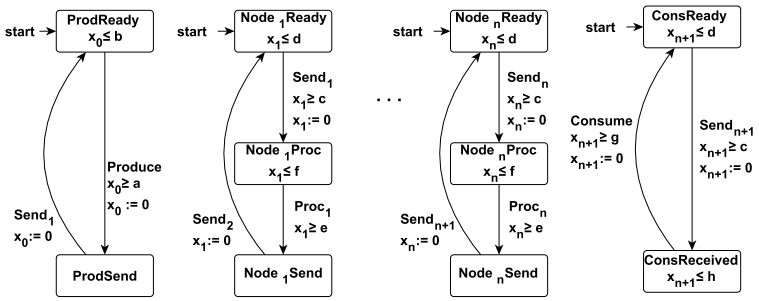

Fig. 1: The TGPP system.

- $\varphi_0 = \mathbf{F}_{[0,2\cdot n+3)}(ConsReceived)$. It expresses that Consumer receives the data in at most $2 \cdot n + 3$ time units.
- $\varphi_1 = \mathbf{G}_{[0,2\cdot n+2)}(ConsReady)$. It states that the Consumer is always forced to receive the data in $2 \cdot n + 2$ time units.
- $\varphi_2 = \mathbf{G}_{[0,\infty)}(ProdReady \vee ConsReady)$. It states that always either the Producer has sent the data or the Consumer has received the data.
- $\varphi_3 = \mathbf{F}_{[0,2\cdot n+3)}(\mathbf{G}_{[0,\infty)}(ProdSend \vee ConsReceived))$. It states that eventually in time less then $2 \cdot n + 3$ it is always the case that the Producer is ready to send the data or the Consumer has received the data.
- $\varphi_4 = \mathbf{G}_{[0,\infty)}(\mathbf{F}_{[0,2\cdot n+3)}(ConsReceived))$. It states that the Consumer infinitely often is receiving the data in time less then $2 \cdot n + 3$.
- $\varphi_5 = \mathbf{G}_{[0,\infty)}(\mathbf{F}_{[0,2\cdot n+3)}(ProdSend) \wedge \mathbf{G}_{[0,\infty)}(\mathbf{F}_{[0,2\cdot n+3)}(ConsReceived))$. It states that the Producer infinitely often is sending the data in the time less then $2 \cdot n + 3$ and the Consumer infinitely often is receiving the data in time less then $2 \cdot n + 3$.

5.2 Performance evaluation

We have performed our experimental results on a computer equipped with I7-3770 processor, 32 GB of RAM, and the operating system Linux with the kernel 4.6.4. Our SAT-based BMC algorithms are implemented as standalone programs written in the programming language `C++`. We used the state of the art SAT-solver CryptoMiniSat5 (`http://www.msoos.org/`).

All the benchmarks together with instructions on how to reproduce our experimental results can be found at the web page `http://ajd.czest.pl/~a.zbrzezny/bmc.html`.

The number of considered k-paths (f_k) for the new translation is always equal to 1 and for the old translation is respectively equal to: $f_k(\varphi_0) = 2$, $f_k(\varphi_1) = 2$, $f_k(\varphi_2) = 2$, $f_k(\varphi_3) = 3$, $f_k(\varphi_4) = 8 \cdot n + 9$, $f_k(\varphi_5) = 16 \cdot n + 17$. The length of the witness for the formula φ_0 is equal to $4 \cdot n + 4$; for the formula φ_1 is equal to $8 \cdot n + 6$; for the formula φ_2 and is equal to $8 \cdot n + 6$; for the formula φ_3 is equal to $8 \cdot n + 15$; for the formula φ_4 is equal to $8 \cdot n + 6$; for the formula φ_5 is equal to $8 \cdot n + 6$.

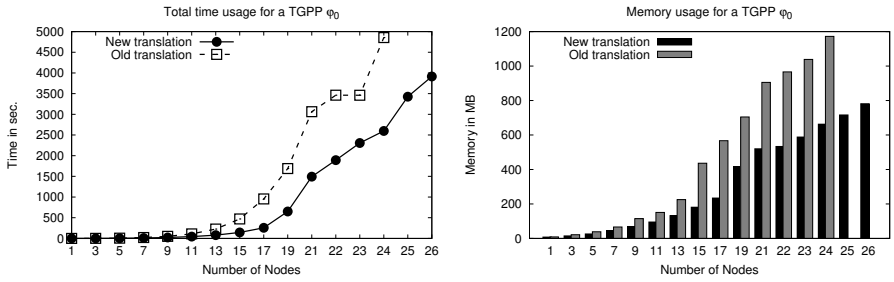

Fig. 2: φ_0: TGPP with n nodes.

From Fig. 2 one can observe that the new method is able to verify the formula φ_0 for TGPP with 27 nodes. The old method is able to verify the formula φ_0 for TGPP with 24 nodes. The memory usage for the old method is 1.72 times higher than for the new method for 24 nodes.

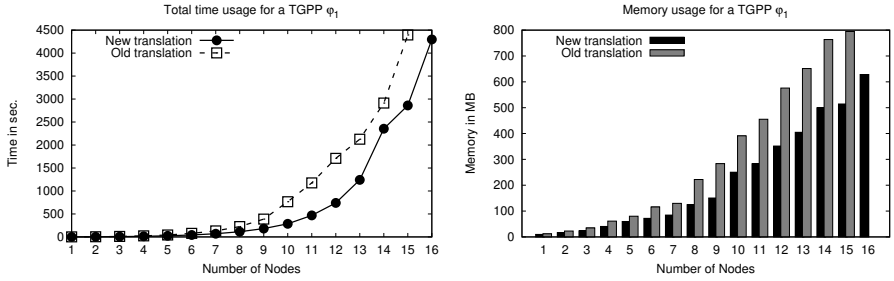

Fig. 3: φ_1: TGPP with n nodes.

From Fig. 3 one can observe that the new method is able to verify the formula φ_1 for TGPP with 17 nodes. The old method is able to verify the formula φ_1 for TGPP with 15 nodes. The memory usage for the old method is 1.50 times higher than for the new method for 15 nodes.

From Fig. 4 one can observe that the new method is able to verify the formula φ_2 for TGPP with 18 nodes. The old method is able to verify the formula φ_2 for TGPP with 15 nodes. The memory usage for the old method is 1.55 times higher than for the new method for 15 nodes.

From Fig. 5 one can observe that the new method is able to verify the formula φ_3 for TGPP with 17 nodes. The old method is able to verify the formula φ_3 for TGPP with 13 nodes. The memory usage for the old method is 2.21 times higher than for the new method for 13 nodes.

From Fig. 6 one can observe that the new method is able to verify the formula φ_4 for TGPP with 13 nodes. The old method is able to verify the formula φ_4 for TGPP

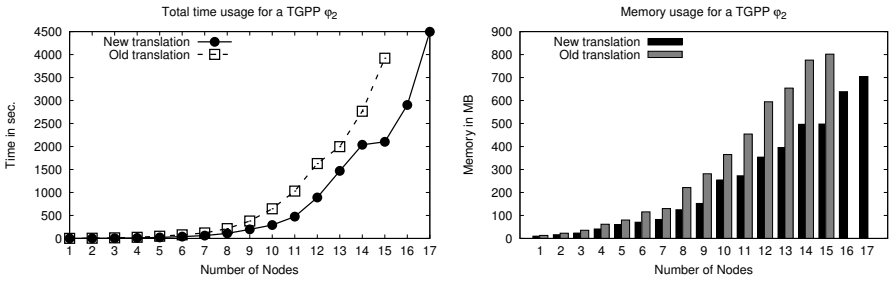

Fig. 4: φ_2: TGPP with n nodes.

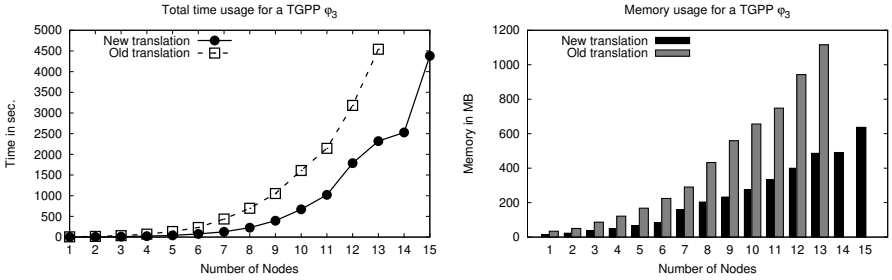

Fig. 5: φ_3: TGPP with n nodes.

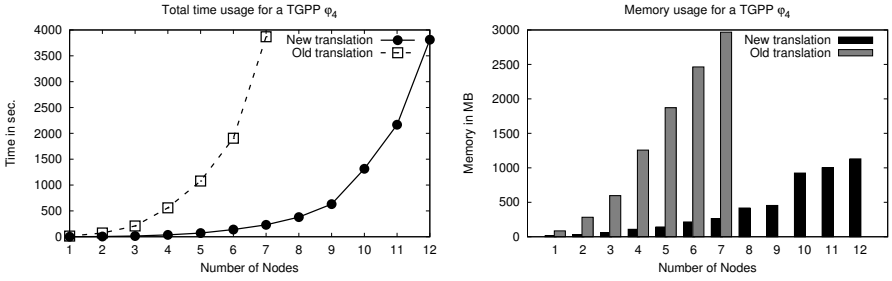

Fig. 6: φ_4: TGPP with n nodes.

with 7 nodes. The memory usage for the old method is 10.32 times higher than for the new method for 7 nodes.

From Fig. 8 one can observe that the new method is able to verify the formula φ_5 for TGPP with 13 nodes. The old method is able to verify the formula φ_5 for TGPP with 6 nodes. The memory usage for the old method is 21.39 times higher than for the new method for 6 nodes.

For the formula φ_4 the new method generates only 395775 variables and 1229009 clauses (Fig. 7) for 7 nodes. The time consumed by BMC to generate the set of clauses is equal 65.35 sec. The memory consumed by BMC to generate the set of clauses is

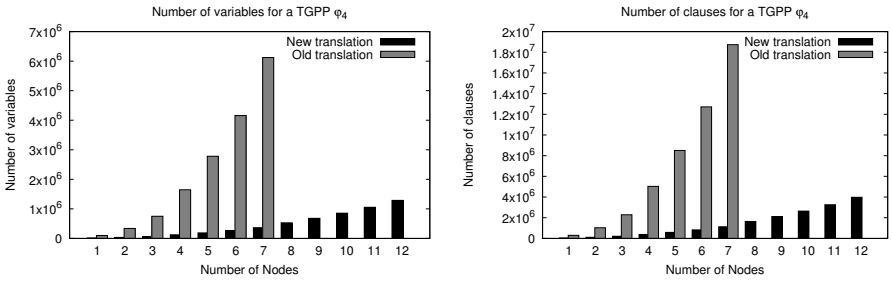

Fig. 7: Number of variables and clauses for φ_4 and TGPP with n nodes.

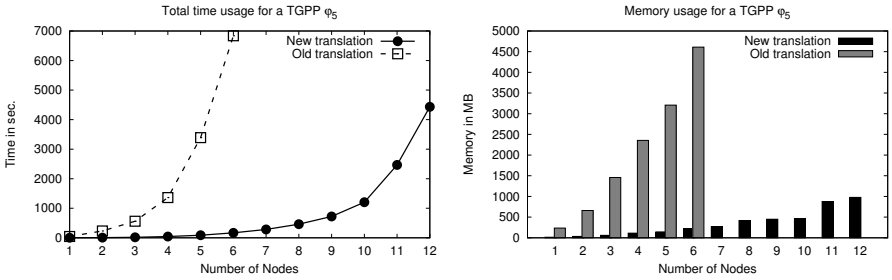

Fig. 8: φ_5: TGPP with n nodes.

equal 100.54 MB. The time and memory consumed by the state of the art SAT solver CryptoMiniSat5, respectively is equal to 112.59 sec. and 287.41 MB.

The old method generates 6122646 variables and 18739998 clauses. The time consumed by BMC to generate the set of clauses is equal 977.18 sec. The memory consumed by BMC to generate the set of clauses is equal 1505.05 MB. The time and memory consumed by the state of the art SAT solver CryptoMiniSat5, respectively is equal to 2892.31 sec. and 2967.82 MB.

The example above shows that our new method results in reducing the overall runtime and memory of BMC to construct a CNF formula, and of SAT solver to check satisfiability of this formula.

6 Conclusions

We have proposed, implemented, and experimentally evaluated SAT-based BMC method for soft real-time systems, which are modelled by discrete timed automata, and for properties expressible in MTL with the semantics over discrete timed automata. The method is based on a translation of the existential model checking for MTL to the existential model checking for LTL_q, and then on the translation of the existential model checking for LTL_q to the propositional satisfiability problem.

In the following table we compare the new BMC method with the old one.

Simple BMC+DTA&MTL	BMC+DTA&MTL[6]
no new clocks	a number of new clocks equal to the number of intervals in the given formula
no new transitions	exponential number of new transitions
only one path	a number of paths depending on the given formula and the length of the k-path
smaller number of propositional variables and clauses	substantially larger number of propositonal variables and clauses
better time and memory usage	worse time and memory usage

Table 1: The comparison of two methods

The experimental results show that our approach is much better than the approach based on translation to HLTL. The reason is that the new method needs only one new path, does not need any new clocks, and does not need any new transitions. The experiments confirm that the improvement in question leads to a reduction of the size of the CNF formulas submitted to the SAT solver, and therefore to a significant reduction both in the time and memory required by the SAT solver to return an answer. The paper presents preliminary experimental results only, but they show that the proposed verification method is quite efficient and worth exploring.

Therefore, in our future work we are going to define the SMT-based BMC encoding for DTA and for LTL_q and compare this encoding with the SAT-based encoding, and we would like to develop SAT-based BMC method for timed automata and properties expressible in TECTL.

References

1. R. Alur and D. Dill. A theory of Timed Automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
2. A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *Proc. of the 5th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'99)*, volume 1579 of *LNCS*, pages 193–207. Springer-Verlag, 1999.
3. Armin Biere, Alessandro Cimatti, Edmund M. Clarke, Ofer Strichman, and Yunshan Zhu. Bounded model checking. *Advances in Computers*, 58:117–148, 2003.
4. Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.
5. W. Penczek, B. Woźna, and A. Zbrzezny. Bounded model checking for the universal fragment of CTL. *Fundamenta Informaticae*, 51(1-2):135–156, 2002.
6. Bożena Woźna-Szcześniak and Andrzej Zbrzezny. A translation of the existential model checking problem from MITL to HLTL. *Fundamenta Informaticae*, 122(4):401–420, 2013.
7. Bożena Woźna-Szcześniak and Andrzej Zbrzezny. Checking MTL properties of discrete timed automata via bounded model checking. *Fundam. Inform.*, 135(4):553–568, 2014.
8. A. Zbrzezny. A new translation from ECTL* to SAT. *Fundamenta Informaticae*, 120(3-4):377–397, 2012.