sensors

Article

Bounded Model Checking for Metric Temporal Logic Properties of Timed Automata with Digital Clocks †

Agnieszka M. Zbrzezny [1] and Andrzej Zbrzezny [2,*]

[1] Faculty of Mathematics and Computer Science, University of Warmia and Mazury, Sloneczna 54, 10-710 Olsztyn, Poland
[2] Department of Mathematics and Computer Science, Jan Dlugosz University in Czestochowa, Armii Krajowej 13/15, 42-200 Czestochowa, Poland
* Correspondence: a.zbrzezny@ujd.edu.pl

† This paper is an extended version of our paper published in Zbrzezny, A.M. and Zbrzezny, A. Simple Bounded MTL Model Checking for Discrete Timed Automata (Extended abstract). In Proceedings of the 23th International Workshop on Concurrency, Specification and Programming (CS&P 2016), Rostock, Germany, 28–30 September 2016; Volume 1698, CEUR Workshop Proceedings, pp. 37–48.

Abstract: Metric temporal logic (MTL) is a popular real-time extension of linear temporal logic (LTL). This paper presents a new simple SAT-based bounded model-checking (SAT-BMC) method for MTL interpreted over discrete infinite timed models generated by discrete timed automata with digital clocks. We show a new translation of the existential part of MTL to the existential part of linear temporal logic with a new set of atomic propositions and present the details of the new translation. We compare the new method's advantages to the old method based on a translation of the hard reset LTL (HLTL). Our method does not need new clocks or new transitions. It uses only one path and requires a smaller number of propositional variables and clauses than the HLTL-based method. We also implemented the new method, and as a case study, we applied the technique to analyze several systems. We support the theoretical description with the experimental results demonstrating the method's efficiency.

Keywords: bounded model checking; timed automata; digital clocks; metric temporal logic; satisfiability problem

check for updates

Citation: Zbrzezny, A.M.; Zbrzezny, A. Bounded Model Checking for Metric Temporal Logic properties of Timed Automata with Digital Clocks. *Sensors* **2022**, *22*, 9552. https://doi.org/10.3390/s22239552

Academic Editor: Christian Haubelt

Received: 3 November 2022
Accepted: 2 December 2022
Published: 6 December 2022

1. Introduction

This paper is a full version of the extended abstract published in informal proceedings of the 25th International Workshop on Concurrency, Specification and Programming [1]. Improvements and extensions compared to that paper are listed in Appendix A.

There are many ways to check the model. Hardware- and software-based systems are increasingly used in safety-critical situations to control and connect physical systems, e.g., simple cardiac pacemakers or very complex space rockets. The complexity and criticality of systems also increase the need for effective verification techniques.

The specification language and the system model usually depend on the type of property we want to verify. Suppose we want to verify the discrete-time execution properties of a system. In that case, discrete-time logic may be the correct specification choice, and the system may best be represented as a discrete timed automaton.

Integrating specialized components is fundamental to many embedded engineering projects. The behavior of these components is often specified in informal timing diagrams that engineers interpret during interface hardware and software design [2–7]. The timed automata are one of the models that enable the formal modeling of those components.

Timed automata (TA) [8] are finite-state automata augmented with a finite set of variables called clocks. The clocks are used to measure the elapsed time. Timed automata are very convenient for modeling and reasoning about timed systems: they combine a powerful formalism with advanced expressiveness and efficient algorithmic and tool support. The timed automata formalism is applied to the analysis of software and asynchronous circuits [9] and real-time control programs [10].

The model-checking technique is widely used in sensor verification [2–7]. The sensor networks are modeled, e.g., by the network of timed automata, and their properties are specified in terms of temporal logic.

One of the most famous frameworks in the specification and verification of computer systems is temporal logic. There are many types of temporal logic to express the requirements of the systems: computation tree logic (CTL) [11], soft real-time CTL (RTCTL) [12], linear temporal logic (LTL) [13], and metric temporal logic (MTL) [14].

Linear temporal logic (LTL) allows expressing properties about each execution of a system, such as the fact that any occurrence of a problem eventually triggers the alarm. Metric temporal logic (MTL) extends LTL by constraining the temporal operators with time intervals. It was introduced by Koymans [14] in 1990 and has appeared as a real-time specification formalism. MTL has two main semantics: ”continuous” and ”pointwise” [15–18]. The pointwise semantics is based on timed words, the widespread interpretation for systems modeled as timed automata [8]. Both semantics have been extensively studied [15–18]. MTL allows expressing, for example, that any occurrence of a problem in a system will trigger the alarm within at most five units of time. Here, we consider MTL with pointwise semantics interpreted over linear discrete infinite digital-clock models [19] generated by timed automata with integer time.

Bounded model checking [20–22] (BMC) is one of the symbolic model-checking techniques 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 BMC problem to the satisfiability problem (SAT). The MTL satisfiability and model-checking problems are undecidable over interval-based semantics [23]. It has led to various restrictions being considered on MTL to recover decidability [24,25].

We provide a new, efficient method of the bounded model-checking technique for metric temporal logic properties of timed automata with digital clocks, which can be successfully used to verify sensor networks.

Developing new model-checking techniques for a network of automata is an essential research direction. It is due to the fact that timed automata are used to verify the modes of, often, life-critical time systems. Systems and their properties are becoming more and more complex. It results in developing model-checking methods that will work faster than the older methods. To be successfully applied, these methods should be faster than the old ones and use as little memory as possible. However, they should also be easy to implement and understand so that everything is evident at the design stage of such a method. For such a reason, in this paper, we developed a completely new and much faster method of bounded model checking than the one presented in [26].

The main contributions of the paper are as follows:

- Defining the 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 LTLq);
- Clarification of the steps of the new method;
- Proving the correctness of the above translation;
- Defining bounded semantics for LTLq;
- Defining the BMC algorithm;
- Implementing the new method;
- A detailed experimental evaluation of the old and the new methods on two earlier presented benchmarks: a timed generic pipeline paradigm (TGPP) and a timed train controller system (TTCS),

- Modeling a dining philosophers problem with time as the timed dining philosophers problem (TDPP);
- A detailed experimental evaluation of the old and the new methods on TDPP.

In this paper, we used the weakly monotonic semantics [27] for timed automata with digital clocks [19,27]. The main steps of our new method for MTL and TA with discrete time can be described as follows: first, the infinite timed model is reduced to a finite model. Next, the MTL formulae are translated to LTLq formulae [1], and eventually, since the interval modalities in MTL appear as literals in the LTLq formula, existential properties are reduced to a satisfiability problem (SAT). Our method's main advantages are that the translation from MTL to LTLq requires neither new clocks nor new transitions. Moreover, our BMC method needs only one path, whereas the BMC method from [26] needs many paths depending on a given formula φ. Thus, one may expect that our method is much more effective since the intuition is that an encoding that results in fewer variables and clauses is usually easier to solve.

We evaluate the BMC method using a timed generic pipeline paradigm (TGPP), a timed train controller system (TTCS), and the timed dining philosophers problem (TDPP), which we model by a network of discrete timed automata and compare with the corresponding method [26].

Related Work

Timed automata were introduced in the early 1990s by Alur and Dill [8] to model real-time systems. Timed automata cause the specification and verification of models of real-time systems to be easier. The two primary semantics are discussed in the literature: the discrete-time and dense-time semantics [8]. However, the dense-time semantics is more natural from a real-life point of view. It allows us to model real-time systems easily.

Our choice of time domain is $\mathbb{N}$, the set of natural numbers. In our method, the key property of the time domain is its discreteness, which implies that a finite amount of events can happen at different times in any interval of nonzero length. There are many methods for verifying real-time systems using discrete-time models [12,28–31]. Authors of [19] established that the timed reachability problem has the same answer, irrespective of the choice between $\mathbb{N}$ and $\mathbb{R}$ under certain restrictions.

The other formalisms for discrete time modeling apart from discrete timed automata were presented, such as durational transition graphs (DTG [32]) and embedded system modeling language (EMLAN [33]).

Discrete time models were also widely used for modeling systems' behaviors [34–38].

MTL has been widely discussed in the literature. Checking properties expressed in MTL in timed automata is still an actual research topic [39–44]. In [45], the authors took into account MTL over the $\mathbb{N}$. They also used the pointwise semantics over the $\mathbb{N}$ and considered two semantic variants: the non-strict and strict semantics. They devised two translations from MTL to LTL:

1. The time difference translation for strict semantics, where new propositional variables encode time differences between states (the time difference translation is similar to the method presented in [1]).
2. The gap translation for the strict semantics uses a new propositional variable, called gap, to encode the jumps between states. The gap is intended to be true in LTL states corresponding to unmapped time points in MTL models. The main idea for their translation is to map each timed state sequence into a state sequence. Both LTL translations are exponential in the size of the MTL input formula due to the binary encoding of the numbers in the intervals.

In [26], the authors investigated a SAT-based BMC method for MTL that is also interpreted over linear discrete infinite time models generated by discrete timed automata. They translated the existential model-checking problem for MTL into the existential model-checking problem for a variant of linear temporal logic (called HLTL). They also provided a SAT-based BMC technique for HLTL. The presented translation requires as many new clocks as there are intervals in a given formula. It also requires adding exponential resetting transitions and many paths that depend on a given MTL formula. The complexity of the satisfiability and model-checking problems for fragments of MTL concerning different semantic models were studied in [46] and in [15]. MTL expressiveness was extensively discussed in [17,47,48]. The BMC problem for MTL properties of timed automata with the dense time was discussed in [49]. However, experimental results have shown that this is not feasible.

Additionally, other types of logic were used for the specification of discrete-time systems, such as QsCTL [29], which extends CTL [11] with quantitative bounded temporal operators and is a variant of RTCTL [11], discrete-time CTL [33], and HyperLTL [50], which is a temporal logic for hyper-properties, which allows reasoning about multiple execution paths simultaneously.

2. Discrete Timed Automata and MTL

In this paper, we used the weakly monotonic semantics [27] for timed automata with digital clocks [19,27]. The paper [19] shows that bounded invariance MTL properties and bounded-response MTL properties are digitizable. That is why we consider timed automata with digital clocks.

The formalism of timed automata was defined in [8] by Alur and Dill for representing systems with real-time constraints. A timed automaton is a finite automaton which manipulates finitely many variables called clocks.

2.1. Discrete Timed Automata

Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of natural numbers, and $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$. We assume a finite set $\mathcal{X} = \{x_0, \ldots, x_{n-1}\}$ of variables, called *clocks*. Each clock is a variable ranging over $\mathbb{N}$. A *clock valuation* is a total function $v \colon \mathcal{X} \to \mathbb{N}$ that assigns to each clock x a non-negative integer value $v(x)$. The set of all the clock valuations is denoted by $\mathbb{N}^{\mathcal{X}}$. For $X \subseteq \mathcal{X}$, the valuation $v' = v[X := 0]$ is defined as $\forall x \in X, v'(x) = 0$ and $\forall x \in \mathcal{X} \setminus X, v'(x) = v(x)$. By δ, we denote a delay of $\delta > 0$ time units. For $\delta \in \mathbb{N}_+$, $v + \delta$ denotes the valuation v'' such that $\forall x \in \mathcal{X}, v''(x) = v(x) + \delta$; let $x \in \mathcal{X}$, $c \in \mathbb{N}$, and $\sim \in \{<, \leqslant, =, \geqslant, >\}$. The set $\mathcal{C}(\mathcal{X})$ of *clock constraints* over the set of clocks $\mathcal{X}$ is defined by the abstract grammar:

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

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

Definition 1. *A discrete timed automaton (DTA for short) is a tuple*

$$\mathcal{A} = (Act, Loc, \ell^0, \mathcal{X}, T, Inv, \mathcal{AP}, V), \textit{ where}$$

- *Act is a finite set of actions,*
- *Loc is a finite set of locations,*
- *$\ell^0 \in Loc$ is the initial location,*
- *$\mathcal{X}$ is a finite set of clocks,*
- *$T \subseteq Loc \times Act \times \mathcal{C}(\mathcal{X}) \times 2^{\mathcal{X}} \times Loc$ is a transition relation,*
- *$Inv \colon Loc \to \mathcal{C}(\mathcal{X})$ is a state invariant function,*
- *$\mathcal{AP}$ is a set of atomic propositions, and*
- *$V \colon Loc \to 2^{\mathcal{AP}}$ is a valuation function assigning to each location a set of atomic propositions true in this location.*

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

2.2. Product of a Network of Discrete Timed Automata

A network of discrete timed automata can be composed into a global (*product*) discrete timed automaton [51] in the following way. The transitions of the discrete timed automata that do not correspond to a shared action are interleaved, whereas the transitions labeled with a shared action, are synchronized.

Let $n \in \mathbb{N}$, $I = \{1, \dots, n\}$ be a non-empty and finite set of indices, $\{\mathcal{A}_i \mid i \in I\}$ be a family of discrete timed automata $\mathcal{A}_i = (Act_i, Loc_i, \ell_i^0, \mathcal{X}_i, T_i, Inv_i, \mathcal{AP}_i, V_i)$ such that $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset$ and $\mathcal{AP}_i \cap \mathcal{AP}_j = \emptyset$ for $i \neq j$. Moreover, let $I(\sigma) = \{i \in I \mid \sigma \in Act_i\}$. The *parallel compositioni* of the family $\{\mathcal{A}_i \mid i \in I\}$ of discrete timed automata is the discrete timed automaton $\mathcal{A} = (Act, Loc, \ell^0, \mathcal{X}, T, Inv, \mathcal{AP}, V)$ such that:

- $Act = \prod_{i \in I}(Act_i)$,
- $Loc = \prod_{i \in I} Loc_i$,
- $\ell^0 = (\ell_1^0, \dots, \ell_n^0)$,
- $\mathcal{X} = \bigcup_{i \in I} \mathcal{X}_i$,
- $Inv(l_1, \dots, l_n) = \bigwedge_{i=1}^{n} Inv_i(l_i)$,
- $\mathcal{AP} = \bigcup_{i \in I} \mathcal{AP}_i$,
- $V(\ell_1, \dots, \ell_n) = \bigcup_{i=1}^{n} V_i(\ell_i)$

and a transition is defined as follows:

$$((\ell_1, \dots, \ell_n), (\sigma_1, \dots, \sigma_n), \textstyle\bigwedge_{i \in I} \mathfrak{cc}_i, \bigcup_{i \in I} X_i, (\ell_1', \dots, \ell_m')) \in T$$
$$\text{iff } (\forall i \in I))(\ell_i, \sigma_i, \mathfrak{cc}_i, X_i, \ell_i') \in T_i) \text{ and } (\forall i \in I \setminus I(\sigma))\ell_i' = \ell_i.$$

2.3. Concrete Model

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

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

The concrete model for $\mathcal{A}$ is a tuple

$$\mathcal{M}_{\mathcal{A}} = (Q, q^0, \longrightarrow, \mathcal{V}), \text{ where}$$

- *$Q = Loc \times \mathbb{N}^{\mathcal{X}}$ is the set of the concrete states.*
- *$q^0 = (\ell^0, v^0)$ is the initial state.*
- *A valuation function $\mathcal{V}\colon Q \to 2^{\mathcal{AP}}$ is defined as $\mathcal{V}((\ell, v)) = V(\ell)$ for each state $(\ell, v) \in Q$ $\longrightarrow \subseteq Q \times (Act \cup \mathbb{N}_+) \times Q$ is a transition relation on Q defined by action and time transitions as follows.*
- *For $a \in Act$ and $\delta \in \mathbb{N}_+$:*
  1. *Action transition: $(\ell, v) \xrightarrow{a} (\ell', v')$ if there is a transition $\ell \xrightarrow{a, \mathfrak{cc}, X} \ell' \in T$ 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 $(\forall 0 < \delta' \leq \delta)$ $v + \delta \models Inv(\ell)$.*

Let us observe that for the considered set of clock constraints $\mathcal{C}(\mathcal{X})$, the condition of the time transition $(l,\, v) \xrightarrow{\delta} (l,\, v + \delta)$ can be replaced by a simpler one. Namely, $v \models Inv(\ell)$ and $v + \delta \models Inv(\ell)$.

A *path* ρ in $\mathcal{A}$ is an infinite sequence of concrete states $q_0, q_1, q_2, \dots$ such that for all $j \in \mathbb{N}$, $q_j \xrightarrow{\mu_j} q_{j+1}$ for some $\mu_j \in \mathbb{N}_+ \cup Act$. Such a definition of the path permits two consecutive actions to be performed one after the other, i.e., no time has to elapse between two consecutive actions. It means that we are dealing with the point-based *weakly monotonic* integer-time semantics.

From now on, for a path $\rho = q_0, q_1, q_2, \dots$, by $\rho(m)$, we denote the state q_m.

2.4. MTL Logic

MTL [14,15] (metric LTL) is the extension of LTL in which temporal operators are replaced by its time-constrained versions. MTL can express many time constraints. For example, we can express a system property: if a system is in the state q, then it will be in the state q' exactly 3 time units later.

2.4.1. Syntax

We begin with some preliminary definitions. Let $p \in \mathcal{AP}$ be an atomic proposition and $\mathcal{I}$ the set of all the intervals in $\mathbb{N}$ of the form $[a, b)$ or $[a, \infty)$, where $a, b \in \mathbb{N}$ and $a < b$, and let $\mathrm{I} \in \mathcal{I}$. Observe that we do not exclude one-element intervals since $[a, a]$ can be expressed as $[a, a\ + 1)$. The MTL logic in positive normal form is defined in the following way:

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

The operators $\mathbf{U}_{\mathrm{I}}$ and $\mathbf{G}_{\mathrm{I}}$ are called *bounded until* and *bounded always*, respectively, and they are read as "until in the interval I" and "always in the interval I". The operator $\mathbf{F}_{\mathrm{I}}$ is defined in the standard way: $\mathbf{F}_{\mathrm{I}}\alpha \overset{df}{=} \mathbf{true}\, \mathbf{U}_{\mathrm{I}}\, \alpha$.

2.4.2. Semantics

There are two possible semantics for metric temporal logic: the "pointwise" semantics and the "continuous" semantics [15]. In the pointwise approach, temporal assertions are interpreted only at time points where the action happens in the observed timed behavior of a system. In the continuous one, it is allowed to assert formulae at arbitrary time points between actions as well. In the presented method, we use the *pointwise* semantics.

Let $\mathcal{A}$ be a DTA, and $\mathcal{M}_{\mathcal{A}}$ the concrete model for $\mathcal{A}$. For a path $\rho = q_0, q_1, \ldots$, let $\Gamma_\rho(j) = \{i \in \mathbb{N} \mid i < j$ and for some $\delta_i \in \mathbb{N}, q_i \xrightarrow{\delta_i} q_{i+1}\}$, i.e $\Gamma_\rho(j)$ is a set of indices of time transitions. Now, we define a function $\zeta_\rho \colon \mathbb{N} \to \mathbb{N}$ such that, for all $j \geqslant 0$, $\zeta_\rho(j) = \sum_{i \in \Gamma_\rho(j)} \delta_i$. For all $j \geqslant 0$, the function $\zeta_\rho(j)$ returns the value of the global time (called "*duration*" in [15]).

Here and in what follows, we use the convention to omit the model from expressions with $\models$ for the sake of brevity.

Definition 3. *Let α and β be* MTL *formulae. The satisfaction relation $\models_{\mathrm{MTL}}$, which defines truth of an* MTL *formula in the concrete model $\mathcal{M}_{\mathcal{A}}$ along a path ρ starting at position $m \in \mathbb{N}$, is defined inductively:*

- $(\rho, m) \models_{\mathrm{MTL}} \mathbf{true}$,
- $(\rho, m) \not\models_{\mathrm{MTL}} \mathbf{false}$,
- $(\rho, m) \models_{\mathrm{MTL}} p$ *iff* $p \in \mathcal{V}(\rho(m))$,
- $(\rho, m) \models_{\mathrm{MTL}} \neg p$ *iff* $p \notin \mathcal{V}(\rho(m))$,
- $(\rho, m) \models_{\mathrm{MTL}} \alpha \wedge \beta$ *iff* $(\rho, m) \models_{\mathrm{MTL}} \alpha$ *and* $(\rho, m) \models_{\mathrm{MTL}} \beta$,
- $(\rho, m) \models_{\mathrm{MTL}} \alpha \vee \beta$ *iff* $(\rho, m) \models_{\mathrm{MTL}} \alpha$ *or* $(\rho, m) \models_{\mathrm{MTL}} \beta$,
- $(\rho, m) \models_{\mathrm{MTL}} \alpha \mathbf{U}_{\mathrm{I}} \beta$ *iff* $(\exists j \geq m)(\zeta_\rho(j) - \zeta_\rho(m) \in \mathrm{I}$ *and* $(\rho, m + j) \models_{\mathrm{MTL}} \beta$ *and* $(\forall m \leqslant j' < j)(\rho, m + j') \models_{\mathrm{MTL}} \alpha)$,
- $(\rho, m) \models_{\mathrm{MTL}} \mathbf{G}_{\mathrm{I}} \beta$ *iff* $(\forall j \geq m)(\zeta_\rho(j) - \zeta_\rho(m) \in \mathrm{I}$ *implies* $(\rho, m + j) \models_{\mathrm{MTL}} \beta)$.

For simplicity of notation, we write ρ instead of $(\rho, 0)$. Therefore, we shall write $\mathcal{M}_{\mathcal{A}}, \rho \models_{\mathrm{MTL}} \varphi$ for $\mathcal{M}_{\mathcal{A}}, (\rho, 0) \models_{\mathrm{MTL}} \varphi$. An MTL formula φ is *existentially valid* in the model $\mathcal{M}_{\mathcal{A}}$, which is denoted as $\mathcal{M}_{\mathcal{A}} \models_{\mathrm{MTL}} \mathbf{E}\varphi$, if and only if $\mathcal{M}_{\mathcal{A}}, \rho \models_{\mathrm{MTL}} \varphi$ for some path 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 the *existential model-checking problem*.

3. Bounded Model Checking

The verification method presented in this paper is based on the translation of MTL formula to LTLq formula. We extend a standard LTL logic by adding an extra set of propositional variables. We compare our new method with the corresponding method presented in [26].

3.1. The Translation

The set of all the clock valuations is infinite, which means that the model has an infinite set of states. We need to abstract the proposed model before we can apply the BMC technique.

3.1.1. Abstract Model

Let $\mathcal{A} = (Act, Loc, \ell^0, T, \mathcal{X}, Inv, \mathcal{AP}, V)$ be a discrete timed automaton with $\mathcal{X} = \{x_0, \ldots, x_{n-1}\}$. For each $j \in \{0, \ldots, n-1\}$, let c_j^{max} be the largest constant appearing in any clock constraint involving clock x_j and used in the state invariants and guards of $\mathcal{A}$. Two clock valuations v and v' in $\mathbb{N}^{\mathcal{X}}$ are equivalent, which is denoted by $v \simeq v'$, if and only if 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 easy to see that the relation $\simeq$ is an equivalence relation, which enables us to construct a finite abstract model.

To this end, we define the set of possible values of 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 $\mathcal{A} = (Act, Loc, \ell^0, \mathcal{X}, T, Inv, \mathcal{AP}, V)$ be a discrete timed automaton. The abstract model for $\mathcal{A}$ is a tuple*

$$\widehat{\mathcal{M}} = (\widehat{S}, s^0, \hookrightarrow, \widehat{\mathcal{V}}), \text{ where}$$

- $\widehat{S} = L \times (\mathbb{D}_0 \times \ldots \times \mathbb{D}_{n-1})$ *is the set of abstract states;*
- $s^0 = (\ell^0, \{0\}^n)$ *is the initial state;*
- $\widehat{\mathcal{V}} : \widehat{S} \to 2^{\mathcal{AP}}$ *is a valuation function such that for all $p \in \mathcal{AP}$, $p \in \widehat{\mathcal{V}}((\ell, v))$ if and only if $p \in V(\ell)$;*
- $\hookrightarrow \subseteq S \times Act' \times S$, *where $Act' = Act \cup \{\tau\}$ is a transition relation defined by the time and action transitions.*
  - *The time transition is defined as $(\ell, v) \overset{\tau}{\hookrightarrow} (\ell, v')$ if and only if $v \models Inv(\ell)$, $v' = succ(v)$ and $v' \models Inv(\ell)$.*
  - *The action transition is defined as follows: for any $a \in Act$, $(\ell, v) \overset{a}{\hookrightarrow} (\ell', v')$ if and only if there exists a transition $\ell \xrightarrow{a, \mathfrak{cc}, X} \ell' \in T$ such that $v \models \mathfrak{cc} \wedge Inv(\ell)$, $v' = v[X := 0]$ and $v' \models Inv(\ell')$.*

Definition 5. *A* path *in the abstract model $\widehat{\mathcal{M}}$ 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 $(s_j \overset{a}{\hookrightarrow} s_{j+1})$, for some action $a \in Act$.*

For a given path π, $\pi(j)$ denotes the j-th state s_j of the path π, $\pi[..j] = (\pi(0), \ldots, \pi(j))$ denotes the j-th prefix of the path π ending 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 $\pi[..j]$.

Definition 6. *Let $\mathcal{M}_{\mathcal{A}}$ be the concrete model for $\mathcal{A}$ and $\widehat{\mathcal{M}}$ the abstract model for $\mathcal{A}$. We say that a state $q = (l, v)$ in the concrete model $\mathcal{M}_{\mathcal{A}}$, and a state $s = (l', v')$ in the abstract model $\widehat{\mathcal{M}}$ are* equivalent, *which is denoted by $q \cong s$, if and only if $l = l'$ and $v \simeq v'$.*

It is well-known [52] that the relation $\cong$ is *weak-time-bisimulation equivalent* between the concrete model and the abstract model. The reason is that one can replace one δ-value time transition in the concrete model by δ individual transitions in the abstract model, whereas δ individual transitions in the abstract model can be replaced by one δ-value time transition in the concrete model.

3.1.2. MTL Semantics in the Abstract Model

Definition 7. *The satisfiability relation $\models^d_{\mathrm{MTL}}$, which defines the truth of an* MTL *formula in the abstract model $\widehat{\mathcal{M}}$ along the abstract path π with the starting point m at the depth $d \geqslant m$, is inductively defined as follows:*

- $(\pi, m) \models^d_{\mathrm{MTL}} \mathbf{true}$,
- $(\pi, m) \not\models^d_{\mathrm{MTL}} \mathbf{false}$,
- $(\pi, m) \models^d_{\mathrm{MTL}} p$ *iff* $p \in \widehat{\mathcal{V}}(\pi(d))$,
- $(\pi, m) \models^d_{\mathrm{MTL}} \neg p$ *iff* $p \notin \widehat{\mathcal{V}}(\pi(d))$,
- $(\pi, m) \models^d_{\mathrm{MTL}} \alpha \wedge \beta$ *iff* $(\pi, m) \models^d_{\mathrm{MTL}} \alpha$ *and* $(\pi, m) \models^d_{\mathrm{MTL}} \beta$,
- $(\pi, m) \models^d_{\mathrm{MTL}} \alpha \vee \beta$ *iff* $(\pi, m) \models^d_{\mathrm{MTL}} \alpha$ *or* $(\pi, m) \models^d_{\mathrm{MTL}} \beta$,
- $(\pi, m) \models^d_{\mathrm{MTL}} \alpha \mathbf{U}_{\mathrm{I}} \beta$ *iff* $(\exists j \geq d)(\zeta_\pi(j) - \zeta_\pi(d) \in \mathrm{I}$ *and* $(\pi, d) \models^j_{\mathrm{MTL}} \beta$ *and* $(\forall d \leqslant i < j)(\pi, d) \models^i_{\mathrm{MTL}} \alpha)$,
- $(\pi, m) \models^d_{\mathrm{MTL}} \mathbf{G}_{\mathrm{I}} \beta$ *iff* $(\forall j \geq d)(\zeta_\pi(j) - \zeta_\pi(d) \in \mathrm{I}$ *implies* $(\pi, d) \models^j_{\mathrm{MTL}} \beta)$.

In the above definition, m does not play itself a part in the satisfaction relation. However, this notation is helpful for Definition 8.

Theorem 1 (The equivalence of the MTL semantics in the concrete and abstract models)**.** *Let $\widehat{\mathcal{M}}$ be the abstract model for the discrete timed automaton $\mathcal{A}$ and $\mathcal{M}_{\mathcal{A}}$ the concrete model for $\mathcal{A}$. Then, for each* MTL *formula φ, the following equivalence holds: $\widehat{\mathcal{M}}, (\pi, m) \models^d_{\mathrm{MTL}} \varphi \iff \mathcal{M}_{\mathcal{A}}, (\rho, m) \models_{\mathrm{MTL}} \varphi$.*

Proof. The proof of Theorem 1 follows from the definition of the satisfiability relation and the weak-timed-bisimulation equivalence of the models $\mathcal{M}_{\mathcal{A}}$ and $\widehat{\mathcal{M}}$. □

3.1.3. LTLq Logic

Let $\mathcal{I}$ be the set of all intervals and $\mathcal{AP}_{\mathcal{I}} = \{q_{\mathrm{I}} \mid \mathrm{I} \in \mathcal{I}\}$ a set of the new propositional variables. An LTLq formula in the negation normal form is defined by the following grammar:

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

where $p \in \mathcal{AP}$ and $q_{\mathrm{I}} \in \mathcal{AP}_{\mathcal{I}}$.

Definition 8. *The satisfaction relation $\models^d$, which defines the truth of an* LTLq *formula in the abstract model $\widehat{\mathcal{M}}$ along the abstract path π at the position m, at depth $d \geqslant m$ is inductively defined as follows:*

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

An LTLq formula ψ is *existentially valid* in the abstract model $\widehat{\mathcal{M}}$, denoted as $\widehat{\mathcal{M}} \models \mathbf{E}\psi$, if and only if $\widehat{\mathcal{M}}, (\pi, 0) \models^0 \psi$ on some path π starting in the initial state of $\widehat{\mathcal{M}}$.

3.1.4. The Translation from MTL to LTLq

Two translations from MTL to LTL were described in [45]. However, in the first translation, the new propositional variables encode time differences between states, and in the second translation a new propositional variable called gap encodes the jumps between states. In the translation presented below, we use *global time* approach [1]. However, in [1] the strongly monotonic semantics was used.

Definition 9. *Let $p \in \mathcal{AP}$, and α, β a* MTL *formulae. The translation from* MTL *to* LTLq *is defined as a function* tr : MTL $\to$ LTLq *by the following equations:*

- $\mathrm{tr}(\mathbf{true}) = \mathbf{true}$,
- $\mathrm{tr}(\mathbf{false}) = \mathbf{false}$,
- $\mathrm{tr}(p) = p$,
- $\mathrm{tr}(\neg p) = \neg p$,
- $\mathrm{tr}(\alpha \wedge \beta) = \mathrm{tr}(\alpha) \wedge \mathrm{tr}(\beta)$,
- $\mathrm{tr}(\alpha \vee \beta) = \mathrm{tr}(\alpha) \vee \mathrm{tr}(\beta)$,
- $\mathrm{tr}(\alpha \mathbf{U}_{\mathrm{I}} \beta) = \mathrm{tr}(\alpha)\mathbf{U}(q_{\mathrm{I}} \wedge \mathrm{tr}(\beta))$, *and*
- $\mathrm{tr}(\mathbf{G}_{\mathrm{I}}\beta) = \mathbf{G}(\neg q_{\mathrm{I}} \vee \mathrm{tr}(\beta))$.

The translation of the $\mathbf{F}_I$ operator follows from its definition in terms of the $\mathbf{U}_I$ operator. Observe that the translation of literals, as well as logical connectives, is straightforward. The translation of the operator $\mathbf{U}_{\mathrm{I}}$ ensures that the formula β holds at some point in the interval I (it is expressed by the requirement $q_{\mathrm{I}} \wedge \mathrm{tr}(\beta)$) and α holds everywhere before β holds. Similarly, the translation of the $\mathbf{G}_{\mathrm{I}}$ operator ensures that β holds at every point in the interval I (it is expressed by the requirement $\neg q_{\mathrm{I}} \vee \mathrm{tr}(\beta)$).

The translation from EMTL to ELTLq is more straightforward than the one presented in [48], e.g., TPTL expressiveness is higher than LTLq. In our case, we do not need this extension of the logic to solve the given problem.

Theorem 2. *Let $\mathcal{A}$ be a discrete timed automaton, φ an* MTL *formula, and $\widehat{\mathcal{M}}$ the abstract model for $\mathcal{A}$. Then $\widehat{\mathcal{M}} \models_{\mathrm{MTL}} \mathbf{E}\,\varphi$ if, and only if $\widehat{\mathcal{M}} \models \mathbf{E}\,\mathrm{tr}(\varphi)$.*

4. Proof of the Theorem 2

A proof of the Theorem 2 follows directly from the Lemmas 1 and 2.

Lemma 1. *Let $\mathcal{A}$ be a discrete timed automaton, φ an* MTL *formula, $\widehat{\mathcal{M}}$ an abstract model for discrete timed automaton $\mathcal{A}$, and π an abstract path in the abstract model $\widehat{\mathcal{M}}$. If $(\pi, m) \models^d_{\mathrm{MTL}} \varphi$, then $(\pi, m) \models^d \mathrm{tr}(\varphi)$.*

Proof. We proceed by induction on the length of a given formula.

Assume that $(\pi, m) \models^d_{\mathrm{MTL}} \varphi$. Consider the following cases:

1. $\varphi \in \mathcal{AP}$. Because $\mathrm{tr}(\varphi) = \varphi$, it is obvious that $\mathrm{tr}(\varphi) \in \mathcal{AP}$. Therefore, $(\pi, m) \models^d_{\mathrm{MTL}} \varphi \iff \varphi \in \widehat{\mathcal{V}}(\pi(d)) \iff \mathrm{tr}(\varphi) \in \widehat{\mathcal{V}}(\pi(d)) \iff (\pi, m) \models^d \mathrm{tr}(\varphi)$.
2. $\varphi = \neg p$, where $p \in \mathcal{AP}$. Thus, $\mathrm{tr}(\varphi) = \varphi$. Therefore, $(\pi, m) \models^d_{\mathrm{MTL}} \varphi \iff \varphi \notin \mathcal{V}(\pi(d)) \iff (\pi, m) \models_{\mathrm{MTL}} \neg p \iff (\pi, m) \models^d \varphi \iff (\pi, m) \models^d \mathrm{tr}(\varphi)$.
3. $\varphi = \alpha \wedge \beta$. From the definition of the satisfiability relation (Definition 7) it follows that $(\pi, m) \models^d_{\mathrm{MTL}} \alpha$ and $(\pi, m) \models^d_{\mathrm{MTL}} \beta$. By inductive hypothesis, we obtain $(\pi, m) \models^d \mathrm{tr}(\alpha)$ and $(\pi, m) \models^d \mathrm{tr}(\beta)$. Therefore, $(\pi, m) \models^d \mathrm{tr}(\alpha) \wedge \mathrm{tr}(\beta)$, and hence $(\pi, m) \models^d \mathrm{tr}(\alpha \wedge \beta) \iff (\pi, m) \models^d \mathrm{tr}(\varphi)$.
4. $\varphi = \alpha \vee \beta$. From the definition of the satisfiability relation (Definition 7) it follows that $(\pi, m) \models^d_{\mathrm{MTL}} \alpha$ or $(\pi, m) \models^d_{\mathrm{MTL}} \beta$. By inductive hypothesis, we obtain that

$(\pi, m) \models^d \mathrm{tr}(\alpha)$ or $(\pi, m) \models^d \mathrm{tr}(\beta)$. Therefore, $(\pi, m) \models^d \mathrm{tr}(\alpha) \vee \mathrm{tr}(\beta)$, and hence $(\pi, m) \models^d \mathrm{tr}(\alpha \vee \beta) \iff (\pi, m) \models^d \mathrm{tr}(\varphi)$.

5. $\varphi = \alpha \mathbf{U}_\mathrm{I} \beta$. Assume that $(\pi, m) \models^d_{\mathrm{MTL}} \varphi$. From the definition of the satisfiability relation (Definition 7), it follows that $\zeta_\pi(j) - \zeta_\pi(d) \in \mathrm{I}$ and $((\pi, d) \models^j_{\mathrm{MTL}} \beta$ and $(\forall d \leqslant i < j)(\pi, d) \models^i_{\mathrm{MTL}} \alpha)$, for some $j \geq d$. By inductive hypothesis, we obtain $\zeta_\pi(j) - \zeta_\pi(d) \in \mathrm{I}$ and $(\pi, d) \models^j \mathrm{tr}(\beta)$, for some $j \geq d$ and $(\pi, d) \models^i \mathrm{tr}(\alpha)$, for all $d \leqslant i < j$. Therefore, $(\pi, d) \models^j q_\mathrm{I} \wedge \mathrm{tr}(\beta)$, for some $j \geq d$, and $(\pi, d) \models^i \mathrm{tr}(\alpha)$, for all $d \leqslant i < j$. Therefore, we conclude that $(\pi, m) \models^d \mathrm{tr}(\alpha \mathbf{U}_\mathrm{I} \beta)$.
6. $\varphi = \mathbf{G}_\mathrm{I} \beta$. Assume that $(\pi, m) \models^d_{\mathrm{MTL}} \varphi$. From the definition of the satisfiability relation (Definition 7), it follows that $(\forall j \geq d)(\zeta_\pi(j) - \zeta_\pi(d) \in \mathrm{I}$ implies $(\pi, d) \models^j_{\mathrm{MTL}} \beta)$, which means that $\zeta_\pi(j) - \zeta_\pi(d) \notin \mathrm{I} \vee (\pi, d) \models^j_{\mathrm{MTL}} \beta$, for all $j \geq d$. By inductive hypothesis, we obtain $\zeta_\pi(j) - \zeta_\pi(d) \notin \mathrm{I} \vee (\pi, d) \models^j \mathrm{tr}(\beta)$, for all $j \geq d$. Hence, $(\pi, d) \models^j \neg q_\mathrm{I} \wedge \mathrm{tr}(\beta)$, for all $j \geq d$. From the semantics of LTLq, it follows that $(\pi, m) \models^d \mathbf{G}(\neg q_\mathrm{I} \vee \mathrm{tr}(\beta))$. So, we can conclude that $(\pi, m) \models^d \mathrm{tr}(\mathbf{G}_\mathrm{I} \beta)$.

□

Lemma 2. *Let $\mathcal{A}$ be a discrete timed automaton, φ an* MTL *formula, $\widehat{\mathcal{M}}$ an abstract model for the discrete timed automaton $\mathcal{A}$, and π an abstract path in the abstract model $\widehat{\mathcal{M}}$. If $(\pi, m) \models^d \mathrm{tr}(\varphi)$, then $(\pi, m) \models^d_{\mathrm{MTL}} \varphi$.*

Proof. We proceed by induction on the length of a given formula.

1. $\varphi \in \mathcal{AP}$. Since $\varphi = \mathrm{tr}(\varphi)$, it follows that $\varphi \in \mathcal{AP}$. Therefore, $(\pi, m) \models^d \mathrm{tr}(\varphi) \iff \mathrm{tr}(\varphi) \in \widehat{\mathcal{V}}(\pi(d)) \iff \varphi \in \widehat{\mathcal{V}}(\pi(d)) \iff (\pi, m) \models^d_{\mathrm{MTL}} \varphi$.
2. $\varphi = \neg p$, where $p \in \mathcal{AP}$. Then $\varphi = \mathrm{tr}(\varphi)$. Therefore, $(\pi, m) \models^d \mathrm{tr}(\varphi) \iff \mathrm{tr}(\varphi) \notin \mathcal{V}(\pi(d)) \iff \varphi \notin \mathcal{V}(\pi(d)) \iff (\pi, m) \models^d_{\mathrm{MTL}} \neg p \iff (\pi, m) \models^d_{\mathrm{MTL}} \varphi$.
3. $\varphi = \alpha \wedge \beta$. Thus, $\mathrm{tr}(\varphi) = \mathrm{tr}(\alpha \wedge \beta) = \mathrm{tr}(\alpha) \wedge \mathrm{tr}(\beta)$. From the definition of the satisfiability relation (Definiton 8) it follows that $(\pi, m) \models^d \mathrm{tr}(\alpha)$ and $(\pi, m) \models^d \mathrm{tr}(\beta)$. By inductive hypothesis, we obtain $(\pi, m) \models^d_{\mathrm{MTL}} \alpha$ and $(\pi, m) \models^d_{\mathrm{MTL}} \beta$. Hence, $(\pi, m) \models^d_{\mathrm{MTL}} \alpha \wedge \beta$ and thus $(\pi, m) \models^d_{\mathrm{MTL}} \alpha \wedge \beta \iff (\pi, m) \models^d_{\mathrm{MTL}} \varphi$.
4. $\varphi = \alpha \vee \beta$. Then $\mathrm{tr}(\varphi) = \mathrm{tr}(\alpha \vee \beta) = \mathrm{tr}(\alpha) \vee \mathrm{tr}(\beta)$. From the definition of the satisfiability relation (Definition 8) it follows that $(\pi, m) \models^d \mathrm{tr}(\alpha)$ or $(\pi, m) \models^d \mathrm{tr}(\beta)$. By inductive hypothesis, we obtain $(\pi, m) \models^d_{\mathrm{MTL}} \mathrm{tr}(\alpha)$ or $(\pi, m) \models^d_{\mathrm{MTL}} \mathrm{tr}(\beta)$. Hence, $(\pi, m) \models^d \alpha \vee \beta$, and thus $(\pi, m) \models^d_{\mathrm{MTL}} \alpha \vee \beta \iff (\pi, m) \models^d_{\mathrm{MTL}} \varphi$.
5. $\varphi = \alpha \mathbf{U}_\mathrm{I} \beta$. Assume that $(\pi, m) \models^d_{\mathrm{MTL}} \varphi$. From the definition of the translation, it follows that $(\pi, m) \models^d \mathrm{tr}(\alpha) \mathbf{U}(q_\mathrm{I} \wedge \mathrm{tr}(\beta))$. From the definition of the satisfiability relation 8, it follows that $(\pi, d) \models^j q_\mathrm{I} \wedge \mathrm{tr}(\beta)$ and $(\forall d \leqslant i < j)(\pi, d) \models^i \mathrm{tr}(\alpha)$, for some $j \geqslant d$. Therefore, $(\pi, d) \models^j \zeta_\pi(j) - \zeta_\pi(d) \in \mathrm{I} \wedge (\pi, d) \models^j \mathrm{tr}(\beta)$ and $(\forall d \leqslant i < j)(\pi, d) \models^i \mathrm{tr}(\alpha)$, for some $j \geqslant d$. From the inductive hypothesis, we obtain $(\pi, d) \models^j_{\mathrm{MTL}} \zeta_\pi(j) - \zeta_\pi(d) \in \mathrm{I} \wedge (\pi, d) \models_{\mathrm{MTL}}{}^j \beta$ and $(\forall d \leqslant i < j)(\pi, d) \models^i_{\mathrm{MTL}} \alpha$, thus $(\pi, d) \models^j_{\mathrm{MTL}} \zeta_\pi(j) - \zeta_\pi(d) \in \mathrm{I} \wedge \beta$ and $(\forall d \leqslant i < j)(\pi, d) \models^i_{\mathrm{MTL}} \alpha$. Thus, we conclude that $(\pi, m) \models^d_{\mathrm{MTL}} \alpha \mathbf{U}_\mathrm{I} \beta$.
6. $\varphi = \mathbf{G}_\mathrm{I} \beta$. Assume that $(\pi, m) \models^d \varphi$. From the definition of the translation, it follows that $(\pi, m) \models^d \mathbf{G}(\neg q_\mathrm{I} \vee \mathrm{tr}(\beta))$. From the definition of the satisfiability relation $(\forall j \geqslant d)((\pi, d) \models^j \neg q_\mathrm{I}$ or $(\pi, d) \models^j \mathrm{tr}(\beta))$, which means $((\pi, d) \models^j \zeta_\pi(j) - \zeta_\pi(d) \notin \mathrm{I}$ or $(\pi, d) \models^j \mathrm{tr}(\beta))$, for all $j \geqslant d$. By inductive hypothesis, we obtain $(\forall j \geqslant d)((\pi, d) \models^j_{\mathrm{MTL}} \zeta_\pi(j) - \zeta_\pi(d) \notin \mathrm{I}$ or $(\pi, d) \models^j \beta)$, which is equivalent to $(\forall j \geqslant d)((\pi, d) \models^j_{\mathrm{MTL}} \zeta_\pi(j) - \zeta_\pi(d) \notin \mathrm{I} \vee \beta)$. Therefore, $(\pi, m) \models_{\mathrm{MTL}}{}^d \mathbf{G}_\mathrm{I} \beta$.

□

Proof of Theorem 2. ($\Longrightarrow$) Assume that $\widehat{\mathcal{M}} \models_{\mathrm{MTL}} \mathbf{E}\varphi$. Therefore, $\widehat{\mathcal{M}}, \pi \models^0_{\mathrm{MTL}} \varphi$, for some abstract path π in $\widehat{\mathcal{M}}$ such that $\pi(0) = s_0$. It means that $\widehat{\mathcal{M}}, (\pi, 0) \models^0_{\mathrm{MTL}} \varphi$. From Lemma 1, it follows that $(\pi, 0) \models^0 \mathrm{tr}(\varphi)$. Therefore, $(\pi, m) \models^0 \mathrm{tr}(\varphi)$, for $m = 0$. Thus $\widehat{\mathcal{M}} \models \mathbf{E}\mathrm{tr}(\varphi)$. ($\Longleftarrow$) Assume that $\widehat{\mathcal{M}} \models \mathbf{E}\mathrm{tr}(\varphi)$. Hence, $\widehat{\mathcal{M}}, \pi \models^0 \mathrm{tr}(\varphi)$, for some abstract path π in $\widehat{\mathcal{M}}$ such that $\pi(0) = s_0$. It means that $(\pi, 0) \models^0 \mathrm{tr}(\varphi)$. From Lemma 2, it follows that $\widehat{\mathcal{M}}, (\pi, 0) \models^0_{\mathrm{MTL}} \varphi$. Therefore, $\widehat{\mathcal{M}}, (\pi, m) \models^0_{\mathrm{MTL}} \varphi$, for $m = 0$. Thus $\widehat{\mathcal{M}} \models_{\mathrm{MTL}} \mathbf{E}\varphi$. $\square$

4.1. Bounded Semantics

To define the bounded semantics, we need to represent infinite paths in the abstract model using k-paths and loops [20,21].

Definition 10. *Let $\widehat{\mathcal{M}}$ be an abstract model, $k \in \mathbb{N}$ and $0 \leqslant l \leqslant k$. A k-path is a pair (π, l), which is also denoted as π_l, where π is a finite sequence of the abstract states $\pi = (s_0, \ldots, s_k)$ such that for each $0 \leqslant j < k$, either $(s_j \overset{\tau}{\hookrightarrow} s_{j+1})$ or $(s_j \overset{a}{\hookrightarrow} s_{j+1})$, for some $a \in Act$. Moreover, every action transition is preceded by at least one time transition. A k-path π_l is a loop, written as $\circlearrowleft(\pi_l)$ 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 $j \in \mathbb{N}$, $\widetilde{\pi}_l(l + j) = \widetilde{\pi}_l(k + j)$.

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

In the definition of bounded semantics for variables from $\mathcal{AP}_{\mathcal{I}}$, 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.

Definition 11 (Bounded semantics)**.** *Let $\widehat{\mathcal{M}}$ be the abstract model, π_l a k-path in $\widehat{\mathcal{M}}$, $0 \leqslant m \leqslant k$, and $0 \leqslant d \leqslant k$. The relation $\models^d_k$ is defined inductively as follows:*

$$\begin{array}{llll}
(\pi_l, m) \models^d_k & \mathbf{true}, & & \\
(\pi_l, m) \not\models^d_k & \mathbf{false}, & & \\
(\pi_l, m) \models^d_k & p & \mathit{iff} & p \in \mathcal{V}(\pi_l(d)), \\
(\pi_l, m) \models^d_k & \neg p & \mathit{iff} & p \notin \mathcal{V}(\pi_l(d)), \\
(\pi_l, m) \models^d_k & q_{\mathrm{I}} & \mathit{iff} & \begin{cases} \zeta_{\pi_l}(d) - \zeta_{\pi_l}(m) \in \mathrm{I}, & \mathit{if}\ \neg\circlearrowleft(\pi_l), \\ \zeta_{\pi_l}(d) - \zeta_{\pi_l}(m) \in \mathrm{I}, & \mathit{if}\ \circlearrowleft(\pi_l)\ \mathit{and}\ d \geqslant m, \\ \zeta_{\widetilde{\pi}_l}(d + k - l) - \zeta_{\widetilde{\pi}_l}(m) \in \mathrm{I}, & \mathit{if}\ \circlearrowleft(\pi_l)\ \mathit{and}\ d < m, \end{cases} \\
(\pi_l, m) \models^d_k & \neg q_{\mathrm{I}} & \mathit{iff} & (\pi_l, m) \not\models^d_k q_{\mathrm{I}}, \\
(\pi_l, m) \models^d_k & \alpha \wedge \beta & \mathit{iff} & (\pi_l, m) \models^d_k \alpha\ \mathit{and}\ (\pi_l, m) \models^d_k \beta, \\
(\pi_l, m) \models^d_k & \alpha \vee \beta & \mathit{iff} & (\pi_l, m) \models^d_k \alpha\ \mathit{or}\ (\pi_l, m) \models^d_k \beta, \\
(\pi_l, m) \models^d_k & \alpha \mathbf{U} \beta & \mathit{iff} & (\exists_{d \leqslant j \leqslant k})\big((\pi_l, d) \models^j_k \beta\ \mathit{and}\ (\forall_{d \leqslant i < j})(\pi_l, d) \models^i_k \alpha\big) \\
 & & & \mathit{or}\ \big(\circlearrowleft(\pi_l)\ \mathit{and}\ (\exists_{l < j < d})(\pi_l, d) \models^j_k \beta\ \mathit{and} \\
 & & & (\forall_{l < i < k})\ (\pi_l, d) \models^j_k \alpha\ \mathit{and}\ (\forall_{d \leqslant i \leqslant k})\ (\pi_l, d) \models^i_k \alpha\big), \\
(\pi_l, m) \models^d_k & \mathbf{G}\beta & \mathit{iff} & \circlearrowleft(\pi_l)\ \mathit{and}\ (\forall_{j \leqslant k}) j \geqslant \min(d, l)\ \mathit{implies}\ (\pi_l, d) \models^j_k \beta.
\end{array}$$

The proof of Lemma 3 below is based on induction on the length of the given formula. It is analogous to the proof of Lemma 7 from the paper [20].

Lemma 3. *Let $\mathcal{A}$ be a discrete timed automaton, φ an* LTL*q formula, and $\widehat{\mathcal{M}}$ the abstract model for the automaton $\mathcal{A}$. For each* LTL*q formula φ, each $k-$path π_l in $\widehat{\mathcal{M}}$, each $0 \leq m \leq k$ and* *each* $0 \leq d \leq k$, *if* $(\pi_l, m) \models_k^d \varphi$, *there exists a path* π' *such that* $\pi'[..k] = \pi_l$ *and* $m \leq d$ *and* $(\pi', m) \models^d \varphi$ *or* $m > d$ *and* $(\pi', m) \models^{d+k-l} \varphi$.

The proof of the Lemma 4 below is based on the well-known fact that if the LTL formula is true on some infinite path, it is also true on an infinite path of the form uv^ω, where u and v are finite sequences of states [20].

Lemma 4. *Let* $\mathcal{A}$ *be a discrete timed automaton,* φ *an* LTL*q formula,* $\widehat{\mathcal{M}}$ *the abstract model for the automaton* $\mathcal{A}$, π *a path in the abstract model, and* $k \geq 0$. *For each* LTL*q formula* φ, *each* $0 \leq m \leq k$ *and each* $0 \leq d \leq k$, *if* $(\pi, m) \models^d \varphi$, *there exists a* $k-$*path* π_l *such that* $(\pi_l, m) \models_k^d \varphi$.

An LTLq formula φ existentially k-holds in the model $\widehat{\mathcal{M}}$, written as $\widehat{\mathcal{M}} \models_k \mathbf{E}\,\varphi$, if and only if $\widehat{\mathcal{M}}, (\pi_l, 0) \models_k^0 \varphi$ for some $k-$path π_l starting at the initial state.

Theorem 3 shows that for some specific bound, bounded and unbounded semantics are equivalent. The proof of Theorem 3 follows directly from Lemmas 3 and 4.

Theorem 3. *Let* $\widehat{\mathcal{M}}$ *be the abstract model and* φ *an* LTL*q formula. Then,* $\widehat{\mathcal{M}} \models \mathbf{E}\,\varphi$ *if and only if there exists a* $k \geqslant 0$ *such that* $\widehat{\mathcal{M}} \models_k \mathbf{E}\,\varphi$.

Example 1. *Figure 1 shows an automaton modeling a simple light switch. It consists of two locations, a and b. When the action on is performed, the clock* x_0 *is reset. The automaton can stay in the location b until the valuation of the clock* x_0 *is less or equal to 6. The transition from location b to location a (action off) can be performed when the valuation of the clock* x_0 *is greater than 3.*

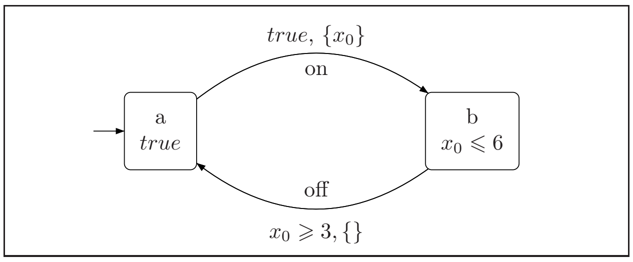
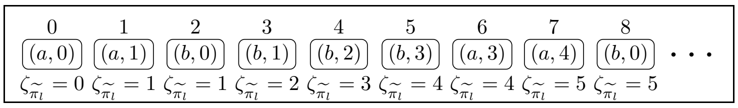


Figure 1. The simple light switch.

Figure 2 shows an abstract path in the abstract model for the simple light switch. Under the states, we show the global time at the given position.



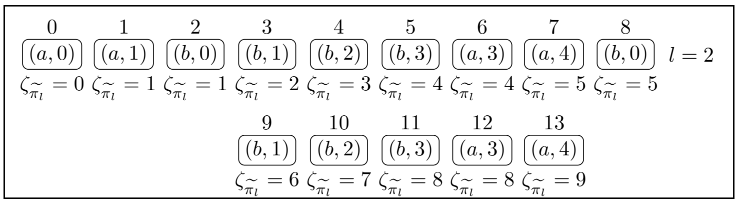
0
1
2
3
4
5
6
7
8
$(a,0)$
$(a,1)$
$(b,0)$
$(b,1)$
$(b,2)$
$(b,3)$
$(a,3)$
$(a,4)$
$(b,0)$
$\cdots$
$\zeta_{\tilde{\pi}_l} = 0$
$\zeta_{\tilde{\pi}_l} = 1$
$\zeta_{\tilde{\pi}_l} = 1$
$\zeta_{\tilde{\pi}_l} = 2$
$\zeta_{\tilde{\pi}_l} = 3$
$\zeta_{\tilde{\pi}_l} = 4$
$\zeta_{\tilde{\pi}_l} = 4$
$\zeta_{\tilde{\pi}_l} = 5$
$\zeta_{\tilde{\pi}_l} = 5$

Figure 2. An example of the path.

Example 2. *Let us check the satisfiability of formula* q_I *for the case when* π_l *is not a loop. Let* $\mathrm{I} = [0, 5)$, $k = 2$, $m = 3$ *and* $d = 5$.

$$(\pi_2, 3) \models_8^5 q_\mathrm{I} \iff \zeta_\pi(5) - \zeta_\pi(3) \in \mathrm{I} \iff 2 \in \mathrm{I}.$$

Figure 3 shows an example of the $k-$*path, which is a loop. Note that for* $l = 2$ *and* $k = 8$, $u = (\pi(0), \pi(1), \pi(2))$, *and* $v = (\pi(3), \pi(4), \pi(5), \pi(6), \pi(7), \pi(8))$. *Under the states, we show the global time in the given state.*



0 1 2 3 4 5 6 7 8
$(a,0)$ $(a,1)$ $(b,0)$ $(b,1)$ $(b,2)$ $(b,3)$ $(a,3)$ $(a,4)$ $(b,0)$ $l=2$
$\zeta_{\widetilde{\pi}_l}=0$ $\zeta_{\widetilde{\pi}_l}=1$ $\zeta_{\widetilde{\pi}_l}=1$ $\zeta_{\widetilde{\pi}_l}=2$ $\zeta_{\widetilde{\pi}_l}=3$ $\zeta_{\widetilde{\pi}_l}=4$ $\zeta_{\widetilde{\pi}_l}=4$ $\zeta_{\widetilde{\pi}_l}=5$ $\zeta_{\widetilde{\pi}_l}=5$
9 10 11 12 13
$(b,1)$ $(b,2)$ $(b,3)$ $(a,3)$ $(a,4)$
$\zeta_{\widetilde{\pi}_l}=6$ $\zeta_{\widetilde{\pi}_l}=7$ $\zeta_{\widetilde{\pi}_l}=8$ $\zeta_{\widetilde{\pi}_l}=8$ $\zeta_{\widetilde{\pi}_l}=9$

Figure 3. An example of the k-path, which is a loop.

Example 3. *Let us check the satisfiability of formula q_{I} for the case when π_l is a loop and $d \geq m$. Let $\mathrm{I} = [0,5)$, $k = 8$, $l = 2$, $m = 3$ and $d = 5$.*

$$(\pi_2, 3) \models^5_8 q_{\mathrm{I}} \iff \zeta_{\widetilde{\pi}_l}(5) - \zeta_{\widetilde{\pi}_l}(3) \in \mathrm{I} \iff 2 \in \mathrm{I}.$$

Example 4. *Let us check the satisfiability of the formula q_{I} for the case when π_l is a loop and $d < m$. Let $\mathrm{I} = [0,5)$, $k = 8$, $l = 2$, $m = 8$ and $d = 5$.*

$$(\pi_2, 8) \models^5_8 q_{\mathrm{I}} \iff \zeta_{\widetilde{\pi}_l}(5+8-2) - \zeta_{\widetilde{\pi}_l}(8) \in \mathrm{I} \iff \zeta_{\widetilde{\pi}_l}(11) - \zeta_{\widetilde{\pi}_l}(8) \in \mathrm{I} \iff 4 \in \mathrm{I}.$$

4.2. Translation to SAT

The last step of our method is the standard one ([26,53]). It consists in encoding the transition relation of $\widehat{\mathcal{M}}$ and the LTLq formula $\mathrm{tr}(\varphi)$. The only novelty lies in the encoding of the finite prefix of the sequence $(\zeta_{\widetilde{\pi}_l}(0), \zeta_{\widetilde{\pi}_l}(1), \ldots)$.

Let $\widehat{\mathcal{M}}$ be the abstract model for the automaton $\mathcal{A}$, $\mathrm{tr}(\varphi)$ be the LTLq formula, and $k \geq 0$ a bound. The formula $[\mathrm{tr}(\varphi)]_k$ encodes a bounded semantics of the LTLq formula $\mathrm{tr}(\varphi)$. It is defined over the same set of the propositional variables as the propositional formula $[\widehat{\mathcal{M}}^{\mathrm{tr}(\varphi),s^0}]_k$.

The definition of the formula $[\widehat{\mathcal{M}}^{\mathrm{tr}(\varphi),s^0}]_k$ assumes that, states and actions in the abstract model $\widehat{\mathcal{M}}$, and passage of time are encoded symbolically. This is possible if the set of states and the set of actions are finite. Formally, each symbolic abstract state $\widehat{s} \in \widehat{S}$ is represented by a vector, $\overline{\mathbf{w}} = ((w_1, v_1), \ldots, (w_r, v_r))$ of propositional variables, where the length r depends on the number of states in the abstract model. This vector is called *a symbolic state*. Each action $a \in Act$ is represented by a vector $\overline{\mathbf{a}} = (a_1, \ldots, a_t)$ of propositional variables, where the length t depends on number of local actions in $\mathcal{A}$. It is called *a symbolic action*.

A pair consisting of a sequence of the symbolic transitions and a symbolic number is called *a symbolic k-path*. Let $\boldsymbol{\pi}$ be a pair which represents a symbolic k-path: $((\overline{\mathbf{w}}_0, \overline{\mathbf{w}}_1, \ldots, \overline{\mathbf{w}}_{k-1}, \overline{\mathbf{w}}_k), \overline{\mathbf{u}})$, where $\overline{\mathbf{w}}_i$ is a symbolic state, for $0 \leq i \leq k$, and $\overline{\mathbf{u}}$ is a symbolic number, which is a vector $\overline{\mathbf{u}} = (u_1, \ldots, u_y)$ of propositional variables with $y = max(1, \lceil \log_2(k+1) \rceil)$. Moreover, let $\overline{\mathbf{a}}_i$, for $0 < i \leq k$, be a symbolic action.

Let $\overline{\mathbf{w}}$ and $\overline{\mathbf{w}}'$ be two different symbolic states, $\overline{\mathbf{a}}$ a symbolic action and $\overline{\mathbf{u}}$ a symbolic number. To define the formula $[\widehat{\mathcal{M}}^{\mathrm{tr}(\varphi),s^0}]_k$, we use the following auxiliary propositional formulae: $I_{\widehat{s}}(\overline{\mathbf{w}})$ encodes a state $\widehat{s}$ in the abstract model $\widehat{\mathcal{M}}$, $H(\overline{\mathbf{w}}, \overline{\mathbf{w}}')$ encodes the equality of two global states, $\mathcal{T}_{Act}(\overline{\mathbf{w}}, \overline{\mathbf{a}}, \overline{\mathbf{w}}')$ encodes an action transition in $\widehat{\mathcal{M}}$, $\mathcal{T}_\tau(\overline{\mathbf{w}}, \tau, \overline{\mathbf{w}}')$ encodes a time transition in $\widehat{\mathcal{M}}$, $\mathcal{B}^{\sim}_j(\overline{\mathbf{u}})$, for $\sim \in \{<, \leq, =, >, \geq\}$ encodes the relation $\sim$ between j and $\overline{\mathbf{u}}$, and $\mathcal{L}^l_{k,m}(\boldsymbol{\pi})$ encodes the existence of a loop for path $\boldsymbol{\pi}$ at position l.

The propositional formula $[\widehat{\mathcal{M}}^{\mathrm{tr}(\varphi),s^0}]_k$, encodes the unfolding of the transition relation of the abstract model $\widehat{\mathcal{M}}$ to the depth k in the following way:

$$\begin{aligned}[\widehat{\mathcal{M}}^{\mathrm{tr}(\varphi),s^0}]_k := \textstyle\bigvee_{s \in s^0} I_s(\overline{\mathbf{w}}_0) \wedge \bigvee_{l=0}^{k} \mathcal{B}^{=}_l(\overline{\mathbf{u}})\\ \wedge \Big(\textstyle\bigwedge_{j=0}^{k-1} \big(\mathcal{T}_\tau(\overline{\mathbf{w}}_j, \tau, \overline{\mathbf{w}}_{j+1}) \vee \mathcal{T}_{Act}(\overline{\mathbf{w}}_j, \overline{\mathbf{a}}_j, \overline{\mathbf{w}}_{j+1})\big)\Big),\end{aligned}$$

where $\overline{\mathbf{w}}_i$, $\overline{\mathbf{a}}_i$ and $\overline{\mathbf{u}}$ are, respectively, symbolic states, symbolic actions and the symbolic number for $0 \leq i \leq k$.

The next step of the method is the translation of the LTLq formula $\mathrm{tr}(\varphi)$ into the propositional formula $[\mathrm{tr}(\varphi)]_k := [\mathrm{tr}(\varphi)]^0_{[k,0]}$. To translate the formula $\mathrm{tr}(\varphi)$ to SAT problem, we use the auxiliary propositional formulae defined in [53] and the propositional formula $Gt^{d,m}_{\mathrm{I}}(\boldsymbol{\pi})$. The formula $Gt^{d,m}_{\mathrm{I}}(\boldsymbol{\pi})$ encodes the condition that says that the difference of the symbolic global time at the depth d and in the starting point m on the symbolic path $\boldsymbol{\pi}$ belongs to the interval I.

Definition 12 (The translation from $\mathbf{ELTL_q}$ to SAT)**.** *Let $\widehat{\mathcal{M}}$ be the abstract model,* $\mathrm{tr}(\varphi)$ *an* LTLq *formula, and $k \geq 0$ a bound. The translation of the formula* $\mathrm{tr}(\varphi)$ *on the path starting at point m at the depth d is defined inductively:*

$$
\begin{aligned}
[\mathbf{true}]^d_{[k,m]} &:= \mathbf{true},\\
[\mathbf{false}]^d_{[k,m]} &:= \mathbf{false},\\
[p]^d_{[k,m]} &:= p(\overline{\mathbf{w}}_d),\\
[\neg p]^d_{[k,m]} &:= \neg p(\overline{\mathbf{w}}_d),\\
[q_{\mathrm{I}}]^d_{[k,m]} &:= \begin{cases} \bigvee_{l=0}^{k-1}\left(Gt^{d,m}_{\mathrm{I}}(\boldsymbol{\pi}) \wedge \neg H(\overline{\mathbf{w}}_k,\overline{\mathbf{w}}_l)\right) \vee \\ \quad \bigvee_{l=0}^{k-1}\left(Gt^{d,m}_{\mathrm{I}}(\boldsymbol{\pi}) \wedge H(\overline{\mathbf{w}}_k,\overline{\mathbf{w}}_l)\right), & \text{if } d \geq m\\ \bigvee_{l=0}^{k-1}\left(Gt^{d,m}_{\mathrm{I}}(\boldsymbol{\pi}) \wedge \neg H(\overline{\mathbf{w}}_k,\overline{\mathbf{w}}_l)\right) \vee \\ \quad \bigvee_{l=0}^{k-1}\left(Gt^{d+k-l,m}_{\mathrm{I}}(\boldsymbol{\pi}) \wedge H(\overline{\mathbf{w}}_k,\overline{\mathbf{w}}_l)\right), & \text{if } d < m \end{cases}\\
[\neg q_{\mathrm{I}}]^d_{[k,m]} &:= \neg[q_{\mathrm{I}}]^d_{[k,m]},\\
[\alpha \wedge \beta]^d_{[k,m]} &:= [\alpha]^d_{[k,m]} \wedge [\beta]^d_{[k,m]},\\
[\alpha \vee \beta]^d_{[k,m]} &:= [\alpha]^d_{[k,m]} \vee [\beta]^d_{[k,m]},\\
[\alpha\mathbf{U}\beta]^d_{[k,m]} &:= \bigvee_{j=d}^{k}\left([\beta]^j_{[k,m]} \wedge \bigwedge_{i=d}^{j-1}[\alpha]^i_{[k,m]}\right) \vee \left(\bigvee_{l=0}^{d-1}\left(\mathcal{L}^l_{k,m}(\boldsymbol{\pi})\right) \wedge \bigvee_{j=0}^{d-1}\left(\mathcal{B}^{>}_j(\overline{\mathbf{u}})\right.\right.\\
&\qquad \left.\left. \wedge [\beta]^j_{[k,m]} \wedge \bigvee_{i=0}^{j-1}(\mathcal{B}^{>}_i(\overline{\mathbf{u}}) \rightarrow [\alpha]^i_{[k,m]}) \wedge \bigwedge_{i=d}^{k}[\alpha]^i_{[k,m]}\right)\right),
\end{aligned}
$$

$$
[\mathbf{G}\alpha]^d_{[k,m]} := \bigvee_{l=0}^{k-1}\left(\mathcal{L}^l_{k,m}(\boldsymbol{\pi})\right) \wedge \bigwedge_{j=0}^{d-1}\left(\mathcal{B}^{\geq}_j(\overline{\mathbf{u}}) \rightarrow [\alpha]^j_{[k,m]}\right) \wedge \bigwedge_{j=d}^{k}[\alpha]^j_{[k,m]}.
$$

Theorem 4. *Let $\widehat{\mathcal{M}}$ be the abstract model. Then for every $k \in \mathbb{N}$, at the depth $d \leq k$, $\widehat{\mathcal{M}} \models^d_k$* $\mathbf{E}\mathrm{tr}(\varphi)$ *if, and only if, the propositional formula $[\widehat{\mathcal{M}}^{\mathrm{tr}(\varphi),s^0}]_k \wedge [\mathrm{tr}(\varphi)]_k$ is satisfiable.*

The proof of the above theorem is analogous to the proofs presented in [26,53].

5. Experimental Results

In this section, we experimentally evaluate the performance of our new translation (We performed our experimental results on a computer equipped with I7-3770 processor, 32 GB of RAM, and the operating system Arch Linux. All the benchmarks together with instructions on how to reproduce our experimental results can be found at the web page https://tinyurl.com/satbmc4dtta-emtl, accessed on 3 November 2022). Our SAT-based BMC algorithm is implemented as a standalone program written in the programming language C++. We compared the new method with the corresponding one from [26]. For both methods, we used the state-of-the-art Kissat SAT solver [54]. We conducted the experiments using the slightly modified TGPP [26], the TTCS [26], and TDPP, and we compared our result with the results generated using the implementation from [26].

5.1. Timed Dining Philosophers

As the first benchmark, we used the well-known dining philosophers problem [55], and we extended it using clocks. The system consists of n discrete timed automata, each of which models a philosopher, together with n automata, each of which models a fork, together with one automaton which models the lackey. The latter automaton is used to coordinate the philosophers' access to the dining room. In fact, this automaton ensures that no deadlock is possible. The global system is obtained as the parallel composition of the components, which are shown in Figure 4.

We assume that one unit of time represents 30 min. A philosopher has to think at least 30 min (1 time unit, $x_j \geq T_2$) and at most 2 h and 30 min (5 time units $x_j \leq T_1$). He also has to eat for, at most, one hour (2 time units, $x_j \leq E_1$), but he also can finish eating earlier ($x_j \geq E_2$).

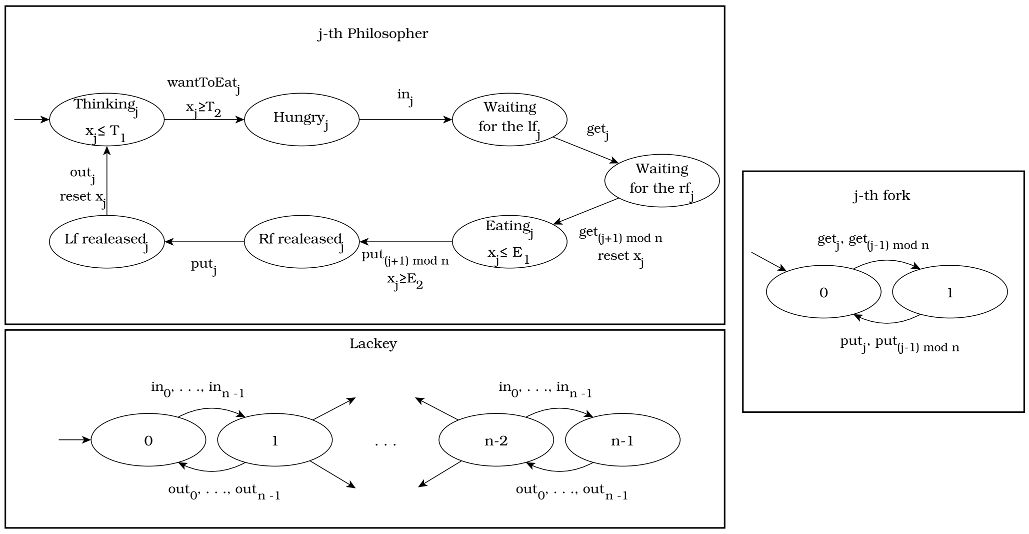


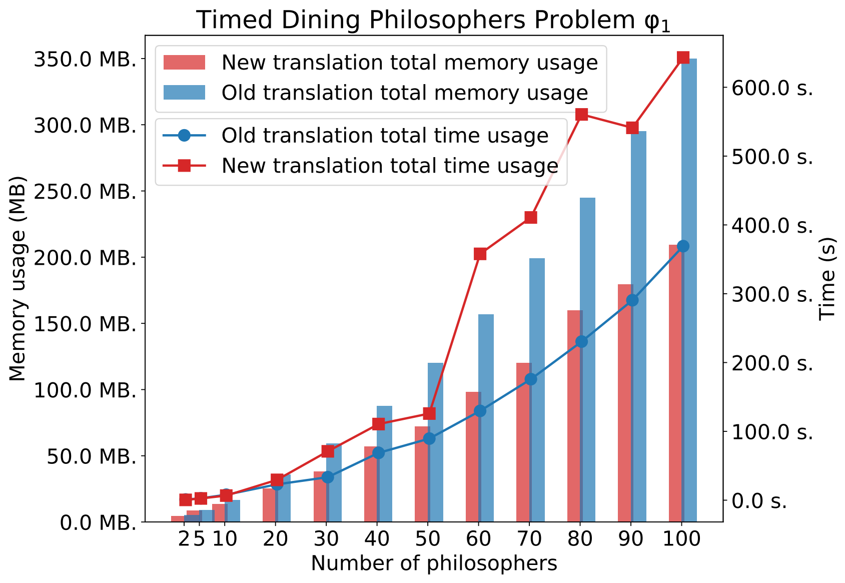
Figure 4. The TDPP system.

Let us consider the following formulae:

- $\varphi_1 = \mathbf{F}_{[0,T_2+E_2+1)}(\bigvee_{j=1}^{n} Lf\ realeased_j)$. At least one philosopher will eventually eat and put down both forks.
- $\varphi_2 = \mathbf{F}_{[0,T_2+1)}(\bigwedge_{j=\{1,3,5,\ldots\}}^{n-1} Eating_j)$. Eventually, every second philosopher (starting with the first one) eats.
- $\varphi_3 = \mathbf{G}(\mathbf{F}_{[0,T_2+1)}(\bigvee_{j=1}^{n} Lf\ realeased_j)$. Every second philosopher (starting with the first one) always eats in the end.

All these formulae are existentially valid in the model of TDPP.

Figure 5 shows experimental results for φ_1 and φ_2. For the simple *eventually* formula φ_1 we can observe that time usage for the method based on the old translation is better than for the method based on the new one. However there is a noticeable difference in memory usage. In this case, the new method is better. For the formulae φ_2 and φ_3, we can see the advantages of the new method.



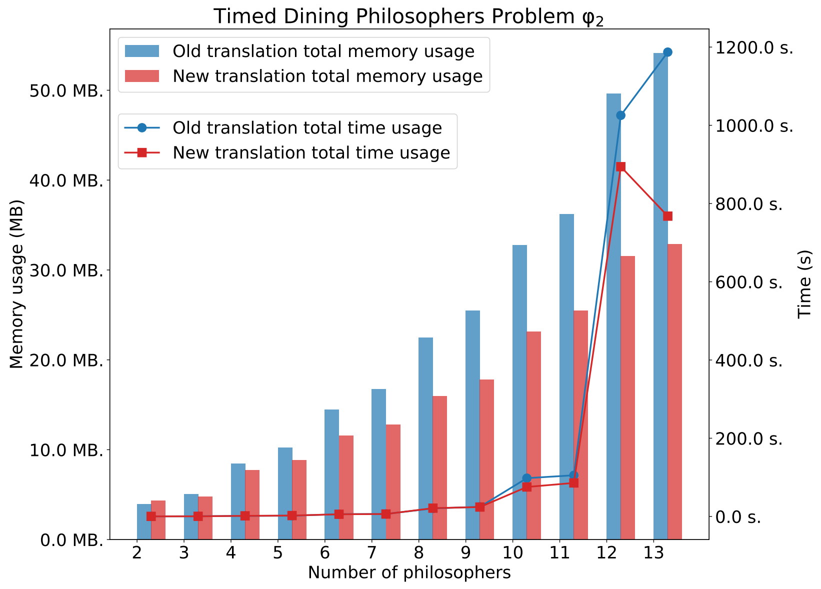


Figure 5. TDPP with n philosophers: φ_1 and φ_2.

Figure 6 shows experimental results for φ_3.

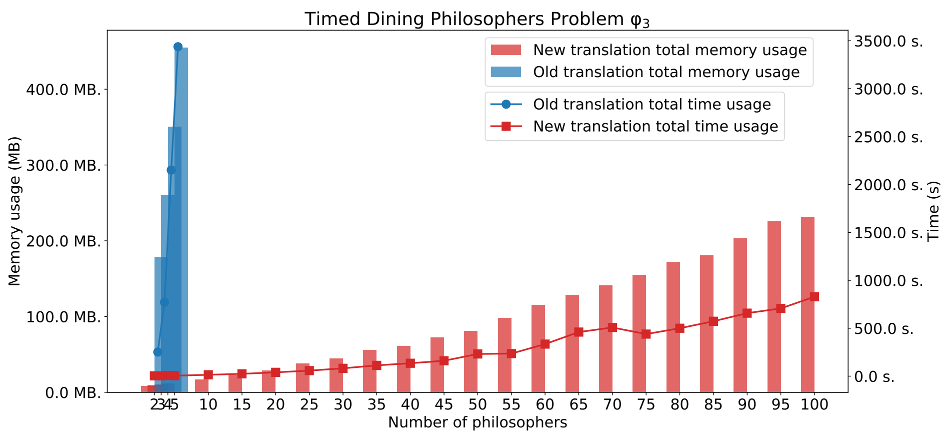


Figure 6. The TDPP with n philosophers: φ_3.

Figure 7 shows generated clauses and variables for φ_1 and φ_3.

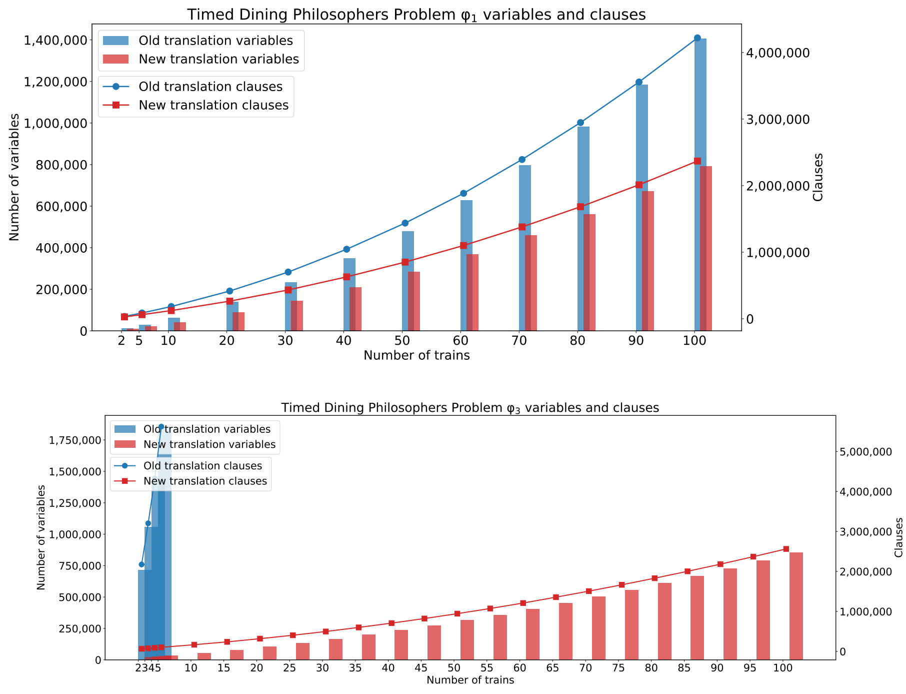


Figure 7. The TDPP with n philosophers clauses and variables: φ_1 and φ_3.

5.2. Timed Generic Pipeline Paradigm

The TGPP (Figure 8) discrete timed automata model [26] consists of a producer producing data within the time interval ($[a, b]$) or being inactive, a consumer receiving data within the time interval ($[c, d]$) or being inactive within the time interval ($[g, h]$), and a chain of n intermediate nodes which can be ready for receiving data within the time interval ($[c, d]$), processing data within the 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.

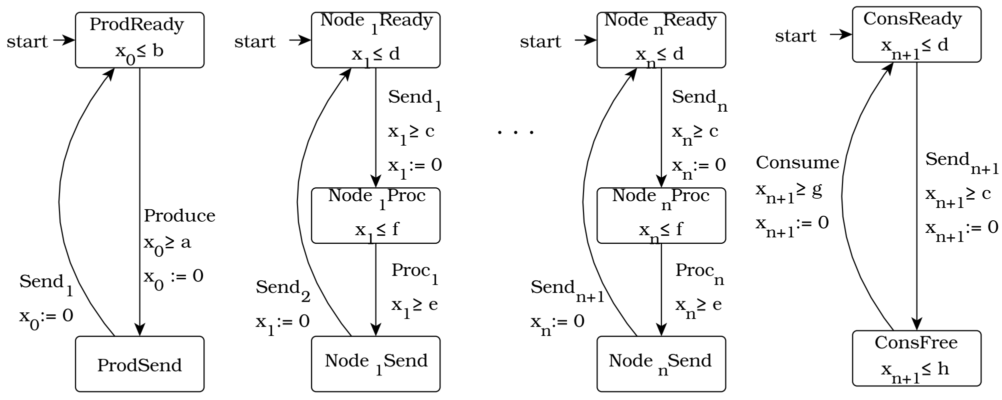


Figure 8. The TGPP system.

To compare our experimental results with [26], we tested the TGPP discrete timed automata model on the following MTL formulae that existentially hold in the model of TGPP (n is the number of nodes). In the below formulae, we use $prod_0$, $prod_1$, $cons_0$, and $cons_1$ for $ProdReady$, $ProdSend$, $ConsReady$, and $ConsFree$ respectively. Moreover, we write $\mathbf{G}$ for $\mathbf{G}_{[0,\infty)}$. Let us consider the following formulae:

- $\varphi_1 = \mathbf{G}(prod_0 \vee cons_0)$. It states that always either the producer has sent the data or the consumer has received the data.
- $\varphi_2 = \mathbf{F}_{[0,2 \cdot n+3)}(\mathbf{G}(prod_1 \vee cons_1))$. 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.
- $\varphi_3 = \mathbf{G}(\mathbf{F}_{[0,2 \cdot n+3)}(cons_1))$. It states that the Consumer infinitely often eventually receives the data in time less than $2 \cdot n + 3$ units.

All these formulae are existentially valid in the model of TGPP.

Charts in Figure 9 show the total time usage and total memory usage for TGPP needed for verification φ_1 and φ_2. In both cases, the new method outperforms the old one. For φ_2, the LTLq-based method was able to verify the system with 19 nodes, and the HLTL-based method was able to verify the system only with 9 nodes. For φ_3, the memory usage is similar in both cases. However, the time usage for the old method exponentially grows. The second plot shows the number of generated clauses and variables.

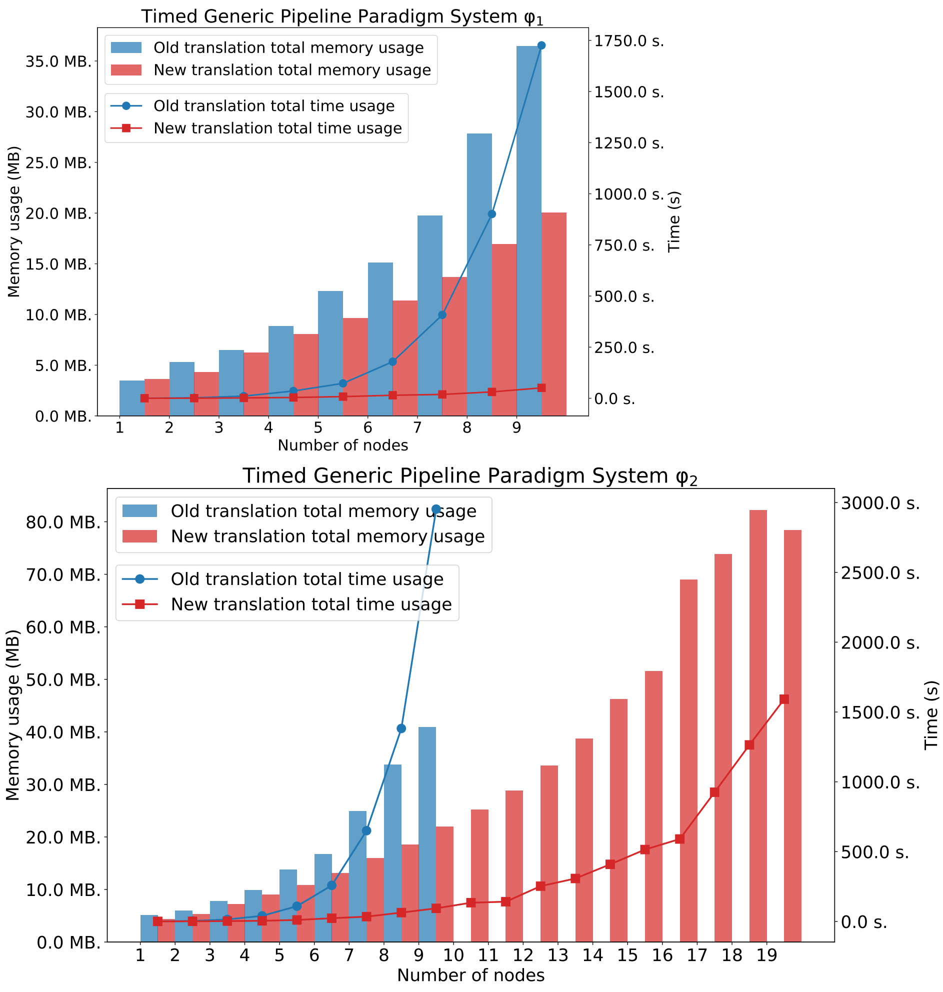


Figure 9. φ_1 and φ_2: TGPP with n nodes.

Charts in Figure 10 shows the total time usage and total memory usage for TGPP needed for verification φ_3. The second plot shows the number of generated clauses and variables.

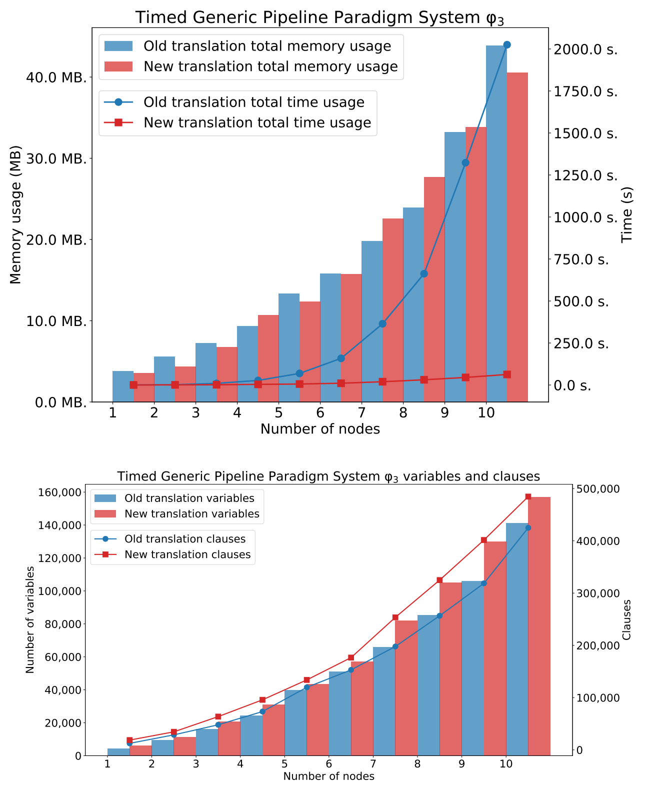


Figure 10. Results for φ_3: TGPP with n nodes. Number of variables and clauses for φ_3.

5.3. Timed Train Controller System

The TTCS (Figure 11) consists of n (for $n \geq 2$) trains $T_1, \ldots, T_n$, each one using its own circular track for traveling in one direction and containing its own clock x_i, together with controller C used to coordinate the access of trains to the tunnel through which all trains have to pass at a certain point. There is only one track in the tunnel, so trains arriving from each direction cannot use it in the same time. There are signals on both sides of the tunnel, which can be either red or green. All trains notify the controller when they request entry to the tunnel or when they leave the tunnel. The controller controls the color of the displayed signal, and the behavior of the scenario depends on the values δ and Δ ($\Delta > \delta + 1$ makes it incorrect—the mutual exclusion does not hold).

Controller C has $n+1$ locations, with the location 0 being the initial one. The action $Start_i$ of train T_i denotes the passage from the location away to the location where the train wishes to obtain access to the tunnel. This is allowed only if controller C is in location 0. Similarly, train T_i synchronizes with controller C on action $approach_i$, which denotes setting C to location $i > 0$, as well as out_i, which denotes setting C to location 0. Finally, action in_i denotes the entering of train T_i into the tunnel.

Moreover, we assume the following set of propositional variables: $\mathcal{AP} = \{tunnel_1, \ldots, tunnel_n\}$.

Let us consider the following formulae:

- $\varphi_1 = \mathbf{F}_{[0,2\cdot\delta+4)}(\bigvee_{i=1}^{n-1}\bigvee_{j=i+1}^{n})(tunnel_i \wedge tunnel_j)$. It expresses that the system violates the mutual exclusion property.
- $\varphi_2 = \mathbf{G}(\mathbf{F}_{[0,2\cdot\delta+1)} tunnel_1)$. It expresses that the first train can infinitely often and from any state enter the tunnel in time less than $2 \cdot \delta + 1$.
- $\varphi_3 = \mathbf{G}(\mathbf{F}_{[0,2\cdot\delta+7)} tunnel_1 \wedge \mathbf{F}_{[0,2\cdot\delta+7)} \neg tunnel_1)$. It expresses that the first train is infinitely often in the tunnel and outside the tunnel in time less than $2 \cdot \delta + 7$.

All these formulae are existentially valid in the model of TTCS.

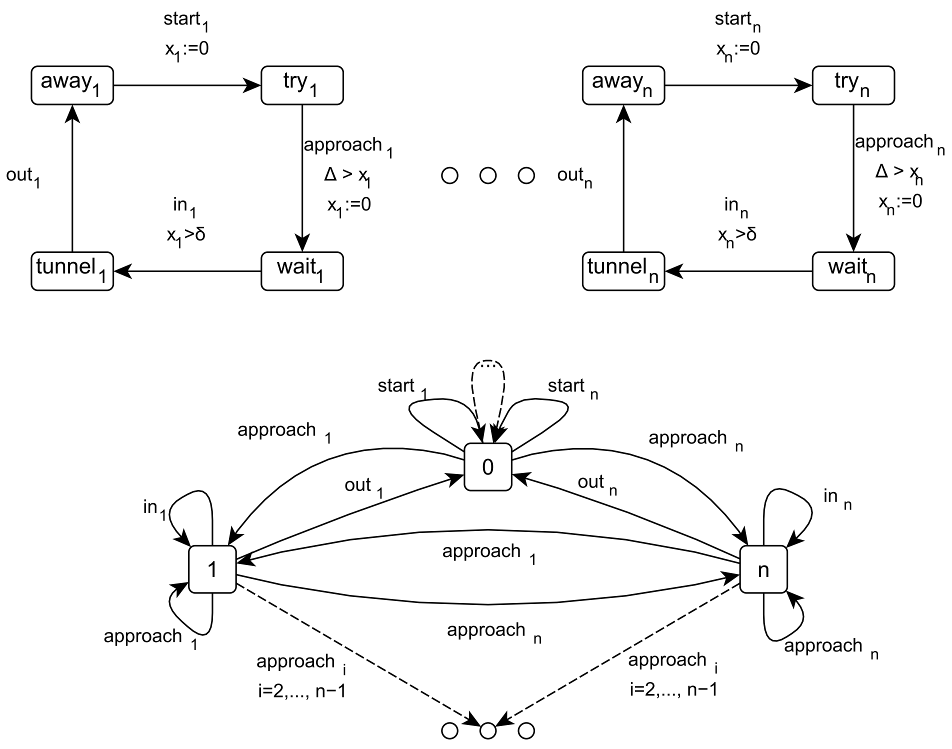


Figure 11. The TTCS system.

As we can see in Figures 12, 13 and 14, the new method surpasses the old one. As we expected, the difference between the two methods is smaller for the simple formula that expresses reachability problem (φ_1). However, a significant difference can be seen for the formulae φ_2 and φ_3. Figure 14 also shows the number of clauses and variables for the new and the old method. As we can see, the numbers of variables and clauses grow exponentially for the old method.

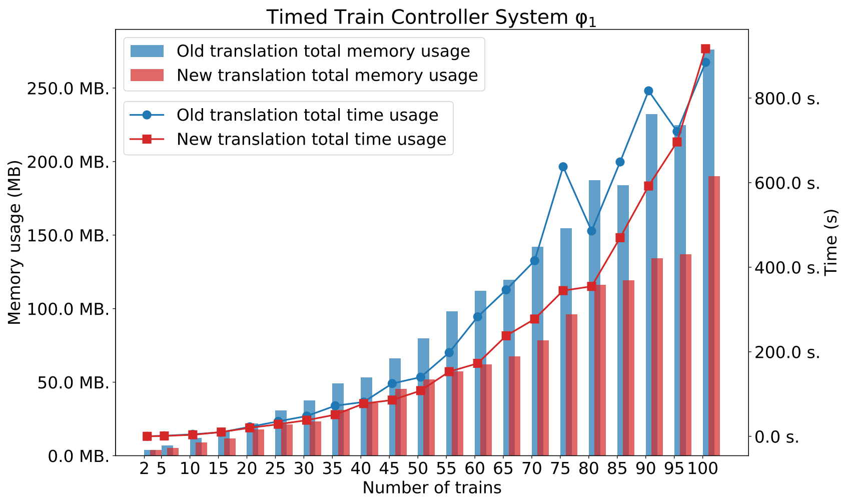


Figure 12. φ_1: TTCS with n trains.

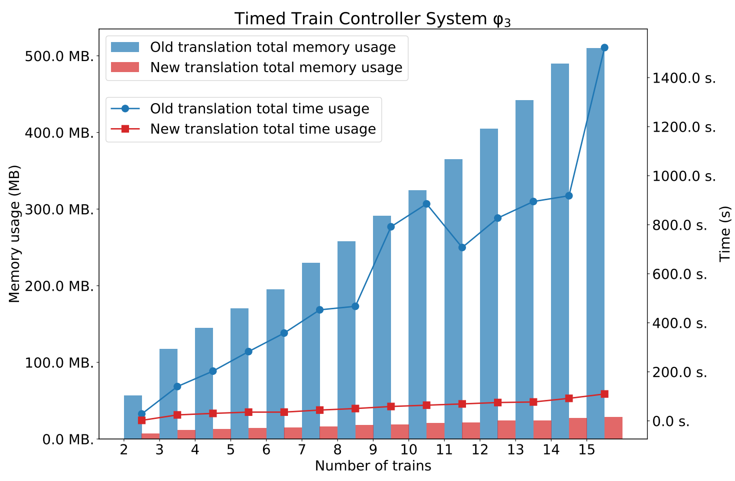
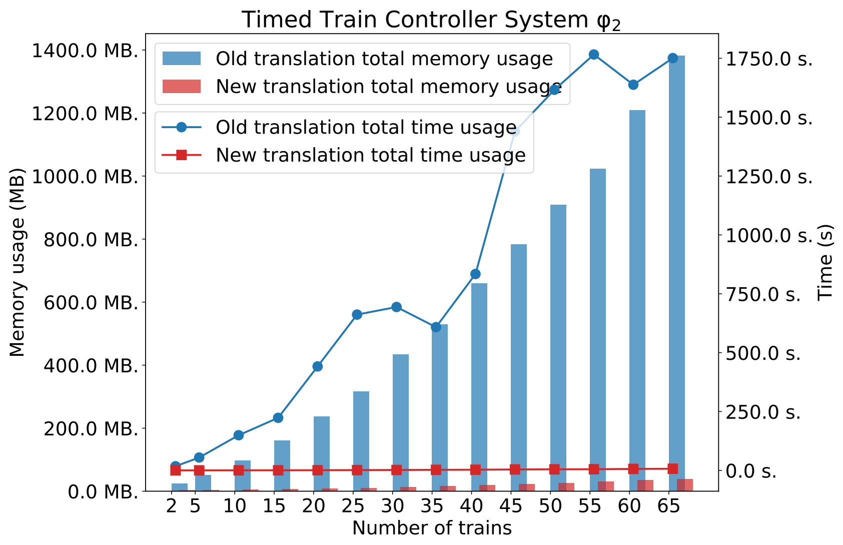


Figure 13. φ_3: TTCS with n trains.



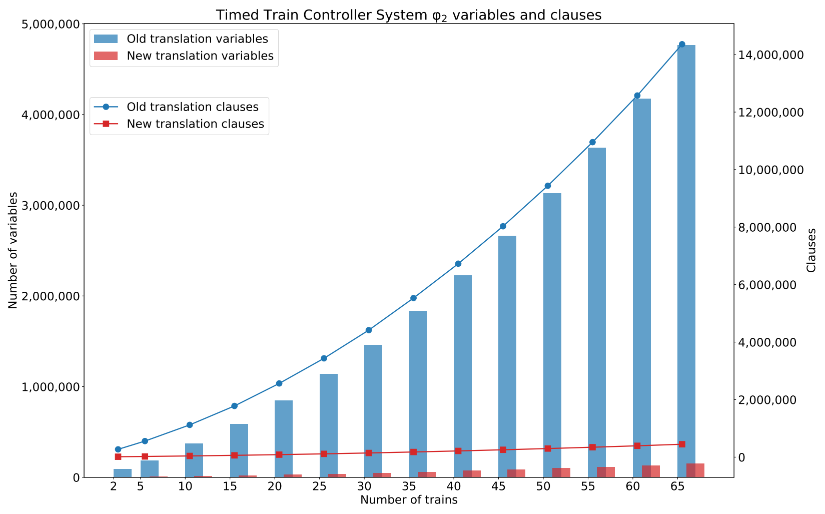


Figure 14. Results for φ_2. Number of variables and clauses for φ_2 and TTCS with n trains.

6. Statistics

We performed one- and two-sided Wicoxon tests for DPTT (Figure 15). Tests showed that the new method outperforms the old one: the new method used less time ($p = 0.36$), and used less memory ($p = 0.99$).

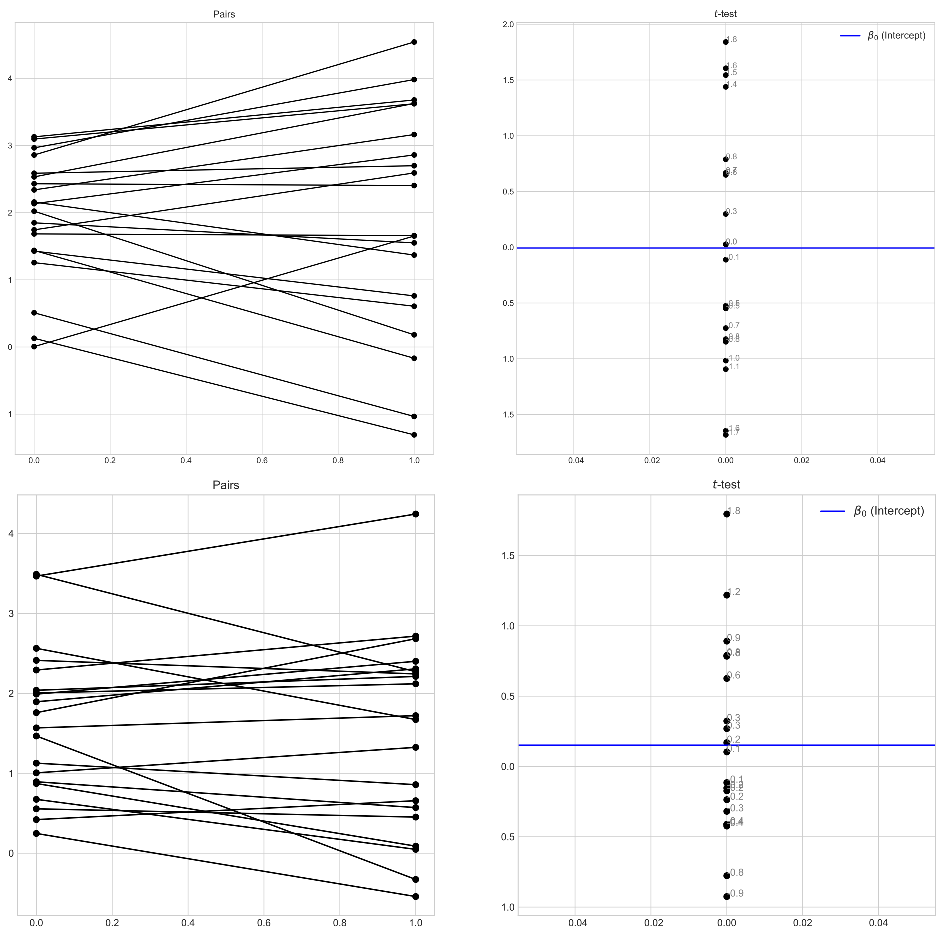


Figure 15. TDPP: Pairs Wilcoxon plots for total time usage and total memory usage for φ_1, φ_2, and φ_3.

We performed the two-sided and one-sided Wilcoxon tests for all the experiments. As a dataset, we took the whole set of the experimental results (note that we deleted some results in the figures in Section 5 to make them clear—whole data can be found in the `.tar.xz` file we delivered).

7. Conclusions

In this work, we proposed a new SAT-based BMC for soft real-time systems modeled by discrete time automata with digital clocks and for properties expressible in metric temporal logic with semantics over discrete time automata with digital clocks.

The first step of this method is translating the existential model-checking problem for MTL into the existential model-checking problem for LTLq logic by replacing temporal operators with intervals (MTL) with temporal operators and new propositional variables corresponding to these intervals (LTLq). The second step is translating the existential model-checking problem for LTLq into the satisfiability problem for the propositional formulae. The efficiency of the new method is due to the fact that only one additional clock for measuring global time is needed, unlike the earlier method [26], which translates the existential model-checking problem for MTL into the existential model-checking problem translation to HLTL.

The earlier method [26] needs to add to a timed automaton one extra clock, one extra path, and an extra transition for each occurrence of the temporal operator in the formula.

We implemented our method as a standalone program written in the programming language C++. This implementation allowed us to experimentally evaluate and compare the new approach with the old one.

The experimental results show that our approach is significantly better than the approach based on translation to HLTL. The new method substantially reduces the conjunction normal form (CNF) formula's size, an input formula for the SAT solver. The reduced size of the CNF formula causes the SAT solver to use much less time and memory to determine the satisfiability of the input formula.

In future work, we plan to extend our method by adding discrete data [56]. We also would like to improve and develop and prove the method presented in [49].

Author Contributions: Conceptualisation, A.M.Z. and A.Z.; methodology, A.M.Z. and A.Z.; software, A.Z.; validation, A.M.Z.; formal analysis, A.M.Z.; proving, A.M.Z.; investigation, A.M.Z. and A.Z.; writing—original draft preparation, A.M.Z.; writing—review and editing, A.M.Z and A.Z.; visualisations, A.M.Z.; supervision, A.Z.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MTL	Metric Temporal Logic
HLTL	Hard Reset LTL
LTL	Linear Temporal Logic
TA	Timed Automata
DTA	Discrete Timed Automaton
BMC	Bounded Model Checking
LTLq	Linear Temporal Logic with Additional Propositional Variables q_I
TGPP	Timed Generic Pipeline Paradigm
TTCS	Timed Train Controller System
TDPP	Timed Dining Philosophers Problem

Appendix A. Improvements and Extensions Compared to the Workshop Paper

1. We improved it and proved the main theorem. The workshop paper presented only the idea of the method;
2. We improved definitions;
3. We changed the semantics: the weakly monotonic semantics seems to be more natural in the case of discrete time. In [1], we used strongly monotonic semantics;
4. We redefined the concrete model. The process of creating the concrete model presented in [26] was unnecessarily complicated;
5. We redefined the semantics of MTL;
6. We showed the translation to SAT for the LTLq formula on the path starting at point m at the depth d;
7. We extended the experimental section by adding the timed dining philosophers problem (to the best of our knowledge, we modeled TDP for the first time as a network of discrete timed automata—we could find only the modeling using timed Petri nets in the literature).

Appendix B. Code Reproducibility

Appendix B.1. Preliminary

This code was successfully tested on the Linux operating system. Requirements for running the benchmarks: a reasonably modern 64-bit Linux environment with Python 3 installed. As all the programs in our package are statically linked, they do not rely on the particular version of libraries available on the final system.

In order to run the code of the experiments, one needs to download the supplementary material (`.tar.xz` file attached from https://tinyurl.com/bmc4dtta-mtl, accessed on 3 November 2022) and unpack it:

```
$ tar -xvf bmc4dtta-emtl.tar.gz
```

Once unpacked, one should go to the directory named `bmc4dtta-emtl` that contains the code necessary to replicate all the experiments:

```
$ cd bmc4dtta-emtl
```

Appendix B.2. Running Experiments

To launch a SAT-based BMC for DTTA and LTLq experiment, go to the `newbmc\{SYSTEM_ACRONYM-f{FORMULA_NR}}\` directory, where `{FORMULA_NR}` is a natural number $n \in \{1,2,3\}$, and `{SYSTEM_ACRONYM}` is either '`ttcs`', '`tgpp`' or '`tdpl`', and run the bash file `all-new-{SYSTEM_ACRONYM}-f{FORMULA_NR}.sh`. We report a few usage examples below.

Appendix B.3. Example-TTCS

Let us suppose that we want to verify TTCS system with 25 trains and formula $\varphi_1 = \mathbf{F}_{[0,2\cdot\delta+7)}(\bigvee_{i=1}^{n-1}\bigvee_{j=i+1}^{n})(tunnel_i \wedge tunnel_j))$. We have to navigate to the directory `ttcs-f1`

```
$ cd ttcs-f1
```

and then execute the BMC algorithm

```
$ nohup nice ./bmcalg-new-ttcs-f1-cms.sh 25 0 &
```

The first argument sets the number of trains in the system to 25, and the second argument sets the initial k-path k to 0.

One can also perform all the experiments for the formula φ_1 using the proper script:

```
$ nohup nice ./all-new-ttcs-f1-cms.sh &
```

It will perform all the experiments for $n \in \{2\ldots100\}$.

References

1. Zbrzezny, A.M.; Zbrzezny, A. Simple Bounded MTL Model Checking for Discrete Timed Automata (Extended abstract). In Proceedings of the 23th International Workshop on Concurrency, Specification and Programming (CS&P 2016), Rostock, Germany, 28–30 September, 2016; Volume 1698, CEUR Workshop Proceedings, pp. 37–48.
2. Bourke, T.; Sowmya, A. Analyzing an Embedded Sensor with Timed Automata in Uppaal. *ACM Trans. Embed. Comput. Syst. (TECS)* **2013**, *13*, 44-1–44-26. [CrossRef]
3. Chen, G.; Jiang, T.; Wang, M.; Tang, X.; Ji, W. Design and model checking of timed automata oriented architecture for Internet of thing. *Int. J. Distrib. Sens. Netw.* **2020**, *16*, 1550147720911008. [CrossRef]
4. Iversen, T.K.; Kristoffersen, K.J.; Larsen, K.G.; Laursen, M.; Madsen, R.G.; Mortensen, S.K.; Pettersson, P.; Thomasen, C.B. Model-checking real-time control programs: Verifying Lego(R) Mindstorms^{TM} systems using UPPAAL. In Proceedings of the 12th Euromicro Conference on Real-Time Systems (ECRTS 2000), Stockholm, Sweden, 19–21 June 2000; IEEE Computer Society: Washington, DC, USA, 2000; pp. 147–155. [CrossRef]
5. Lahtinen, J. *Model Checking Timed Safety Instrumented Systems*; Research Report TKK-ICS-R3; Helsinki University of Technology, Department of Information and Computer Science: Espoo, Finland, 2008.

6. Hammal, Y.; Monnet, Q.; Mokdad, L.; Ben-Othman, J.; Abdelli, A. Timed automata based modeling and verification of denial of service attacks in wireless sensor networks. *Stud. Inform. Universalis* **2014**, *12*, 1–46.
7. Mouradian, A.; Augé-Blum, I. Modeling Local Broadcast Behavior of Wireless Sensor Networks with Timed Automata for Model Checking of WCTT. In Proceedings of the WCTT'12, San Juan, Puerto Rico, 4 December 2012; pp. 23–30.
8. Alur, R.; Dill, D. A Theory of Timed Automata. *Theor. Comput. Sci.* **1994**, *126*, 183–235. [CrossRef]
9. Bozga, M.; Hou, J.; Maler, O.; Yovine, S. Verification of Asynchronous Circuits using Timed Automata. *Electr. Notes Theor. Comput. Sci.* **2002**, *65*, 47–59. [CrossRef]
10. Dierks, H. PLC-automata: A new class of implementable real-time automata. *Theor. Comput. Sci.* **2001**, *253*, 61–93. [CrossRef]
11. Clarke, E.M.; Emerson, E.A. Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In Proceedings of the Logics of Programs, Yorktown Heights, NY, USA, 4–6 May 1981; Springer: Berlin/Heidelberg, Germany, 1981; Volume 131, LNCS, pp. 52–71.
12. Emerson, E.A.; Mok, A.K.; Sistla, A.P.; Srinivasan, J. Quantitative Temporal Reasoning. *Real-Time Syst.* **1992**, *4*, 331–352. [CrossRef]
13. Pnueli, A. The Temporal Logic of Programs. In Proceedings of the 18th Annual Symposium on Foundations of Computer Science, Providence, RI, USA, 20–23 October 1977; pp. 46–57.
14. Koymans, R. Specifying Real-Time Properties with Metric Temporal Logic. *Real-Time Syst.* **1990**, *2*, 255–299. [CrossRef]
15. Bouyer, P. Model-checking Timed Temporal Logics. *Electr. Notes Theor. Comput. Sci.* **2009**, *231*, 323–341. [CrossRef]
16. Furia, C.A.; Spoletini, P. Tomorrow and All our Yesterdays: MTL Satisfiability over the Integers. In Proceedings of the ICTAC, Istanbul, Turkey, 1–3 September 2008; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5160, LNCS; pp. 126–140.
17. Ho, H.; Ouaknine, J.; Worrell, J. On the Expressiveness and Monitoring of Metric Temporal Logic. *Logical Methods in Comp. Sci.* **2019**, *15* . [CrossRef]
18. Pradella, M.; Morzenti, A.; Pietro, P.S. Bounded satisfiability checking of metric temporal logic specifications. *ACM Trans. Softw. Eng. Methodol.* **2013**, *22*, 20:1–20:54. [CrossRef]
19. Henzinger, T.; Manna, Z.; Pnueli, A. What good are digital clocks? In Proceedings of the ICALP 92: Automata, Languages, and Programming, Wien, Austria, 13–17 July 1992; Kuich, W., Ed.; Lecture Notes in Computer Science 623; Springer: Berlin/Heidelberg, Germany, 1992; pp. 545–558.
20. Biere, A.; Cimatti, A.; Clarke, E.; Zhu, Y. Symbolic Model Checking without BDDs. In Proceedings of the TACAS'99, Amsterdam, The Netherlands, 22–28 March 1999; Springer: Berlin/Heidelberg, Germany, 1999; Volume 1579, LNCS, pp. 193–207.
21. Biere, A.; Cimatti, A.; Clarke, E.M.; Strichman, O.; Zhu, Y. Bounded Model Checking. *Adv. Comput.* **2003**, *58*, 117–148.
22. Penczek, W.; Woźna, B.; Zbrzezny, A. Bounded Model Checking for the Universal Fragment of CTL. *Fundam. Inform.* **2002**, *51*, 135–156.
23. Alur, R.; Henzinger, T.A. Real-time Logics: Complexity and Expressiveness. In Proceedings of the LICS '90, Philadelphia, PA, USA, 4–7 June 1990; pp. 390–401.
24. Alur, R.; Feder, T.; Henzinger, T.A. The Benefits of Relaxing Punctuality. *J. ACM* **1996**, *43*, 116–146. [CrossRef]
25. Wilke, T. Specifying Timed State Sequences in Powerful Decidable Logics and Timed Automata. In Proceedings of the Formal Techniques in Real-Time and Fault-Tolerant Systems, Lübeck, Germany, 19–23 September 1994; pp. 694–715.
26. Woźna-Szcześniak, B.; Zbrzezny, A. Checking MTL Properties of Discrete Timed Automata via Bounded Model Checking. *Fundam. Inform.* **2014**, *135*, 553–568. [CrossRef]
27. Alur, R.; Henzinger, T.A. Logics and Models of Real Time: A Survey. In Proceedings of the Real-Time: Theory in Practice, REX Workshop, Mook, The Netherlands, 3–7 June 1991; de Bakker, J.W., Huizing, C., de Roever, W.P., Rozenberg, G., Eds.; Springer: Berlin/Heidelberg, Germany, 1991; Volume 600, Lecture Notes in Computer Science, pp. 74–106. [CrossRef]
28. Bozga, M.; Maler, O.; Tripakis, S. Efficient Verification of Timed Automata Using Dense and Discrete Time Semantics. In Proceedings of the Correct Hardware Design and Verification Methods, 10th IFIP WG 10.5 Advanced Research Working Conference, CHARME '99, Bad Herrenalb, Germany, 27–29 September 1999; Pierre, L., Kropf, T., Eds.; Springer: Berlin/Heidelberg, Germany, 1999; Volume 1703, Lecture Notes in Computer Science, pp. 125–141. [CrossRef]
29. Ruf, J.; Kropf, T. Symbolic Verification and Analysis of Discrete Timed Systems. *Form. Methods Syst. Des.* **2003**, *23*, 67–108. [CrossRef]
30. Cimatti, A.; Griggio, A.; Magnago, E.; Roveri, M.; Tonetta, S. Extending nuXmv with timed transition systems and timed temporal properties. In Proceedings of the International Conference on Computer Aided Verification, New York, NY, USA, 15–18 July 2019; Springer: Cham, Switzerland, 2019; pp. 376–386.
31. Gao, Y.; Abate, A.; Jiang, F.J.; Giacobbe, M.; Xie, L.; Johansson, K.H. Temporal logic trees for model checking and control synthesis of uncertain discrete-time systems. *IEEE Trans. Autom. Control* **2021**, *67*, 5071–5086. [CrossRef]
32. Laroussinie, F.; Markey, N.; Schnoebelen, P. Efficient timed model checking for discrete-time systems. *Theor. Comput. Sci.* **2006**, *353*, 249–271. [CrossRef]
33. Krystosik, A. Embedded Systems Modeling Language. In Proceedings of the 2006 International Conference on Dependability of Computer Systems (DepCoS-RELCOMEX 2006), Szklarska Poreba, Poland, 24–28 May 2006; IEEE Computer Society: Washington, DC, USA, 2006; pp. 27–34. [CrossRef]
34. Bruneel, H.; Kim, B.G. *Discrete-Time Models for Communication Systems Including ATM*; Springer Science & Business Media: New York, NY, USA, 2012; Volume 205.

35. Belta, C.; Yordanov, B.; Gol, E.A. *Formal Methods for Discrete-Time Dynamical Systems*; Springer: Cham, Switzerland, 2017; Volume 15.
36. Allen, L.; Jones, M.; Martin, C. A discrete-time model with vaccination for a measles epidemic. *Math. Biosci.* **1991**, *105*, 111–131. [CrossRef] [PubMed]
37. Li, S.; Lu, Y.; Garrido, J. A review of discrete-time risk models. *RACSAM-Rev. De La Real Acad. De Cienc. Exactas Fis. Y Nat. Ser. A Mat.* **2009**, *103*, 321–337. [CrossRef]
38. Oli, M.K.; Venkataraman, M.; Klein, P.A.; Wendland, L.D.; Brown, M.B. Population dynamics of infectious diseases: A discrete time model. *Ecol. Model.* **2006**, *198*, 183–194. [CrossRef]
39. Quaas, K. MTL-Model Checking of One-Clock Parametric Timed Automata is Undecidable. In Proceedings of the 1st International Workshop on Synthesis of Continuous Parameters, SynCoP 2014, Grenoble, France, 6 April 2014; André, É., Frehse, G., Eds.; Open Publishing Association: Waterloo, Australia, 2014; Volume 145, EPTCS, pp. 5–17. [CrossRef]
40. Bae, K.; Lee, J. Bounded model checking of signal temporal logic properties using syntactic separation. *Proc. ACM Program. Lang.* **2019**, *3*, 1–30. [CrossRef]
41. Li, J.; Vardi, M.Y.; Rozier, K.Y. Satisfiability checking for mission-time LTL. In Proceedings of the International Conference on Computer Aided Verification, New York, NY, USA, 15–18 July 2019; Springer: Cham, Switzerland, 2019, pp. 3–22.
42. Jonk, R.; Voeten, J.; Geilen, M.; Basten, T.; Schiffelers, R. SMT-based verification of temporal properties for component-based software systems. *IFAC-PapersOnLine* **2020**, *53*, 493–500. [CrossRef]
43. Smith, R.L.; Bersani, M.M.; Rossi, M.; San Pietro, P. Improved Bounded Model Checking of Timed Automata. In Proceedings of the 9th IEEE/ACM International Conference on Formal Methods in Software Engineering, FormaliSE@ICSE 2021, Madrid, Spain, 17–21 May 2021; Bliudze, S., Gnesi, S., Plat, N., Semini, L., Eds.; IEEE: Piscataway, NJ, USA, 2021; pp. 97–110. [CrossRef]
44. Hofmann, T.; Schupp, S. Controlling Timed Automata against MTL Specifications with TACoS. *Sci. Comput. Program.* **2022**, *225*, 102898. [CrossRef]
45. Hustadt, U.; Ozaki, A.; Dixon, C. Theorem Proving for Pointwise Metric Temporal Logic Over the Naturals via Translations. *J. Autom. Reason.* **2020**, *64*, 1553–1610. [CrossRef]
46. Ouaknine, J.; Worrell, J. Some Recent Results in Metric Temporal Logic. In Proceedings of the Formal Modeling and Analysis of Timed Systems, 6th International Conference, FORMATS 2008, Saint Malo, France, 15–17 September 2008; Cassez, F., Jard, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5215, Lecture Notes in Computer Science, pp. 1–13. [CrossRef]
47. D'Souza, D.; Prabhakar, P. On the expressiveness of MTL in the pointwise and continuous semantics. *Int. J. Softw. Tools Technol. Transf.* **2007**, *9*, 1–4. [CrossRef]
48. Bouyer, P.; Chevalier, F.; Markey, N. On the expressiveness of TPTL and MTL. *Inf. Comput.* **2010**, *208*, 97–116. [CrossRef]
49. Zbrzezny, A.M.; Zbrzezny, A. Checking MTL Properties of Timed Automata with Dense Time using Satisfiability Modulo Theories (Extended Abstract). In Proceedings of the 28th Intl. Workshop on CS&P, Olsztyn, Poland, 24–26 September 2019; Volume 2571, CEUR Workshop Proc.
50. Bonakdarpour, B.; Prabhakar, P.; Sánchez, C. Model checking timed hyperproperties in discrete-time systems. In Proceedings of the NASA Formal Methods Symposium, Moffett Field, CA, USA, 11–15 May 2020; Springer: Cham, Switzerland, 2020; pp. 311–328.
51. Penczek, W.; Półrola, A. *Advances in Verification of Time Petri Nets and Timed Automata: A Temporal Logic Approach*; Studies in Computational Intelligence; Springer: Berlin/Heidelberg, Germany, 2006; Volume 20.
52. Tripakis, S.; Yovine, S. Analysis of Timed Systems Using Time-Abstracting Bisimulations. *Form. Methods Syst. Des.* **2001**, *18*, 25–68. [CrossRef]
53. Zbrzezny, A. A new translation from ECTL* to SAT. *Fundam. Informaticae* **2012**, *120*, 377–397. [CrossRef]
54. Biere, A.; Fazekas, K.; Fleury, M.; Heisinger, M. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling Entering the SAT Competition 2020. In Proceedings of the SAT Competition 2020–Solver and Benchmark Descriptions, virtual event affiliated with the 23rd International Conference on Theory and Applications of Satisfiability Testing, Alghero, Italy, 5–9 July 2020; Balyo, T., Froleyks, N., Heule, M., Iser, M., Järvisalo, M., Suda, M., Eds.; University of Helsinki: Helsinki, Finland, 2020; Volume B-2020-1, Department of Computer Science Report Series B, pp. 51–53.
55. Probst, D.K.; Li, H.F. Verifying Timed Behavior Automata with Nonbinary Delay Constraints. In Proceedings of the Computer Aided Verification, Fourth International Workshop, CAV '92, Montreal, QC, Canada, 29 June–1 July 1992; von Bochmann, G., Probst, D.K., Eds.; Springer: Berlin/Heidelberg, Germany, 1992; Volume 663, Lecture Notes in Computer Science, pp. 123–136. [CrossRef]
56. Zbrzezny, A.; Pólrola, A. SAT-Based Reachability Checking for Timed Automata with Discrete Data. *Fundam. Informaticae* **2007**, *79*, 579–593.