Practical Automated Partial Verification of Multi-Paradigm Real-Time Models

Carlo A. Furia[1], Matteo Pradella[2], and Matteo Rossi[1]

[1] Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy
[2] CNR IEIIT-MI, Milano, Italy
{furia, pradella, rossi}@elet.polimi.it
http://home.dei.polimi.it/lastname/

Abstract. This article introduces a fully automated verification technique that permits to analyze real-time systems described using a continuous notion of time and a mixture of operational (i.e., automata-based) and descriptive (i.e., logic-based) formalisms. The technique relies on the reduction, under reasonable assumptions, of the continuous-time verification problem to its discrete-time counterpart. This reconciles in a viable and effective way the dense/discrete and operational/descriptive dichotomies that are often encountered in practice when it comes to specifying and analyzing complex critical systems. The article investigates the applicability of the technique through a significant example centered on a communication protocol. Concurrent runs of the protocol are formalized by parallel instances of a Timed Automaton, while the synchronization rules between these instances are specified through Metric Temporal Logic formulas, thus creating a multi-paradigm model. Verification tests run on this model using a bounded satisfiability checker implementing the technique show consistent results and interesting performances.

Keywords: Metric temporal logic, timed automata, discretization, dense time, bounded model checking.

1 Introduction

There is a tension between the standpoints of modeling and of verification when it comes to choosing a formal notation. The ideal modeling language would be very expressive, thus capturing sophisticated features of systems in a natural and straightforward manner; in particular, for concurrent and real-time systems, a dense time model is the intuitive choice to model true asynchrony. On the other hand, expressiveness is often traded off against complexity (and decidability), hence the desire for a feasible and fully automated verification process pulls in the opposite direction of more primitive, and less expressive, models of time and systems. Discrete time, for instance, is usually more amenable to automated verification, and quite mature techniques and tools can be deployed to verify systems modeled under this assumption.

Another, orthogonal, concern of the real-time modeler is the choice between operational and descriptive modeling languages. Typical examples of operational notations are Timed Automata (TA) and Timed Petri Nets, while temporal logics are popular

instances of descriptive notations. Operational and descriptive notations have complementary strengths and weaknesses. For instance, temporal logics are very effective for describing partial models or requirements about the past (through the natural use of past operators); automata-based notations, on the other hand, model systems through the notions of state and transition, and are typically easy to simulate and visualize. From a modeling viewpoint, the possibility of integrating multiple modeling paradigms in formalizing a system would be highly desirable.

This paper introduces a verification technique that, under suitable assumptions, reconciles the dense/discrete and operational/descriptive dichotomies in an effective way. Its goal is to provide a practical means to carry out verification of real-time systems described using a dense notion of time and a mixture of operational and descriptive notations. This approach both permits to analyze continuous-time models using fully automated, discrete-time verification techniques, and allows users to mix operational (TA) and descriptive (Metric Temporal Logic, MTL) formalisms in the same specification. The technique involves an automated translation of the operational component into temporal logic notation. The resulting MTL model, which describes both the system and the properties to be verified, is then discretized according to the technique introduced in [8]. The technique is partial in two respects: it can fail to provide conclusive answers, and only dense-time behaviors with bounded variability are verified. The most common approaches to similar verification problems are in fact usually complementary, and involve translating the logic into automata [2]. Our choice is mainly justified by the fact that logic formulas are naturally compositional, which facilitates our ultimate goal of formally combining heterogeneous models.

In this article, we start by providing a dense-time MTL axiomatization of TA. Due to a well-known expressiveness gap between temporal logics and automata [12] in general it is impossible to describe the language accepted by a TA through an MTL formula. What we provide is instead an MTL formalization of the *accepting runs* of a TA; i.e., we model the overall behavior of TA through a set of MTL axioms. It is well-known that MTL is undecidable over dense time [4]; however, this obstacle can be mitigated in practice through the *discretization* technique introduced — and demonstrated to be practically appealing — in [8]. The undecidability of dense-time MTL entails that the reduction technique must be incomplete, i.e., there are cases in which we are unable to solve the verification problem in a conclusive manner. However, as shown in [8], the impact of this shortcoming can be reduced in many practical cases. We then show that this approach yields poor results if done naïvely. Hence, we carefully revise the axiomatization and put it in a form that is much more amenable to discretization, obtaining a set of discretized MTL formulas describing TA runs. These axioms can then be combined with other modules written in MTL, and with the properties to be verified. The resulting model can be analyzed by means of automated discrete-time tools; the results of this analysis are then used to finally infer results about the verification of the original dense-time model. We provide an implementation based on the ℤot bounded satisfiability checker [16].

To investigate the effectiveness of the technique, we experimented with a significant example centered on the description of a communication protocol by means of a TA. Concurrent runs of the protocol are formalized by parallel instances of the same

automaton; additionally, the synchronization rules between these instances are formalized by means of MTL formulas, thus building a multi-paradigm model. Verification tests were run on these models using the ℤot-based tool. The experimental results are encouraging, both in terms of performances and in terms of "completeness coverage" of the method.

In fact, our approach aims at providing a *practical* approach to the verification of multi-paradigm models. Hence, we sacrifice completeness in order to have a lightweight and flexible technique. Also note that, although in this paper TA are the operational formalism of choice, the same approach could be applied to other operational formalisms, such as Timed Petri Nets.

The paper is organized as follows. Section 1.1 briefly summarizes some related research. Section 2 introduces the technical definitions that are needed in the remainder, namely the syntax and semantics of MTL and TA, and the discretization technique from [10, 8] that will be used. Section 3 shows how to formalize the behavior of TA as a set of dense-time MTL formulas. Then, Section 4 re-examines the axioms and suitably modifies them in a way which is most amenable to the application of the discretization technique. Section 5 describes the example of a simple communication protocol and reports on the experiments conducted on it with the SAT-based implementation of the technique. Finally, Section 6 draws some conclusions.

1.1 Related Work

To the best of our knowledge, our approach is rather unique in combining operational and descriptive formalisms over dense time, trading-off verification completeness against more performing and practical verification results. On the other hand, each of the "ingredients" of our method has been studied in isolation in the literature. In this section we briefly recall a few of the most important results in this respect.

Dense-time verification of operational models is an active field, and it has produced a few very performing tools and methods. Let us mention, for instance, Uppaal [14] for the verification of TA. Although tools such as Uppaal exploit a descriptive notation to express the properties to be verified, the temporal logic subset is quite simple and of limited expressive power. In contrast, we allow basically full MTL to be freely used in both the description of the model and in the formalization of the properties to be verified, at the price of sacrificing completeness of verification.

MTL verification is also a well-understood research topic. MTL is known to be undecidable over dense time domains [4]. A well-known solution to this limitation restricts the syntax of MTL formulas to disallow the expression of exact (i.e., punctual) time distances [2]. The resulting logic, called MITL, is fully decidable over dense time. However, the associated decision procedures are rather difficult to implement in practice and, even if significant progresses have recently been made in simplifying them [15], a serviceable implementation is still lacking.

Another approach to circumvent the undecidability of dense-time MTL builds upon the fact that the same logic is decidable over discrete time. A few approaches introduce some notion of discretization, that is partial reduction of the verification problem from dense to discrete time. The present paper goes in this direction by extending previous work on MTL [8] to the case of TA. A different discretization technique, based on the

notion of robust satisfiability of MTL specifications, has been introduced in [5]. Another well-known notion of discretization is the one based on the concept of *digitization* [11], which has been applied by several authors to the practical verification of descriptive or operational formalisms. The interested reader may also see the related work section of [8] for a more thorough comparison of other discretization techniques.

2 Preliminaries and Definitions

2.1 Behaviors

Real-time system models describe the temporal behavior of some basic items and propositions, which represent the observable "facts" of the system. More precisely, an item it is characterized by a finite domain $\mathcal{D}^{\mathsf{it}}$ (and we write $\mathsf{it} : \mathcal{D}^{\mathsf{it}}$) such that at any instant of time it takes one of the values in $\mathcal{D}^{\mathsf{it}}$. On the other hand, a proposition p is simply a fact which can be true or false at any instant of time.

A *behavior* is a formal model of a *trace* (or *run*) of some real-time system. Given a time domain $\mathbb{T}$, a finite set $\mathcal{P}$ of atomic propositions, and a finite set of items $\mathcal{I}$, a behavior b is a mapping $b : \mathbb{T} \to \mathcal{D}^{\mathsf{it}_1} \times \mathcal{D}^{\mathsf{it}_2} \times \cdots \times \mathcal{D}^{\mathsf{it}_{|\mathcal{I}|}} \times 2^{\mathcal{P}}$ which associates with every time instant $t \in \mathbb{T}$ the tuple $b(t) = \langle v_1, v_2, \ldots, v_{|\mathcal{I}|}, P \rangle$ of item values and propositions that are true at t. $\mathcal{B}_{\mathbb{T}}$ denotes the set of all behaviors over $\mathbb{T}$, for an implicit fixed set of items and propositions. $b(t)|_{\mathsf{it}}$ and $b(t)|_{\mathcal{P}}$ denote the projection of the tuple $b(t)$ over the component corresponding to item it and the set of propositions in $2^{\mathcal{P}}$ respectively. Also, $t \in \mathbb{T}$ is a *transition point* for behavior b if t is a discontinuity point of the mapping b. Depending on whether $\mathbb{T}$ is a discrete, dense, or continuous set, we call a behavior over $\mathbb{T}$ discrete-, dense-, or continuous-time respectively. In this paper, we consider the natural numbers $\mathbb{N}$ as discrete-time domain and the nonnegative real numbers $\mathbb{R}_{\geq 0}$ as continuous-time (and dense-) time domain.

Non-Zeno and non-Berkeley. Over dense-time domains, it is customary to consider only physically meaningful behaviors, namely those respecting the so-called non-Zeno property. A behavior b is non-Zeno if the sequence of transition points of b has no accumulation points. For a non-Zeno behavior b, it is well-defined the notions of values to the left and to the right of any transition point $t > 0$, which we denote as $b^-(t)$ and $b^+(t)$, respectively. In this paper, we are interested in behaviors with a stronger requirement, called *non-Berkeleyness*. Informally, a behavior b is non-Berkeley for some positive constant $\delta \in \mathbb{R}_{>0}$ if, for all $t \in \mathbb{T}$, there exists a closed interval $[u, u + \delta]$ of size δ such that $t \in [u, u + \delta]$ and b is constant throughout $[u, u + \delta]$. Notice that a non-Berkeley behavior (for any δ) is non-Zeno *a fortiori*. The set of all non-Berkeley dense-time behaviors for $\delta > 0$ is denoted by $\mathcal{B}^{\delta}_{\chi} \subset \mathcal{B}_{\mathbb{R}_{\geq 0}}$. In the following we always assume behaviors to be non-Berkeley, unless explicitly stated otherwise.

Syntax and semantics. From a purely semantic point of view, one can consider the model of a (real-time) system simply as a set of behaviors [3, 7] over some time domain $\mathbb{T}$ and sets of items and propositions. In practice, however, every system is specified using some suitable notation. In this paper system models are represented through a mixture of MTL formulas [13, 4] and TA [1, 2]. The syntax and semantics of MTL and

TA are defined in the following. Given an MTL formula or a TA μ, and a behavior b, we write $b \models \mu$ to denote that b represents a system evolution which satisfies all the constraints imposed by μ. If $b \models \mu$ for some $b \in \mathcal{B}_{\mathbb{T}}$, μ is called $\mathbb{T}$-satisfiable; if $b \models \mu$ for all $b \in \mathcal{B}_{\mathbb{T}}$, μ is called $\mathbb{T}$-valid. Similarly, if $b \models \mu$ for some $b \in \mathcal{B}^{\delta}_{\chi}$, μ is called χ^{δ}-satisfiable; if $b \models \mu$ for all $b \in \mathcal{B}^{\delta}_{\chi}$, μ is called χ^{δ}-valid.

2.2 Metric Temporal Logic

Let $\mathcal{P}$ be a finite (non-empty) set of atomic propositions, $\mathcal{I}$ be a finite set of items, and $\mathcal{J}$ be the set of all (possibly unbounded) intervals of the time domain $\mathbb{T}$ with rational endpoints.We abbreviate intervals with pseudo-arithmetic expressions, such as $= d$, $< d$, $\geq d$, for $[d, d]$, $(0, d)$, and $[d, +\infty)$, respectively.

MTL syntax. The following grammar defines the syntax of MTL, where $I \in \mathcal{J}$ and β is a Boolean combination of atomic propositions or conditions over items.

$$\phi ::= \beta \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \mathsf{U}_I(\beta_1, \beta_2) \mid \mathsf{S}_I(\beta_1, \beta_2) \mid \mathsf{R}_I(\beta_1, \beta_2) \mid \mathsf{T}_I(\beta_1, \beta_2)$$

In order to ease the presentation of the discretization techniques in Section 2.4, MTL formulas are introduced in a ***flat*** normal form where negations are pushed down to (Boolean combinations of) atomic propositions, and temporal operators are not nested. It should be clear, however, that any MTL formula can be put into this form, possibly by introducing auxiliary propositional letters [6]. The basic temporal operators of MTL are the *bounded until* U_I (and its past counterpart *bounded since* S_I), as well as its dual *bounded release* R_I (and its past counterpart *bounded trigger* T_I). The subscripts I denote the interval of time over which every operator predicates. Throughout the paper we omit the explicit treatment of past operators (i.e., S_I and T_I) as it can be trivially derived from that of the corresponding future operators. In the following we assume a number of standard abbreviations, such as $\bot, \top, \Rightarrow, \Leftrightarrow$, and, when $I = (0, \infty)$, we drop the subscript interval of operators.

MTL semantics. MTL semantics is defined over behaviors, parametrically with respect to the choice of the time domain $\mathbb{T}$. In particular, the definition of the basic temporal operators is the following:

$b(t) \models_{\mathbb{T}} \mathsf{U}_I(\beta_1, \beta_2)$	iff	there exists $d \in I$ such that: $b(t+d) \models_{\mathbb{T}} \beta_2$ and, for all $u \in [0, d]$ it is $b(t+u) \models_{\mathbb{T}} \beta_1$
$b(t) \models_{\mathbb{T}} \mathsf{R}_I(\beta_1, \beta_2)$	iff	for all $d \in I$ it is: $b(t+d) \models_{\mathbb{T}} \beta_2$ or there exists a $u \in [0, d)$ such that $b(t+u) \models_{\mathbb{T}} \beta_1$
$b \models_{\mathbb{T}} \phi$	iff	for all $t \in \mathbb{T}$: $b(t) \models_{\mathbb{T}} \phi$

We remark that a global satisfiability semantics is assumed, i.e., the satisfiability of formulas is implicitly evaluated over *all* time instants in the time domain. This permits the direct and natural expression of most common real-time specifications (e.g., time-bounded response) without resorting to nesting of temporal operators. In addition, every generic MTL formulas with nesting temporal operators can be "flattened" to the form we introduced beforehand by introducing auxiliary propositions; in other words flat MTL and full MTL are equi-satisfiable .

Granularity. For an MTL formula ϕ, let $\mathcal{J}_\phi$ be the set of all non-null, finite interval bounds appearing in ϕ. Then, $\mathcal{D}_\phi$ is the set of positive values δ such that any interval bound in $\mathcal{J}_\phi$ is an integer if divided by δ.

OPERATOR	$\equiv$	DEFINITION
$\Diamond_I(\beta)$	$\equiv$	$\mathsf{U}_I(\top, \beta)$
$\overleftarrow{\Diamond}_I(\beta)$	$\equiv$	$\mathsf{S}_I(\top, \beta)$
$\Box_I(\beta)$	$\equiv$	$\mathsf{R}_I(\bot, \beta)$
$\overleftarrow{\Box}_I(\beta)$	$\equiv$	$\mathsf{T}_I(\bot, \beta)$
$\widetilde{\bigcirc}(\beta)$	$\equiv$	$\mathsf{U}_{(0,+\infty)}(\beta, \top) \vee (\neg\beta \wedge \mathsf{R}_{(0,+\infty)}(\beta, \bot))$
$\overleftarrow{\widetilde{\bigcirc}}(\beta)$	$\equiv$	$\mathsf{S}_{(0,+\infty)}(\beta, \top) \vee (\neg\beta \wedge \mathsf{T}_{(0,+\infty)}(\beta, \bot))$
$\bigcirc(\beta)$	$\equiv$	$\beta \wedge \widetilde{\bigcirc}(\beta)$
$\overleftarrow{\bigcirc}(\beta)$	$\equiv$	$\beta \wedge \overleftarrow{\widetilde{\bigcirc}}(\beta)$
$\triangle(\beta_1, \beta_2)$	$\equiv$	$\begin{cases} \overleftarrow{\widetilde{\bigcirc}}(\beta_1) \wedge \left(\beta_2 \vee \widetilde{\bigcirc}(\beta_2)\right) & \text{if } \mathbb{T} = \mathbb{R}_{\geq 0} \\ \overleftarrow{\Diamond}_{=1}(\beta_1) \wedge \Diamond_{[0,1]}(\beta_2) & \text{if } \mathbb{T} = \mathbb{N} \end{cases}$
$\blacktriangle(\beta_1, \beta_2)$	$\equiv$	$\begin{cases} \beta_1 \wedge \Diamond_{=\delta}(\beta_2) & \text{if } \mathbb{T} = \mathbb{R}_{\geq 0} \\ \beta_1 \wedge \Diamond_{=1}(\beta_2) & \text{if } \mathbb{T} = \mathbb{N} \end{cases}$

Table 1. MTL derived temporal operators

Derived Temporal Operators. It is useful to introduce a number of derived temporal operators, to be used as shorthands in writing specification formulas. Those used in this paper are listed in Table 1 ($\delta \in \mathbb{R}_{>0}$ is a parameter used in the discretization techniques, discussed shortly).

We describe informally the meaning of such derived operators, focusing on future ones (the meaning of the corresponding past operators is easily derivable). $\Diamond_I(\beta)$ means that β happens within time interval I in the future. $\Box_I(\beta)$ means that β holds throughout the whole interval I in the future. $\widetilde{\bigcirc}(\beta)$ denotes that β holds throughout some non-empty interval in the strict future; in other words, if t is the current instant, there exists some $t' > t$ such that β holds over (t, t'). Similarly, $\bigcirc(\beta)$ denotes that β holds throughout some non-empty interval which includes the current instant, i.e., over some $[t, t')$. Then, $\triangle(\beta_1, \beta_2)$ describes a switch from condition β_1 to condition β_2, without specifying which value holds at the current instant. On the other hand, $\blacktriangle(\beta_1, \beta_2)$ describes a switch from condition β_1 to condition β_2 such that β_1 holds at the current instant; more precisely if $\triangle(\beta_1, \beta_2)$ holds at some instant t, $\blacktriangle(\beta_1, \beta_2)$ holds over $(t - \delta, t)$. In addition, for an item it we introduce the shorthand $\triangle(\mathsf{it}, v^-, v^+)$ for $\triangle(\mathsf{it} = v^-, \mathsf{it} = v^+)$. A similar abbreviation is assumed for $\blacktriangle(\mathsf{it}, v^-, v^+)$. Finally, we use $\mathrm{Alw}(\phi)$ to denote $\phi \wedge \Box_{(0,+\infty)}(\phi) \wedge \overleftarrow{\Box}_{(0,+\infty)}(\phi)$. Since $b \models_{\mathbb{T}} \mathrm{Alw}(\phi)$ iff $b \models_{\mathbb{T}} \phi$, for any behavior b, $\mathrm{Alw}(\phi)$ can be expressed without nesting if ϕ is flat, through the global satisfiability semantics introduced beforehand.

2.3 Operational Model: Timed Automata

We introduce a variant of TA which differs from the classical definitions (e.g., [1]) in that it recognizes behaviors, rather than timed words [2, 15]. Correspondingly, input symbols are associated with locations rather than with transitions. Also, we introduce the following simplifications that are known to be without loss of generality: we do not define location clock invariants (also called staying conditions) and use transition guards only, and we forbid self-loop transitions.

We introduce one additional variant which does impact expressiveness, namely clock constraints do not distinguish between different transition edges, that is between transitions occurring right- and left-continuously. This restriction is motivated by our ultimate goal of *discretizing* TA: as it will be explained later, such distinctions would inevitably be lost in the discretization process, hence we give them up already.

Finally, for the sake of simplicity, we do not consider acceptance conditions, that is let us assume that all states are accepting. Notice that introducing acceptance conditions (e.g., Büchi, Muller, etc.) in the formalization would be routine.

TA syntax. For a set C of clock variables, the set $\Phi(C)$ of *clock constraints* ξ is defined inductively by

$$\xi ::= c < k \mid c \geq k \mid \xi_1 \wedge \xi_2 \mid \xi_1 \vee \xi_2$$

where c is a clock in C and k is a constant in $\mathbb{Q}_{\geq 0}$.

A *timed automaton* A is a tuple $\langle \Sigma, S, S_0, \alpha, C, E \rangle$, where:

- Σ is a finite (input) alphabet,
- S is a finite set of locations,
- $S_0 \subseteq S$ is a finite set of initial locations,
- $\alpha : S \to 2^{\Sigma}$ is a location labeling function that assigns to each location $s \in S$ a set $\alpha(s)$ of propositions,
- C is a finite set of clocks, and
- $E \subseteq S \times S \times 2^{C} \times \Phi(C)$ is a set of transitions. An edge $\langle s, s', \Lambda, \xi \rangle$ represents a transition from state s to state $s' \neq s$; the set $\Lambda \subseteq C$ identifies the clocks to be reset with this transition, and ξ is a clock constraint over C.

TA semantics. In defining the semantics of TA over behaviors we deviate from the standard presentation (e.g., [2, 15]) in that we do not represent TA as acceptors of behaviors over the input alphabet Σ, but rather as acceptors of behaviors representing what are usually called *runs* of the automaton. In other words, we introduce automata as acceptors of behaviors over the items st and in representing, respectively, the current location and the current input symbol, as well as propositions $\{\mathsf{rs}_c \mid c \in C\}$ representing the clock reset status. This departure from more traditional presentations is justified by the fact that we intend to provide an MTL axiomatic description of TA runs — rather than accepted languages, which would be impossible for a well-known expressiveness gap [12] — hence we define the semantics of automata over this "extended" state from the beginning.

Here we sketch an informal description of the semantics. Initially, all clocks are reset and the automaton is in state $s_0 \in S_0$. At any given time t, when the automaton is

in some state s, it can take nondeterministically a transition $\langle s, s', \Lambda, \xi \rangle$ to some other state s', only if the last time (before t) each clock has been reset is compatible with constraint ξ. If the transition is taken, all clocks in Λ are reset, whereas all the other clocks keep on running. Finally, as long as the automaton is in any state s, the input has to satisfy the location labeling function $\alpha(s)$, namely the current input corresponds to exactly one of the propositions in $\alpha(s)$.

A timed automaton $A = \langle \Sigma, S, S_0, \alpha, C, E \rangle$ is interpreted over behaviors over items $\mathsf{st} : S, \mathsf{in} : \Sigma$ and propositions $R = \{\mathsf{rs}_c \mid c \in C\}$. At any instant of time t, $\mathsf{st} = s$ means that the automaton is in state s, $\mathsf{in} = \sigma$ means that the input symbol is σ, and rs_c keeps track of resets of clock c (we model such resets through switches, from false to true or *vice versa*, of rs_c). Let b be such a behavior, and let t be one of its transition points. Satisfaction of clock constraints at t is defined as follows:

$b(t) \models c < k$ iff either $b^-(t) \models \mathsf{rs}_c$ and there exists a $t - k < t' < t$ s.t. $b(t') \not\models \mathsf{rs}_c$;
or $b^-(t) \not\models \mathsf{rs}_c$ and there exists a $t - k < t' < t$ s.t. $b(t') \models \mathsf{rs}_c$

$b(t) \models c \geq k$ iff either $b^-(t) \models \mathsf{rs}_c$ and for all $t - k < t' < t : b'(t) \models \mathsf{rs}_c$;
or $b^-(t) \not\models \mathsf{rs}_c$ and for all $t - k < t' < t : b(t') \not\models \mathsf{rs}_c$

Notice that this corresponds to looking for the previous time the proposition rs_c switched (from false to true or from true to false) and counting time since then. This requires a little "hack" in the definition of the semantics: namely, a first start reset of all clocks is issued before the "real" run begins; this is represented by time instant t_{start} in the formal semantics below.

Formally, a behavior b over $\mathsf{st} : S, \mathsf{in} : \Sigma, R$ (with $b : \mathbb{R}_{\geq 0} \to S \times \Sigma \times 2^R$) is a *run* of the automaton A, and we write $b \models_{\mathbb{R}_{\geq 0}} A$, iff:

- $b(0) = \langle s_0, \sigma, \bigcup_{c \in C} \{\mathsf{rs}_c\} \rangle$ and $\sigma \in \alpha(s_0)$ for some $s_0 \in S_0$;
- there exists a transition instant $t_{\mathrm{start}} > 0$[3] such that: $b(t)|_{\mathsf{st}} = s_0$ and $b(t)|_R = R$ for all $0 \leq t \leq t_{\mathrm{start}}$, $b^-(t_{\mathrm{start}}) = \langle s_0, \sigma^-, \rho^- \rangle$ and $b^+(t_{\mathrm{start}}) = \langle s^+, \sigma^+, \rho^+ \rangle$ with $\rho^- = R$ and $\rho^+ = \emptyset$;
- for all $t \in \mathbb{R}_{\geq 0}$: $b(t)|_{\mathsf{in}} \in \alpha(b(t)|_{\mathsf{st}})$;
- for all transition instants $t > t_{\mathrm{start}}$ of $b|_{\mathsf{st}}$ or $b|_R$ such that $b^-(t) = \langle s^-, \sigma^-, \rho^- \rangle$ and $b^+(t) = \langle s^+, \sigma^+, \rho^+ \rangle$, it is: $\langle s^-, s^+, \Lambda, \xi \rangle \in E$, $\sigma^- \in \alpha(s^-)$, $\sigma^+ \in \alpha(s^+)$, $\rho = \bigcup_{c \in \Lambda} \{\mathsf{rs}_c\}$, $\rho^+ = \rho^- \triangle \rho = (\rho^- \setminus \rho) \cup (\rho \setminus \rho^-)$, and $b(t) \models \xi$.

2.4 Discrete-Time Approximations of Continuous-Time Specifications

This section concisely summarizes the fundamental results from [8] that are needed in the remainder of the paper, and provides some intuition about how they can be applied to the discretization problem.

The technique of [8] is based on two approximation functions for MTL formulas, called under- and over-approximation. The under-approximation function $\Omega_\delta(\cdot)$ maps dense-time MTL formulas to discrete-time formulas such that the non-validity of the latter implies the non-validity of the former, over behaviors in $\mathcal{B}^\delta_\chi$. The over-approximation function $\mathrm{O}_\delta(\cdot)$ maps dense-time MTL formulas to discrete-time MTL

[3] In the following, we will assume that $t_{\mathrm{start}} \in (\delta, 2\delta)$ for the discretization parameter $\delta > 0$.

formulas such that the validity of the latter implies the validity of the former, over behaviors in $\mathcal{B}^{\delta}_{\chi}$. We have the following fundamental verification result, which provides a justification for the TA verification technique discussed in this paper.

Proposition 1 (Approximations [8]). *For any MTL formulas ϕ_1, ϕ_2, and for any $\delta \in \mathcal{D}_{\phi_1,\phi_2}$: (1) if $\mathrm{Alw}(\Omega_\delta(\phi_1)) \Rightarrow \mathrm{Alw}(\mathrm{O}_\delta(\phi_2))$ is $\mathbb{N}$-valid, then $\mathrm{Alw}(\phi_1) \Rightarrow \mathrm{Alw}(\phi_2)$ is χ^δ-valid; and (2) if $\mathrm{Alw}(\mathrm{O}_\delta(\phi_1)) \Rightarrow \mathrm{Alw}(\Omega_\delta(\phi_2))$ is not $\mathbb{N}$-valid, then $\mathrm{Alw}(\phi_1) \Rightarrow \mathrm{Alw}(\phi_2)$ is not χ^δ-valid.*

Discussion. Proposition 1 suggests a verification technique which builds two formulas through a suitable composition of over- and under-approximations of the system description and the putative properties, and it infers the validity of the properties from the results of a discrete-time validity checking. The technique is incomplete as, in particular, when approximation (1) is not valid and approximation (2) is valid nothing can be inferred about the validity of the property in the original system over dense time.

It is important to notice that equivalent dense-time formulas can yield dramatically different — in terms of usefulness — approximated discrete-time formulas. For instance, consider dense-time MTL formula $\theta_1 = \Box_{[0,2\delta]}(\mathsf{p})$ which, under the global satisfiability semantics, says that p is *always* true. Its under-approximation is $\Omega_\delta(\theta_1) = \Box_{\emptyset}(\mathsf{p})$ which holds for any discrete-time behavior! Thus, we have an under-approximation which is likely too coarse, as it basically adds no information to the discrete-time representation. So, if we build formula (1) from Proposition 1 with $\Omega_\delta(\theta_1)$ in it, it is likely that the antecedent will be trivially satisfiable (because $\Omega_\delta(\theta_1)$ introduces no constraint) and hence formula (1) will be non-valid, yielding no information to the verification process. If, however, we modify θ_1 into the *equivalent* $\theta_1' = \mathsf{p} \wedge \theta_1$ we get an under-approximation which can be written simply as $\Omega_\delta(\theta_1') = \mathsf{p}$, which correctly entails that p is always true over discrete-time as well. This is likely a much better approximation, one which better preserves the original "meaning" of θ_1.

3 Formalizing Timed Automata in MTL

Consider a TA $A = \langle \Sigma, S, S_0, \alpha, C, E \rangle$; this section introduces an MTL formalization of the runs of A over non-Berkeley behaviors, for some $\delta > 0$. In other words, this section provides a set of formulas $\phi_1, \ldots, \phi_6$ such that, for all non-Berkeley behaviors b, $b \models A$ iff $b \models \phi_j$ for all $j = 1, \ldots, 6$.

Clock constraints. Given a clock constraint ξ, we represent by $\Xi(\xi)$ an MTL formula such that $b(t) \models \xi$ iff $b(t) \models \Xi(\xi)$ at all transition points t. $\Xi(\xi)$ can be defined inductively as:

$$
\begin{array}{lcl}
\Xi(c < k) & \equiv & \overleftarrow{\widetilde{\bigcirc}}(\mathsf{rs}_c) \wedge \overleftarrow{\Diamond}_{(0,k)}(\neg \mathsf{rs}_c) \ \vee \ \overleftarrow{\widetilde{\bigcirc}}(\neg \mathsf{rs}_c) \wedge \overleftarrow{\Diamond}_{(0,k)}(\mathsf{rs}_c) \\
\Xi(c \geq k) & \equiv & \overleftarrow{\widetilde{\bigcirc}}(\mathsf{rs}_c) \wedge \overleftarrow{\Box}_{(0,k)}(\mathsf{rs}_c) \ \vee \ \overleftarrow{\widetilde{\bigcirc}}(\neg \mathsf{rs}_c) \wedge \overleftarrow{\Box}_{(0,k)}(\neg \mathsf{rs}_c) \\
\Xi(\xi_1 \wedge \xi_2) & \equiv & \Xi(\xi_1) \wedge \Xi(\xi_1) \\
\Xi(\xi_1 \vee \xi_2) & \equiv & \Xi(\xi_1) \vee \Xi(\xi_1)
\end{array}
$$

Essentially, Ξ translates guard ξ by comparing the current time to the last time a reset for the clock c happened, where a reset is represented by a switching of item rs_c. Notice that, to compute the approximations of the clock-constraint formulas, every constant k used in the definition of the TA must be an integral multiple of δ.

Necessary conditions for state change. Let us state the necessary conditions that characterize a state change. For any pair of states $s_i, s_j \in S$ such that there are K transitions $\langle s_i, s_j, \Lambda^k, \xi^k \rangle \in E$ for all $1 \leq k \leq K$, we introduce the following axiom:

$$\triangle(\mathsf{st}, s_i, s_j) \quad \Rightarrow \quad \bigvee_k \left(\Xi(\xi^k) \wedge \bigwedge_{c \in \Lambda^k} \Big(\triangle(\neg \mathsf{rs}_c, \mathsf{rs}_c) \vee \triangle(\mathsf{rs}_c, \neg \mathsf{rs}_c) \Big) \right) \tag{1}$$

Also, we introduce an axiom asserting that, for any pair of states $s_i \neq s_j \in S$ such that $\langle s_i, s_j, \Lambda, \xi \rangle \notin E$ for any Λ, ξ (i.e., for any pair of states that are not connected by any edge), there cannot be a transition from s_i to s_j:

$$\neg \triangle(\mathsf{st}, s_i, s_j) \tag{2}$$

Sufficient conditions for state change. There are multiple sufficient conditions for state changes; basically, they account for reactions to reading input symbols and resetting clocks. Let us consider input first: the staying condition in every state must be satisfied always, so for all $s \in S$ the following axiom is added:

$$\mathsf{st} = s \quad \Rightarrow \quad \mathsf{in} \in \alpha(s) \tag{3}$$

Then, for each reset of a clock $c \in C$, for all $1 \leq k \leq K$ such that $\langle s_i^k, s_j^k, \Lambda^k, \xi^k \rangle \in E$ is an edge such that $c \in \Lambda^k$ (i.e., on which c is reset), the following axiom is introduced (a similar one for the transition of rs_c from true to false is also included):

$$\triangle(\neg \mathsf{rs}_c, \mathsf{rs}_c) \quad \Rightarrow \quad \bigvee_k \triangle\big(\mathsf{st}, s_i^k, s_j^k\big) \tag{4}$$

Initialization and liveness condition. The axiomatization is completed by including formulas describing initialization and liveness conditions. The following axiom, describing system initialization, is only evaluated at 0:

$$\text{at } 0\text{:} \quad \bigwedge_{c \in C} \mathsf{rs}_c \wedge \Diamond_{[0,2\delta]} \left(\bigwedge_{c \in C} \neg \mathsf{rs}_c \right) \wedge \bigvee_{s_0 \in S_0} \bigcirc(\mathsf{st} = s_0) \tag{5}$$

Finally, we introduce a "liveness" condition stating that the automaton must eventually move out of every state. Thus, for every state $s \in S$, if $S'_s \subset S$ is the set of states that are directly reachable from s through a single transition the following axiom asserts that, if the automaton is in s, it must eventually move to a state in S'_s:

$$\mathsf{st} = s \;\Rightarrow\; \Diamond \left(\bigvee_{s' \in S'_s} \mathsf{st} = s' \right) \tag{6}$$

Since axiom (6) does not mandate that only some particular states must be traversed infinitely often, this corresponds to the condition that all states are accepting *à la* Büchi.

The next proposition states the axiomatization correctness (see [9] for details).

Proposition 2 (MTL TA Axiomatization). *Let $A = \langle \Sigma, S, S_0, \alpha, C, E \rangle$ be a timed automaton, $\phi_1^A, \ldots, \phi_6^A$ be formulas (1–6) for TA A, and let $b \in \mathcal{B}_\chi^\delta$ be any non-Berkeley behavior over items $\mathsf{st} : S, \mathsf{in} : \Sigma$ and propositions in R. Then $b \models A$ if and only if $b \models \bigwedge_{1 \leq j \leq 6} \phi_j^A$.*

4 Discrete-Time Approximations of Timed Automata

While formulas (1–6) correctly formalize the behavior of TA as defined in Section 2.3, they yield approximations of little use for verification purposes, since they are very likely to produce inconclusive results due to the incompleteness of the technique. To avoid such problems, instead of computing approximations directly from axioms (1–6), we introduce new formulas, which are equivalent to (1–6) over non-Berkeley behaviors, but whose form yields better approximations.

In the rest of this section we first compute the under-approximation of formulas (1–6) (Section 4.1), and then their over-approximation (Section 4.2).

4.1 Under-Approximation

As mentioned above, the form of some of the axioms (1–6) produces under-approximations that are ill-suited to perform verification through the discretization technique of [8], due to the inherent incompleteness of the latter.

While the details underlying this issue are outside the scope of this article (and can be found in [9]), let us hint at some of the problems that arise from the under-approximation of formulas (1–6). First, it can be shown that, in general, $\Omega_\delta\left(\neg\triangle(\beta_1, \beta_2)\right) \neq \neg\Omega_\delta\left(\triangle(\beta_1, \beta_2)\right)$; since subformulas of the form $\triangle(\mathsf{st} = s_i, \mathsf{st} = s_j)$ are used in (1–6) to describe state transitions, there are discrete-time behaviors where such a transition both occurs and does not occur, i.e., $\Omega_\delta\left(\triangle(\mathsf{st} = s_i, \mathsf{st} = s_j)\right)$ and $\Omega_\delta\left(\neg\triangle(\mathsf{st} = s_i, \mathsf{st} = s_j)\right)$ are both true, which is an approximation too coarse to be useful. Second, $\Omega_\delta\left(\neg\triangle(\beta_1, \beta_2)\right)$ is a very weak formula, in that it can be shown to be true, in particular, whenever β_1 or β_2 are false; then, antecedents $\triangle(\mathsf{st} = s_i, \mathsf{st} = s_j)$ in (1–6) are trivially true because it can never be that both $\mathsf{st} = s_i$ and $\mathsf{st} = s_j$ when $s_i \neq s_j$.

To obtain better approximations, in the new axiomatization every occurrence of $\triangle(\beta_1, \beta_2)$ is replaced with $\blacktriangle(\beta_1, \beta_2)$. This entails that formulas $\Xi(\xi)$ representing clock constraints must also be changed in $\overrightarrow{\Xi}(\xi)$, where $\overrightarrow{\Xi}$ is defined below. Hence, formulas (1–2),(4) become:

$$\blacktriangle(\mathsf{st}, s_i, s_j) \quad \Rightarrow \quad \bigvee_k \overrightarrow{\Xi}(\xi^k) \ \wedge \ \bigwedge_{c \in \Lambda^k} \Big(\blacktriangle(\neg\mathsf{rs}_c, \mathsf{rs}_c) \ \vee \ \blacktriangle(\mathsf{rs}_c, \neg\mathsf{rs}_c) \Big) \tag{7}$$

$$\neg\blacktriangle(\mathsf{st}, s_i, s_j) \tag{8}$$

$$\blacktriangle(\neg \mathsf{rs}_c, \mathsf{rs}_c) \quad \Rightarrow \quad \bigvee_k \blacktriangle\big(\mathsf{st}, s_i^k, s_j^k\big) \tag{9}$$

where $\overrightarrow{\Xi}$ is defined as follows:

$$\begin{array}{l} \overrightarrow{\Xi}\,(c < k) \equiv \mathsf{rs}_c \wedge \overleftarrow{\Diamond}_{(0,k)}(\neg \mathsf{rs}_c) \quad \vee \quad \neg \mathsf{rs}_c \wedge \overleftarrow{\Diamond}_{(0,k)}(\mathsf{rs}_c) \\ \overrightarrow{\Xi}\,(c \geq k) \equiv \mathsf{rs}_c \wedge \overleftarrow{\Box}_{(0,k-\delta)}(\mathsf{rs}_c) \quad \vee \quad \neg \mathsf{rs}_c \wedge \overleftarrow{\Box}_{(0,k-\delta)}(\neg \mathsf{rs}_c) \end{array}$$

It can be shown that, given a non-Berkeley behavior $b \in \mathcal{B}_\chi^\delta$, $b \models (1)$ iff $b \models (7)$, $b \models (2)$ iff $b \models (8)$ and $b \models (4)$ iff $b \models (9)$.

Proof. Let us first show that (1) implies (7), so let t be the current instant, assume that (1) and the antecedent $\blacktriangle(\mathsf{st}, s_i, s_j)$ of (7) hold: we establish that the consequent of (7) holds. $\blacktriangle(\mathsf{st}, s_i, s_j)$ means that $\mathsf{st} = s_i$ at t and $\mathsf{st} = s_j \neq s_i$ at $t + \delta$; hence there must be a transition instant t' of item st somewhere in $[t, t + \delta]$. Then (1) evaluated at t' entails that t' is a transition instant for some propositions $\mathsf{rs}_c|_{c \in \Lambda^k}$ as well. Let $d \in C$ be anyone of such clocks and assume that $\triangle(\mathsf{rs}_d, \neg \mathsf{rs}_d)$ holds at t'. Let us first assume $t' \in (t, t + \delta)$; correspondingly, from the non-Berkeleyness assumption, rs_d holds over $[t, t')$ and $\neg \mathsf{rs}_d$ holds over $(t', t + \delta]$. In particular, rs_d holds at t and $\neg \mathsf{rs}_d$ holds at $t + \delta$, so $\blacktriangle(\mathsf{rs}_d, \neg \mathsf{rs}_d)$ holds at t. Otherwise, let $t' = t$, so st changes its value left-continuously at t. Then, again from (1) and the non-Berkeleyness assumption, rs_d also changes its value left-continuously, so rs_d holds at t and $\neg \mathsf{rs}_d$ holds at $t+\delta$. Finally, if $t' = t + \delta$, st changes its value right-continuously at t', so rs_d also changes its value right-continuously, so rs_d holds at t and $\neg \mathsf{rs}_d$ holds at $t+\delta$. In all, since d is generic, and the same reasoning applies for the converse transition $\triangle(\neg \mathsf{rs}_d, \mathsf{rs}_d)$, we have established that $\bigwedge_{c \in \Lambda^k} (\blacktriangle(\neg \mathsf{rs}_c, \mathsf{rs}_c) \vee \blacktriangle(\mathsf{rs}_c, \neg \mathsf{rs}_c))$ holds at t.
Next, let us establish $\overrightarrow{\Xi}(\xi^k)$ from $\Xi(\xi^k)$. Let us first consider some $\Xi\,(d < k)$ such that $\overleftarrow{\bigcirc}(\mathsf{rs}_d) \wedge \overleftarrow{\Diamond}_{(0,k)}(\neg \mathsf{rs}_d)$ at t'. So, let $t'' \in (t' - k, t')$ be the largest instant with a transition from $\neg \mathsf{rs}_d$ to rs_d. Note that it must actually be $t'' \in (t' - k, t]$ because $t' - t \leq \delta$ and the non-Berkeleyness assumption. If $t'' \in (t' - k, t) \subseteq (t - k, t)$ then $\mathsf{rs}_d \wedge \overleftarrow{\Diamond}_{(0,k)}(\neg \mathsf{rs}_d)$ holds at t, hence $\overrightarrow{\Xi}\,(d < k)$ is established. If $t'' = t$ then rs_d switches to true right-continuously at t, so $\mathsf{rs}_d \wedge \overleftarrow{\bigcirc}(\neg \mathsf{rs}_d)$ at t which also entails $\overrightarrow{\Xi}\,(d < k)$. The same reasoning applies if $\overleftarrow{\bigcirc}(\neg \mathsf{rs}_c) \wedge \overleftarrow{\Diamond}_{(0,k)}(\mathsf{rs}_c)$ holds at t'. Finally, consider some $\Xi\,(d \geq k)$ such that $\overleftarrow{\bigcirc}(\mathsf{rs}_d) \wedge \overleftarrow{\Box}_{(0,k)}(\mathsf{rs}_d)$ holds at t, thus rs_d holds over $(t - k, t)$. From $t \leq t' + \delta$ we have $t' + \delta - k \geq t + k$ so $(t' - k + \delta, t') \subseteq (t - k, t)$, which shows that $\overleftarrow{\Box}_{(0,k-\delta)}(\mathsf{rs}_d)$ holds at t'. The usual reasoning about transition edges would allow us to establish that also rs_d holds at t'. Since the same reasoning applies if $\overleftarrow{\bigcirc}(\neg \mathsf{rs}_d) \wedge \overleftarrow{\Box}_{(0,k)}(\neg \mathsf{rs}_d)$, we have established that $\overrightarrow{\Xi}(d \geq k)$ holds at t'. Since d is generic, we have that $\overrightarrow{\Xi}(\xi^k)$ holds at t'.

Let us now prove (7) implies (1), so let t be the current instant, assume that (7) and the antecedent $\triangle(\mathsf{st}, s_i, s_j)$ of (1) hold: we establish that the consequent of (1) holds. So, there is a transition of st from s_i to $s_j \neq s_i$ at t; from the non-Berkeleyness assumption we have that $\mathsf{st} = s_i$ and $\mathsf{st} = s_j$ hold over $[t - \delta, t)$ and $(t, t + \delta]$, respectively. If

the transition of st is left-continuous (i.e., $\mathsf{st} = s_i$ holds at t), consider (7) at t, where the antecedent holds. So, $\overrightarrow{\Xi}(\xi^k) \wedge \bigwedge_{c \in \Lambda^k} (\blacktriangle(\neg \mathsf{rs}_c, \mathsf{rs}_c) \vee \blacktriangle(\mathsf{rs}_c, \neg \mathsf{rs}_c))$ holds at t for some k. Let $d \in \Lambda^k$ be such that $\blacktriangle(\neg \mathsf{rs}_d, \mathsf{rs}_d)$ holds, that is $\neg \mathsf{rs}_d$ holds at t and rs_d holds at $t + \delta$. This entails that there exists a transition point $t' \in [t, t + \delta]$ of rs_d. However, t is already a transition point, thus it must be $t' = t$; this shows $\triangle(\neg \mathsf{rs}_d, \mathsf{rs}_d)$ at d. Recall that d is generic, and the same reasoning applies for the converse transition from rs_d to $\neg \mathsf{rs}_d$. If, instead, the transition of st is right-continuous (i.e., $\mathsf{st} = s_j$ holds at t), we consider (7) at $t - \delta$ and perform a similar reasoning. All in all, we have established that $\bigwedge_{c \in \Lambda^k} (\triangle(\neg \mathsf{rs}_c, \mathsf{rs}_c) \vee \triangle(\mathsf{rs}_c, \neg \mathsf{rs}_c))$ holds at t.
The clock constraint formula $\Xi(\xi^k)$ can also be proved along the same lines. For instance, assume that the transition of st at t is left-continuous and $\overleftarrow{\bigcirc}(\mathsf{rs}_d)$ holds at t for some $d \in C$, and consider a constraint $\overrightarrow{\Xi}(d < k)$ at t. We have that $\overleftarrow{\Diamond}_{(0,k)}(\neg \mathsf{rs}_d)$ must holds at t, which establishes that $\Xi(d < k)$ holds at t. Similar reasonings apply to the other cases.

See [9] for proofs of the other equivalences. □

Then, the axiomatization of TA given by formulas (7–9),(3),(5–6) yields the following under-approximations.

$$\Omega_\delta\left((7)\right) \quad \equiv \quad \blacktriangle(\mathsf{st}, s_i, s_j) \quad \Rightarrow \quad \bigvee_k \Omega_\delta\left(\overrightarrow{\Xi}(\xi^k)\right) \wedge \bigwedge_{c \in \Lambda^k} \begin{pmatrix} \blacktriangle(\neg \mathsf{rs}_c, \mathsf{rs}_c) \\ \vee \\ \blacktriangle(\mathsf{rs}_c, \neg \mathsf{rs}_c) \end{pmatrix} \tag{10}$$

where:

$$\begin{aligned} \Omega_\delta\left(\overrightarrow{\Xi}\,(c < k)\right) \quad &\equiv \quad \mathsf{rs}_c \wedge \overleftarrow{\Diamond}_{[0,k/\delta]}(\neg \mathsf{rs}_c) \quad \vee \quad \neg \mathsf{rs}_c \wedge \overleftarrow{\Diamond}_{[0,k/\delta]}(\mathsf{rs}_c) \\ \Omega_\delta\left(\overrightarrow{\Xi}\,(c \geq k)\right) \quad &\equiv \quad \mathsf{rs}_c \wedge \overleftarrow{\Box}_{[1,k/\delta-2]}(\mathsf{rs}_c) \quad \vee \quad \neg \mathsf{rs}_c \wedge \overleftarrow{\Box}_{[0,k/\delta-2]}(\neg \mathsf{rs}_c) \end{aligned}$$

In addition the following can be proved to hold:

$$\Omega_\delta\left((8)\right) \quad \equiv \quad \neg\blacktriangle(\mathsf{st}, s_i, s_j) \tag{11}$$

$$\Omega_\delta\left((9)\right) \quad \equiv \quad \blacktriangle(\neg \mathsf{rs}_c, \mathsf{rs}_c) \Rightarrow \bigvee_k \blacktriangle\left(\mathsf{st}, s_i^k, s_j^k\right) \tag{12}$$

$$\Omega_\delta\left((3)\right) \quad \equiv \quad (3) \tag{13}$$

$$\text{at } 0\text{:} \quad \Omega_\delta\left((5)\right) \quad \equiv \quad \bigwedge_{c \in C} \mathsf{rs}_c \wedge \Diamond_{[1,2]} \left(\bigwedge_{c \in C} \neg \mathsf{rs}_c \right) \wedge \bigvee_{s_0 \in S_0} \mathsf{st} = s_0 \tag{14}$$

$$\Omega_\delta\left((6)\right) \quad \equiv \quad (6) \tag{15}$$

4.2 Over-Approximation

While the over-approximation of axioms (1–6) poses less problems than their under-approximation, it must nonetheless be carried out very carefully, and some modifications to the axioms are in order in this case, too. Notice that the following equalities hold:

- $\mathrm{O}_\delta\left(\widetilde{\bigcirc}(\beta)\right) = \Box_{[0,1]}(\beta)$.
- $\mathrm{O}_\delta\left(\Diamond_{[0,2\delta]}(\beta)\right) = \Diamond_{=1}(\beta)$.
- $\mathrm{O}_\delta\left(\bigcirc(\beta)\right) = \Box_{[0,1]}(\beta)$.
- $\mathrm{O}_\delta\left(\overleftarrow{\widetilde{\bigcirc}}(\beta)\right) = \mathrm{O}_\delta\left(\overleftarrow{\bigcirc}(\beta)\right) = \overleftarrow{\Box}_{[0,1]}(\beta)$.
- $\mathrm{O}_\delta\left(\neg\triangle(\beta_1,\beta_2)\right) = \neg(\triangle(\beta_1,\beta_2) \vee \blacktriangle(\beta_1,\beta_2))$ if β_1 and β_2 cannot hold at the same time (i.e., if $\neg(\beta_1 \wedge \beta_2)$).

The over-approximations of the clock constraints (i.e., $\mathrm{O}_\delta\left(\Xi(\xi)\right)$) pose little problems when ξ is of the form $c < k$; however, when ξ is of the form $c \geq k$, they yield formulas that are unsatisfiable if there are some transitions that reset c and whose guard is $c \geq k$. Hence, the definition of $\Xi(c \geq k)$ must be modified in the following way (which is equivalent to the previous formulation for non-Berkeley behaviors):

$$\Xi\left(c \geq k\right) \quad \equiv \quad \overleftarrow{\widetilde{\bigcirc}}(\mathsf{rs}_c) \wedge \overleftarrow{\Box}_{[\delta,k)}(\mathsf{rs}_c) \;\vee\; \overleftarrow{\widetilde{\bigcirc}}(\neg\mathsf{rs}_c) \wedge \overleftarrow{\Box}_{[\delta,k)}(\neg\mathsf{rs}_c)$$

Therefore, the over-approximations of the new clock constraints are the following:

$$\mathrm{O}_\delta\left(\Xi\left(c < k\right)\right) \equiv \overleftarrow{\Box}_{[0,1]}(\mathsf{rs}_c) \wedge \overleftarrow{\Diamond}_{[1,k/\delta-1]}(\neg\mathsf{rs}_c) \vee \overleftarrow{\Box}_{[0,1]}(\neg\mathsf{rs}_c) \wedge \overleftarrow{\Diamond}_{[1,k/\delta-1]}(\mathsf{rs}_c)$$
$$\mathrm{O}_\delta\left(\Xi\left(c \geq k\right)\right) \equiv \overleftarrow{\Box}_{[0,1]}(\mathsf{rs}_c) \wedge \overleftarrow{\Box}_{[0,k/\delta+1]}(\mathsf{rs}_c) \vee \overleftarrow{\Box}_{[0,1]}(\neg\mathsf{rs}_c) \wedge \overleftarrow{\Box}_{[0,k/\delta+1]}(\neg\mathsf{rs}_c)$$

The over-approximation of formula (1), instead, is very poor verification-wise, because subformulas of the form $\triangle(\beta_1,\beta_2)$ such that β_1,β_2 cannot hold at the same instant produce over-approximations that are unsatisfiable. The same problems arise with the over-approximation of formula (4).

However, similarly to what was done in Section 4.1, it is possible to rewrite axioms (1) and (4) so that they yield better over-approximations. The two following formulas are equivalent, over non-Berkeley behaviors, to (1) and (4), respectively (see [9] for equivalence proofs).

$$\triangle(\mathsf{st}, s_i, s_j) \;\Rightarrow\; \bigvee_k \left(\Xi(\xi^k) \wedge \bigwedge\nolimits_{c\in\Lambda^k} \left(\begin{array}{c} \overleftarrow{\widetilde{\bigcirc}}(\neg\mathsf{rs}_c) \wedge \Box_{=\delta}(\mathsf{st} = s_j \Rightarrow \mathsf{rs}_c) \\ \vee \\ \overleftarrow{\widetilde{\bigcirc}}(\mathsf{rs}_c) \wedge \Box_{=\delta}(\mathsf{st} = s_j \Rightarrow \neg\mathsf{rs}_c) \end{array} \right) \right) \tag{16}$$

$$\triangle(\neg\mathsf{rs}_c, \mathsf{rs}_c) \quad \Rightarrow \quad \bigvee_k \left(\overleftarrow{\widetilde{\bigcirc}}\big(\mathsf{st} = s_i^k\big) \wedge \Box_{=\delta}\big(\mathsf{rs}_c \Rightarrow \mathsf{st} = s_j^k\big) \right) \tag{17}$$

Intuitively, the equivalence of (1) and (16) can be proved noting that, if $\triangle(\mathsf{st}, s_i, s_j)$ holds at instant t in some non-Berkeley behavior b because a transition $\langle s_i, s_j, \Lambda^k, \xi^k\rangle \in E$ is taken, then, for the non-Berkeleyness of b it must be $\mathsf{st} = s_j$ throughout $(t, t+\delta]$. As a consequence of (16), for any $c \in \Lambda^k$, if $\overleftarrow{\widetilde{\bigcirc}}(\neg\mathsf{rs}_c)$ at t then rs_c holds at $t+\delta$. For the non-Berkeleyness of b it must be that $\widetilde{\bigcirc}(\mathsf{rs}_c)$ holds at t, hence $\triangle(\neg\mathsf{rs}_c, \mathsf{rs}_c)$ also holds

at t. The case $\widetilde{\bigcirc}(\neg \mathsf{rs}_c)$ at t is handled in the same way. Similar reasoning can be used to prove the equivalence of (4) and (17).

Finally, formulas (16–17),(2–3),(5–6) yield the following over-approximations, after performing some discrete-time simplifications:

$$\mathrm{O}_\delta\left((16)\right) \equiv$$
$$\blacktriangle(\mathsf{st}, s_i, s_j) \Rightarrow \bigvee_k \left(\begin{array}{c} \mathrm{O}_\delta\left(\Xi(\xi^k)\right) \wedge \\ \bigwedge_{c \in \Lambda^k} \left(\begin{array}{c} \overleftarrow{\Box}_{[0,1]}(\neg \mathsf{rs}_c) \wedge \Box_{[0,2]}(\mathsf{st} = s_j \Rightarrow \mathsf{rs}_c) \\ \vee \\ \overleftarrow{\Box}_{[0,1]}(\mathsf{rs}_c) \wedge \Box_{[0,2]}(\mathsf{st} = s_j \Rightarrow \neg \mathsf{rs}_c) \end{array} \right) \end{array} \right) \tag{18}$$

$$\mathrm{O}_\delta\left((17)\right) \quad \equiv \quad \blacktriangle(\neg \mathsf{rs}_c, \mathsf{rs}_c) \Rightarrow \bigvee_k \left(\overleftarrow{\Box}_{[0,1]}\big(\mathsf{st} = s_i^k\big) \wedge \Box_{[0,2]}\big(\mathsf{rs}_c \Rightarrow \mathsf{st} = s_j^k\big) \right) \tag{19}$$

$$\mathrm{O}_\delta\left((2)\right) \quad \equiv \quad \neg(\triangle(\mathsf{st}, s_i, s_j) \vee \blacktriangle(\mathsf{st}, s_i, s_j)) \tag{20}$$

$$\mathrm{O}_\delta\left((3)\right) \quad \equiv \quad (3) \tag{21}$$

$$\text{at } 0\text{:} \quad \mathrm{O}_\delta\left((5)\right) \quad \equiv \quad \bigwedge_{c \in C} \mathsf{rs}_c \wedge \Diamond_{=1} \left(\bigwedge_{c \in C} \neg \mathsf{rs}_c \right) \wedge \bigvee_{s_0 \in S_0} \Box_{[0,1]}(\mathsf{st} = s_0) \tag{22}$$

$$\mathrm{O}_\delta\left((6)\right) \quad \equiv \quad (6) \tag{23}$$

4.3 Summary

The following proposition, following from Propositions 1–2 and the results of the previous sections, summarizes the results of the discrete-time approximation formulas.

Proposition 3. *Let S be a real-time system described by TA $A = \langle \Sigma, S, S_0, \alpha, C, E \rangle$ and by a set of MTL specification formulas $\{\phi_j^{\mathsf{sys}}\}_j$ over items in $\mathcal{I}$ and propositions in $\mathcal{P}$. Also, let ϕ^{prop} be another MTL formula over items in $\mathcal{I} \cup \{\mathsf{st} : S, \mathsf{in} : \Sigma\}$ and propositions in $\mathcal{P} \cup R$. Then:*

– *if:*

$$\mathrm{Alw}\left(\phi^A_{(10)} \wedge \phi^A_{(11)} \wedge \phi^A_{(12)} \wedge \phi^A_{(13)} \wedge \phi^A_{(14)} \wedge \phi^A_{(15)} \wedge \bigwedge_j \Omega_\delta\left(\phi_j^{\mathsf{sys}}\right) \right) \Rightarrow \mathrm{Alw}(\mathrm{O}_\delta\left(\phi^{\mathsf{prop}}\right))$$

is $\mathbb{N}$-valid, then ϕ^{prop} is satisfied by all non-Berkeley runs $b \in \mathcal{B}^\delta_\chi$ of the system;

– *if:*

$$\mathrm{Alw}\left(\phi^A_{(18)} \wedge \phi^A_{(19)} \wedge \phi^A_{(20)} \wedge \phi^A_{(21)} \wedge \phi^A_{(22)} \wedge \phi^A_{(23)} \wedge \bigwedge_j \mathrm{O}_\delta\left(\phi^{\mathsf{sys}}_j\right)\right) \Rightarrow \mathrm{Alw}(\Omega_\delta\left(\phi^{\mathsf{prop}}\right))$$

is not $\mathbb{N}$*-valid, then* ϕ^{prop} *is false in some non-Berkeley run* $b \in \mathcal{B}^\delta_\chi$ *of the system.*

5 Implementation and Example

We implemented the verification technique of this paper as a plugin to the ℤot bounded satisfiability checker [16, 17]. The plugin provides a set of primitives by which the user can define the description of a TA, of a set of MTL axioms, and a set of MTL properties to be verified. The tool then automatically builds the two discrete-time approximation formulas of Proposition 3. These are checked for validity over time $\mathbb{N}$; the results of the validity check allows one to infer the validity of the original dense-time models, according to Proposition 3.

The verification process in TAℤot consists of three sequential phases. First, the discrete-time MTL formulas of Proposition 3 are built and are translated into a propositional satisfiability (SAT) problem. Second, the SAT instance is put into conjunctive normal form (CNF), a standard input format for SAT solvers. Third, the CNF formula is fed to a SAT solving engine (such as MiniSat, zChaff, or MiraXT).

5.1 A Communication Protocol Example

We demonstrate the practical feasibility of our verification techniques by means of an example, where we verify certain properties of the following communication protocol

Consider a server accepting requests from clients to perform a certain service (the exact nature of the service is irrelevant for our purposes). Initially, the server is *idle* in a passive open state. At any time, a client can initiate a protocol run; when this is the case, the server moves to a *try* state. Within T_1 time units, the state moves to a new s_1 state, characterizing the first request of the client for the service. The request can either terminate within T_2 time units, or time-out after T_2 time units have elapsed. When it terminates, it can do so either successfully (*ok*) or unsuccessfully (*ko*). In case of success, the protocol run is completed afterward, and the server goes back to being *idle*. In case of failure or time-out, the server moves to a new s_2 state for a second attempt. The second attempt is executed all similarly to the first one, with the only exception that the system goes back to the *idle* state afterward, regardless of the outcome (success, failure, or time-out). The timed automaton of Figure 1 models the protocol.[4]

We verified the following 5 properties of a single instance of the automaton:

[4] Since the definition of clock constraints forbids the introduction of exact constraints such as $A = T_2$, such constraints represent a shorthand for the valid clock constraint $T_2 \leq A < T + \delta$.

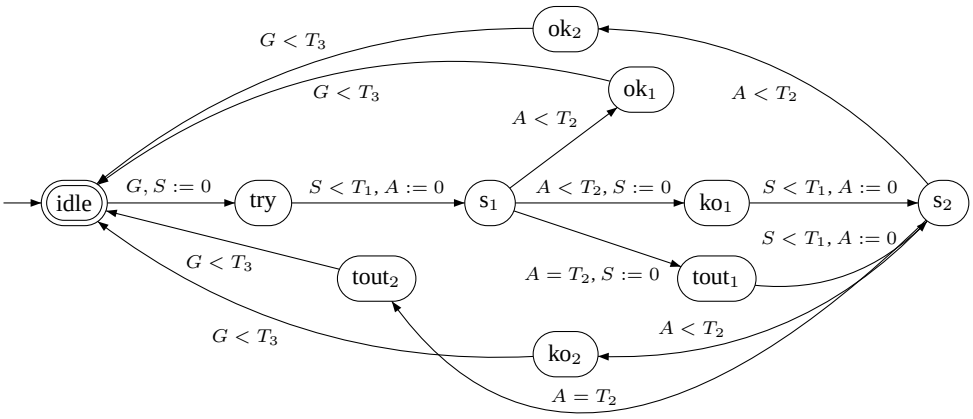

Fig. 1. Timed automaton modeling the communication protocol.

1. "If there is a success, the server goes back to idle without passing through error states."
$$\mathrm{ok}_1 \vee \mathrm{ok}_2 \quad \Rightarrow \quad \mathsf{U}(\mathrm{ko}_1 \vee \mathrm{ko}_2, \mathrm{idle})$$
2. "If there is a failure, the server goes back to idle without passing through success states."
$$\mathrm{ko}_1 \vee \mathrm{ko}_2 \quad \Rightarrow \quad \mathsf{U}(\mathrm{ok}_1 \vee \mathrm{ok}_2, \mathrm{idle})$$
This property is false, and in fact counterexamples are produced in the tests.
3. "A full run of the protocol executes in no more than T_3 time units."
$$\mathrm{try} \quad \Rightarrow \quad \Diamond_{(0,T_3)}(\mathrm{idle})$$
This property cannot be verified due to the incompleteness of the method: whether a run is completed in T_3/δ time instants depends sensibly on how the sampling is chosen. However, if we slightly weaken the property by changing T_3 into $T_3+\delta$ the method is successful in verifying the property. In the tables, the (verified) property — modified in this way — is labeled 3'.
4. "The first attempt of the protocol is initiated no later than $2T_1 + T_2 + \delta$ time units after the run has been initiated."
$$\mathrm{s}_1 \quad \Rightarrow \quad \overleftarrow{\Diamond}_{(0,2T_1+T_2+\delta)}(\mathrm{try})$$
5. "A run is terminated within T_3 time units after a successful outcome, without going through failure states."
$$\mathrm{ok}_1 \quad \Rightarrow \quad \mathsf{U}_{(0,T_3)}(\neg(\mathrm{ko}_1 \vee \mathrm{ko}_2), \mathrm{idle})$$

We also considered concurrent runs of $N_r \geq 2$ instances of the automaton, synchronized under the assumption that two parallel protocol runs that are initiated concurrently

either both terminate successfully, or both terminate unsuccessfully. This is formalized by the following MTL formula:

$$\begin{aligned}\forall 1 \leq i < j \leq N_r : \quad & \mathrm{try}^i \wedge \mathrm{try}^j \Rightarrow \\ & \mathsf{U}\big(\neg(\mathrm{tout_2}^i \vee \mathrm{ko_2}^i), \mathrm{ok_1}^i \vee \mathrm{ok_2}^i\big) \wedge \mathsf{U}\big(\neg(\mathrm{tout_2}^j \vee \mathrm{ko_2}^j), \mathrm{ok_1}^j \vee \mathrm{ok_2}^j\big) \\ & \qquad \vee \\ & \mathsf{U}\big(\neg(\mathrm{ok_1}^i \vee \mathrm{ok_2}^i), \mathrm{tout_2}^i \vee \mathrm{ko_2}^i\big) \wedge \mathsf{U}\big(\neg(\mathrm{ok_1}^j \vee \mathrm{ok_2}^j), \mathrm{tout_2}^j \vee \mathrm{ko_2}^j\big)\end{aligned}$$

Correspondingly, we introduce the following two properties to be verified in this concurrent system.

6. "If at some time one process succeeds and the other fails, then they have not begun the current run together."

$$\mathrm{ok_2}^A \wedge \mathrm{ko_2}^B \quad \Rightarrow \quad \mathsf{S}_{(0,T_3)}\big(\neg(\mathrm{try}^A \wedge \mathrm{try}^B), \mathrm{try}^A \vee \mathrm{try}^B\big)$$

7. "If at some time one process succeeds and the other failed recently, then they have not begun the current run together."

$$\mathrm{ok_2}^A \wedge \overleftarrow{\Diamond}_{(0,T_1)}\Big(\mathrm{ko_2}^B\Big) \quad \Rightarrow \quad \mathsf{S}_{(0,T_3)}\big(\neg(\mathrm{try}^A \wedge \mathrm{try}^B), \mathrm{try}^A \vee \mathrm{try}^B\big)$$

5.2 Experimental Evaluation

Tables 2 shows some results obtained in tests with TA$\mathbb{Z}$ot verifying the properties above. In all tests it is $\delta = 1$. For each test the table reports: the checked property; the number N_r of parallel protocol runs, according to which the discretizations are built; the values of other parameters in the model (i.e., T_1, T_2, T_3); the temporal bound k of the time domain (as $\mathbb{Z}$ot is a bounded satisfiability checker, it considers all the behaviors with period $\leq k$); the total amount of time and space (in MBytes) to perform each phase of the verification, namely formula building (FB), transformation into conjunctive normal form (CNF), and propositional satisfiability checking (SAT). The tests have been performed on a PC equipped with an AMD Athlon64 X2 Dual-Core Processor 4000+, 2 Gb of RAM, and Kubuntu GNU/Linux (kernel 2.6.22). TA$\mathbb{Z}$ot used GNU CLisp 2.41 and MiniSat 2.0 as SAT-solving engine.

The experiments clearly shows that the formula building time is usually negligible; the satisfiability checking time is also usually acceptably small, at least within the parameter range for the experiments we considered. On the contrary, the time to convert formulas in conjunctive normal form usually dominates in our tests. This indicates that there is significant room for practical scalability of our verification technique. In fact, from a computational complexity standpoint, the SAT phase is clearly the critical one, as it involves solving an NP-complete problem. On the other hand, the CNF routine has a quadratic running time.

Another straightforward optimization could be the implementation of the TA encoding directly in CNF, to bypass the `sat2cnf` routine. This can easily be done, because the structure of the formulas in the axiomatization is fixed. In conclusion, we can safely claim that the performances obtained in the tests are satisfactory in perspective, and they successfully demonstrate the practical feasibility of our verification technique.

PR.#	N_r	T_1, T_2, T_3	k	FB (time/mem)	CNF (time/mem)	SAT (time/mem)
1	1	3,6,18	30	0.1 min/115.6 Mb	4.0 min	0.3 min/90.6 Mb
2	1	3,6,18	30	0.1 min/230.6 Mb	8.0 min	0.6 min/180.9 Mb
3	1	3,6,18	30	0.2 min/246.3 Mb	9.2 min	0.7 min/196.2 Mb
3'	1	3,6,18	30	0.1 min/123.5 Mb	4.6 min	0.4 min/98.3 Mb
4	1	3,6,18	30	0.1 min/122.4 Mb	4.5 min	0.3 min/97.8 Mb
5	1	3,6,18	30	0.1 min/123.7 Mb	4.6 min	0.4 min/98.2 Mb
1	1	3,6,24	36	0.1 min/148.1 Mb	6.5 min	0.5 min/118.3 Mb
2	1	3,6,24	36	0.2 min/295.5 Mb	13.0 min	0.9 min/236.2 Mb
3	1	3,6,24	36	0.2 min/321.6 Mb	15.6 min	1.3 min/259.7 Mb
3'	1	3,6,24	36	0.1 min/161.2 Mb	8.1 min	0.7 min/130.1 Mb
4	1	3,6,24	36	0.1 min/156.3 Mb	7.3 min	0.5 min/127.2 Mb
5	1	3,6,24	36	0.1 min/161.6 Mb	7.6 min	0.7 min/130.1 Mb
1	1	4,8,24	40	0.1 min/173.3 Mb	9.1 min	0.7 min/136.6 Mb
2	1	4,8,24	40	0.2 min/346.0 Mb	17.7 min	1.3 min/272.9 Mb
3	1	4,8,24	40	0.3 min/375.0 Mb	21.6 min	1.7 min/298.4 Mb
3'	1	4,8,24	40	0.1 min/187.9 Mb	10.8 min	0.9 min/149.8 Mb
4	1	4,8,24	40	0.1 min/186.1 Mb	10.1 min	0.8 min/149.0 Mb
5	1	4,8,24	40	0.1 min/188.4 Mb	10.6 min	1.3 min/149.8 Mb
1	1	3,15,90	105	2.1 min/823.3 Mb	190.3 min	17.9 min/678.3 Mb
2	1	3,15,90	105	4.2 min/1644.8 Mb	381.2 min	31.9 min/1361.4 Mb
3	1	3,15,90	105	5.8 min/1953.2 Mb	559.6 min	96.2 min/876.5 Mb
3'	1	3,15,90	105	2.9 min/977.7 Mb	281.8 min	48.6 min/437.1 Mb
4	1	3,15,90	105	2.1 min/868.3 Mb	213.3 min	12.5 min/381.9 Mb
5	1	3,15,90	105	3.2 min/984.9 Mb	279.9 min	155.2 min/467.7 Mb
6	2	3,6,18	30	0.2 min/273.1 Mb	21.3 min	1.8 min/216.2 Mb
7	2	3,6,18	30	0.2 min/276.4 Mb	21.5 min	1.7 min/218.2 Mb
6	2	3,6,24	36	0.2 min/351.5 Mb	35.9 min	3.7 min/283.4 Mb
7	2	3,6,24	36	0.3 min/355.3 Mb	36.4 min	4.1 min/285.7 Mb
6	2	4,8,24	40	0.3 min/408.1 Mb	45.6 min	5.1 min/325.5 Mb
7	2	4,8,24	40	0.3 min/413.3 Mb	47.1 min	5.1 min/329.6 Mb
6	4	3,6,18	30	0.4 min/535.3 Mb	80.0 min	6.9 min/425.9 Mb
7	4	3,6,18	30	0.4 min/538.6 Mb	83.8 min	6.9 min/428.1 Mb

Table 2. Checking properties of the communication protocol.

6 Conclusion

In this paper, we introduced a technique to perform a partial verification of real-time systems modeled with a dense time models and using mixed operational and descriptive components. The proposed approach is fully automated and implemented on top of a discrete-time bounded satisfiability checker. We experimented with a non-trivial example of a communication protocol, where concurrent runs of the protocol are synchronized through additional MTL formulas, hence building a mixed model. Verification tests showed consistent results and reasonable performances. As future work, we intend to improve the efficiency of the technique by using a pure CNF encoding. Another approach we wish to investigate is the use of other operational formalisms, such as timed Petri nets.

References

1. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
2. R. Alur, T. Feder, and T. A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, 1996.
3. R. Alur and T. A. Henzinger. Logics and models of real time: A survey. In *Proc. of Real-Time: Theory in Practice*, volume 600 of *LNCS*, pages 74–106, 1992.
4. R. Alur and T. A. Henzinger. Real-time logics: Complexity and expressiveness. *Information and Computation*, 104(1):35–77, 1993.
5. G. E. Fainekos and G. J. Pappas. Robust sampling for MITL specifications. In *Proc. of FORMATS'07*, volume 4763 of *LNCS*, 2007.
6. C. A. Furia. *Scaling up the formal analysis of real-time systems*. PhD thesis, DEI, Politecnico di Milano, May 2007.
7. C. A. Furia, D. Mandrioli, A. Morzenti, and M. Rossi. Modeling time in computing. Technical Report 2007.22, DEI, Politecnico di Milano, January 2007.
8. C. A. Furia, M. Pradella, and M. Rossi. Automated verification of dense-time MTL specifications via discrete-time approximation. In *Proc. of FM'08*, volume 5014 of *LNCS*, pages 132–147, 2008.
9. C. A. Furia, M. Pradella, and M. Rossi. Practical automated partial verification of multi-paradigm real-time models. Technical report, DEI, Politecnico di Milano, April 2008.
10. C. A. Furia and M. Rossi. Integrating discrete- and continuous-time metric temporal logics through sampling. In *Proc. of FORMATS'06*, volume 4202 of *LNCS*, pages 215–229, 2006.
11. T. A. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In *Proc. of ICALP'92*, volume 623 of *LNCS*, pages 545–558, 1992.
12. T. A. Henzinger, J.-F. Raskin, and P.-Y. Schobbens. The regular real-time languages. In *Proc. of ICALP'98*, volume 1443 of *LNCS*, pages 580–591, 1998.
13. R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.
14. K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1–2), 1997.
15. O. Maler, D. Nickovic, and A. Pnueli. From MITL to timed automata. In *Proc. of FORMATS'06*, volume 4202 of *LNCS*, pages 274–289, 2006.
16. M. Pradella. ℤot. `http://home.dei.polimi.it/pradella`, March 2007.
17. M. Pradella, A. Morzenti, and P. San Pietro. The symmetry of the past and of the future: bi-infinite time in the verification of temporal properties. In *Proc. of ESEC/FSE 2007*, 2007.