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

Carlo A. Furia, Matteo Pradella, and Matteo Rossi

April 2008

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. More precisely, 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 validity checker implementing the technique show consistent results and interesting performances.

Contents

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 seamlessly. 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. Hence, 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. More precisely: (1) it permits to analyze continuous-time models using fully automated, discrete-time verification techniques; and (2) it allows users to mix operational (TA) and descriptive (metric temporal logic, MTL) components in the system specification. The technique is partial in two respects: it can fail to provide conclusive answers, and only dense-time behaviors with bounded variability are verified. It involves an automated translation of the operational part into temporal logic notation, based on an MTL axiomatization discussed in this paper. The resulting MTL model, describing both the system and the properties to be verified, is then discretized according to the techniques introduced in [16]. The discrete-time approximation can be analyzed through conventional tools; we provide an implementation based on the ℤot bounded satisfiability checker [32].

We experimented with a significant example based on the description of a communication protocol by means of a timed automaton. Concurrent runs of the protocol are formalized by parallel instances of the same automaton; additionally, the simple synchronization rules between these instances is naturally formalized by means of additional MTL formulas, hence building a mixed model. Verification tests run on these models showed consistent results, and acceptable performances.

An interesting auxiliary contribution of the discretizable axiomatization of TA in MTL is a set of "rules of thumb" about how to describe systems based on the notion of state and transition with a logic formalism, in a way which is also amenable to discretization (according to the notion of [16]). Section 4 discusses this issue with great detail.

Finally, let us stress that our approach aims at providing a *practical* approach to the verification of operational (and mixed) 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.

Structure of the paper. The paper is organized as follows. Section 1.1 provides a sketch of the whole technique with as little technical details as possible. Section 1.2 briefly summarizes some research related to the content of this paper. Section 2 introduces the technical definitions that are needed in the remainder, namely the syntax and semantics of MTL and TA, and the discretization techniques from [17, 16] 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; the overall result is a set of discrete-time MTL formulas whose satisfiability is linked to the satisfiability of the original dense-time formulas according to the rules 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 Overview

The goal of our technique is to provide a means to carry out practical verification technique of real-time systems described using a dense notion of time and a mixture of operational and descriptive notations. In particular, we assume a model of real time based on the notion of *behavior*, which is basically a continuous-time signal, and we consider a variant of TA as operational formalism and MTL as descriptive formalism.

The most common approaches to similar verification problems involve translating the logic into automata [2]. In this paper we take the mirror approach of describing TA through MTL formulas. This choice is mainly justified by the fact that logic formulas are naturally compositional, hence our ultimate goal of formally combining mixed models is facilitated by this choice. It is well-known that MTL is undecidable over dense time [4]; this hurdle is however practically mitigated by employing the *discretization* technique for MTL introduced — and demonstrated to be practically appealing — in [16]. Note that 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 have a conclusive outcome to the verification problem. However, as demonstrated in [16], and further shown here, the impact of this shortcoming can be rendered small in many practical cases.

We start by providing a dense-time MTL axiomatization of TA. Notice that, due to a well-known expressiveness gap between temporal logics and automata [23] it is impossible to describe the language accepted by a generic TA as an MTL formula. What we provide is instead a formal description of *accepting runs* of a TA as an MTL formula; in other words, we model the overall behavior of TA with a set of MTL axioms. The resulting MTL axioms are discretized according to the rules provided in [16]. We show that this yields poor results

if done naïvely; hence, we carefully revise the axiomatization and put it in a way which is much more amenable to discretization. The result is a set of discretized MTL axioms describing TA runs. These axioms can be combined with additional pieces of specification, written in MTL, and with the properties to be verified. The resulting complete model can then be analyzed by means of automated discrete-time tools; the results of the discrete-time analysis are then used, as defined in [16], to finally infer results about the verification of the original dense-time model. The experimental results are encouraging, both in terms of performances and in terms of "completeness coverage" of the method.

In this paper we justify the soundness of the technique, which requires several analyses of the axiomatization and of the discretizations that are produced. It is important to understand, however, that the resulting technique (and tool) is completely automated, and the user has just to provide the dense-time model of the system (i.e., TA and MTL formulas) and the putative properties to be verified.

1.2 Related Work

To the best of our knowledge, our approach is rather unique in trying to combine operational and descriptive formalisms over dense time, then trading-off verification completeness against better performance 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 a very active field, and it has produced a few high-performance tools and methods. Let us mention, for instance, Uppaal [27], Kronos [35], HyTech [21], and PHAVer [14] for the verification of timed (and hybrid) automata. Notice that, although tools such as Uppaal allow the usage of a descriptive notation to express the properties to be verified, the temporal logic subset is very simple and of very 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.

Metric temporal logic (MTL) verification is also a well-understood research topic. MTL is however 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 recently significant progress has been made in simplifying them [28], a serviceable implementation is still lacking.

Another stance at working around the undecidability of dense-time MTL builds upon the fact that the same logic is decidable over discrete time. Hence, 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 [16] to the case of TA. A different discretization technique, based on the notion of robust satisfiability of MTL specifications, has been introduced in [13]. Other work also deals with notions of robustness in order to guarantee that dense-time TA are implementable with non-ideal architectures [11]. Another well-known notion of

discretization is the one based on the concept of *digitization* [22]; several authors have applied this quite general notion to the practical verification of descriptive [30, 24, 9, 34] or operational [20, 26, 7, 6, 29, 8, 5, 31, 10] formalisms. See also the related work section of [16] for more references about 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.

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, a (real-time) system model is simply a set of behaviors [3, 15] over some time domain $\mathbb{T}$ and sets of items and propositions. In practice, however, the modeler specifies a

system through some suitable notation. In this paper we consider Metric Temporal Logic (MTL) [25, 4] as descriptive notation, and TA [1, 2] as operational notation. Their syntax and semantics 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 describes 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.[1] Usually, one considers intervals with non-negative endpoints, but we permit negative endpoints to render the presentation more uniform and straightforward. Also, 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, i.e., $\beta ::= \mathsf{p} \mid \mathsf{it} = v \mid \neg\beta \mid \beta_1 \wedge \beta_2$ for $\mathsf{p} \in \mathcal{P}$, $\mathsf{it} \in \mathcal{I}$, $v \in \mathcal{D}^{\mathsf{it}}$.[2]

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

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 [12, 19]. 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. 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. The precedence order of logic connectives is, from the one of highest binding power: $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$.

MTL semantics. MTL *semantics* is defined over behaviors, parametrically with respect to the choice of the time domain $\mathbb{T}$.

[1] That is any $\mathcal{I} \ni I = \langle l, u \rangle$ for some $l \leq u$ where $l \in \mathbb{T} \cap \mathbb{Q}$ and $u \in (\mathbb{T} \cap \mathbb{Q}) \cup \{\pm\infty\}$, $\langle$ is one of (and [, and similarly for $\rangle$.

[2] Note that $\neg(\mathsf{it} = v)$ can be abbreviated as $\mathsf{it} \neq v$.

$b(t) \models_{\mathbb{T}} \mathsf{p}$	iff	$\mathsf{p} \in b(t)\|_{\mathcal{P}}$
$b(t) \models_{\mathbb{T}} \neg\mathsf{p}$	iff	$\mathsf{p} \notin b(t)\|_{\mathcal{P}}$
$b(t) \models_{\mathbb{T}} \mathsf{it} = v$	iff	$v = b(t)\|_{\mathsf{it}}$
$b(t) \models_{\mathbb{T}} \mathsf{it} \neq v$	iff	$v \neq b(t)\|_{\mathsf{it}}$
$b(t) \models_{\mathbb{T}} \mathsf{U}_I(\beta_1, \beta_2)$	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 $b(t+u) \models_{\mathbb{T}} \beta_1$
$b(t) \models_{\mathbb{T}} \mathsf{S}_I(\beta_1, \beta_2)$	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 $b(t+u) \models_{\mathbb{T}} \beta_1$
$b(t) \models_{\mathbb{T}} \mathsf{T}_I(\beta_1, \beta_2)$	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 $b(t-u) \models_{\mathbb{T}} \beta_1$
$b(t) \models_{\mathbb{T}} \phi_1 \wedge \phi_2$	iff	$b(t) \models_{\mathbb{T}} \phi_1$ and $b(t) \models_{\mathbb{T}} \phi_2$
$b(t) \models_{\mathbb{T}} \phi_1 \vee \phi_2$	iff	$b(t) \models_{\mathbb{T}} \phi_1$ or $b(t) \models_{\mathbb{T}} \phi_2$
$b \models_{\mathbb{T}} \phi$	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. Also notice that our MTL variant uses operators that are *non-strict* in their first argument, i.e., the future and past include the present instant, and the *until* and *since* operators are *matching*, i.e., they require their two arguments to hold together at some instant in I. Other work [18] analyzes the impact of these variants on expressiveness.

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.

2.2.1 MTL$^+$/MTL* syntax and semantics.

In order to express the discretization relations in Section 2.4, it is necessary to introduce some variations of the four basic temporal operators *until*, *since*, *release*, and *trigger*, denoted as $\mathsf{U}_I^{\uparrow}$, $\mathsf{S}_I^{\uparrow}$, $\mathsf{R}_I^{\downarrow}$, and $\mathsf{T}_I^{\downarrow}$, respectively. Notice that they are not part of the language in which dense-time specifications and properties are to be expressed, and they are needed only to illustrate the discretization techniques. We call "MTL$^+$" the *extension* of MTL with these operators, and "MTL*" the variant where we *replace* the operators U_I, S_I, R_I, T_I with $\mathsf{U}_I^{\uparrow}$, $\mathsf{S}_I^{\uparrow}$, $\mathsf{R}_I^{\downarrow}$, and $\mathsf{T}_I^{\downarrow}$, respectively.

Let us define the semantics of the new variants of *until* and *release*.

$b(t) \models_{\mathbb{T}} \mathsf{U}_I^{\uparrow}(\beta_1, \beta_2)$	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 $b(t+u) \models_{\mathbb{T}} \beta_1$
$b(t) \models_{\mathbb{T}} \mathsf{S}_I^{\uparrow}(\phi_1, \phi_2)$	iff	there exists $d \in I$ such that: $b(t-d) \models_{\mathbb{T}} \phi_2$ and, for all $u \in [0, d)$ it is $b(t-u) \models_{\mathbb{T}} \phi_1$
$b(t) \models_{\mathbb{T}} \mathsf{R}_I^{\downarrow}(\phi_1, \phi_2)$	iff	for all $d \in I$ it is: $b(t+d) \models_{\mathbb{T}} \phi_2$ or there exists a $u \in [0, d]$ such that $b(t+u) \models_{\mathbb{T}} \phi_1$
$b(t) \models_{\mathbb{T}} \mathsf{T}_I^{\downarrow}(\phi_1, \phi_2)$	iff	for all $d \in I$ it is: $b(t-d) \models_{\mathbb{T}} \phi_2$ or there exists a $u \in [0, d]$ such that $b(t-u) \models_{\mathbb{T}} \phi_1$

2.2.2 Derived Temporal Operators

It is useful to introduce a number of derived temporal operators, to be used as shorthands in writing specification formulas. We consider those listed in Table 1 ($\delta \in \mathbb{R}_{>0}$ is a parameter that will be used in the discretization technique described shortly).

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

Let us 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.

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, let us abbreviate by $\mathrm{Alw}(\phi)$ the nesting MTL formula $\phi \wedge \Box_{(0,+\infty)}(\phi) \wedge \overleftarrow{\Box}_{(0,+\infty)}(\phi)$; $b \models_{\mathbb{T}} \mathrm{Alw}(\phi)$ iff $b \models_{\mathbb{T}} \phi$, for any behavior b, so $\mathrm{Alw}(\phi)$ 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, 28]. 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.

On the other hand, 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, let us not consider acceptance conditions, that is let us assume that all states are accepting. Note, however, that introducing acceptance conditions (e.g., Büchi, Muller, etc.) in the formalization would be routine.

Timed automata 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.

Timed automata semantics. In defining the semantics of TA over behaviors we deviate from the standard presentation (e.g., [2, 28]) 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 $\mathsf{rs}_c|_{c \in C}$ 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 [23] — hence we define the semantics of automata over this "extended" state from the beginning.

Let us first define the semantics only informally. Initially, all clocks are reset and the automaton sits in some 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 to

some other state s' such that $\langle s, s', \Lambda, \xi \rangle$ is a valid transition, provided the last time (before t) each clock has been reset is compatible with the constraint ξ. If the transition is taken, all clocks in Λ are reset, whereas all the other clocks keep on running unchanged. Finally, as long as the automaton sits 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 $\alpha(s)$.

Formally, 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 $R = \{\mathsf{rs}_c\}_{c \in C}$. Intuitively, at any instant of time t, $\mathsf{st} = s$ means that the automaton is in state s, $\mathsf{in} = \sigma$ means that the automaton is reading symbol σ, and rs_c keeps track of resets of clock c (more precisely, 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 $t - k < t' < t$ such that $b(t') \not\models \mathsf{rs}_c$; or $b^-(t) \not\models \mathsf{rs}_c$ and there exists a $t - k < t' < t$ such that $b(t') \models \mathsf{rs}_c$
$b(t) \models c \geq k$	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.

Then, 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 $t_{\text{start}} > 0$ such that: $b(t)|_{\mathsf{st}} = s_0$ and $b(t)|_R = R$ for all $0 \leq t \leq t_{\text{start}}$, $b^-(t_{\text{start}}) = \langle s_0, \sigma^-, \rho^- \rangle$ and $b^+(t_{\text{start}}) = \langle s^+, \sigma^+, \rho^+ \rangle$ 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 $t > t_{\text{start}}$ 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

In [16] we presented a technique to reduce the validity problem for MTL specifications over dense time to the same problem over discrete time. In this section we concisely summarize the fundamental results from [16] that are needed in the remainder of the paper, and we provide some intuition about how they can be applied to our discretization problem.

2.4.1 Under- and Over-approximations of Formulas

We introduce two approximations of MTL formulas, called under- and over-approximation.

Under-approximation. The approximation function $\Omega_\delta(\cdot)$ maps dense-time MTL formulas to discrete-time MTL* formulas such that the non-validity of the latter implies the non-validity of the former, over behaviors in $\mathcal{B}^\delta_\chi$. More precisely, for MTL formulas such that the chosen sampling period δ is in $\mathcal{D}_\phi$, $\Omega_\delta(\cdot)$ is defined as follows.

$$
\begin{array}{lcl}
\Omega_\delta(\beta) & \equiv & \beta \\
\Omega_\delta(\phi_1 \wedge \phi_2) & \equiv & \Omega_\delta(\phi_1) \wedge \Omega_\delta(\phi_2) \\
\Omega_\delta(\phi_1 \vee \phi_2) & \equiv & \Omega_\delta(\phi_1) \vee \Omega_\delta(\phi_2) \\
\Omega_\delta\left(\mathsf{U}_{\langle l,u\rangle}(\phi_1,\phi_2)\right) & \equiv & \mathsf{U}^{\uparrow}_{[l/\delta,u/\delta]}(\Omega_\delta(\phi_1),\Omega_\delta(\phi_2)) \\
\Omega_\delta\left(\mathsf{S}_{\langle l,u\rangle}(\phi_1,\phi_2)\right) & \equiv & \mathsf{S}^{\uparrow}_{[l/\delta,u/\delta]}(\Omega_\delta(\phi_1),\Omega_\delta(\phi_2)) \\
\Omega_\delta\left(\mathsf{R}_{\langle l,u\rangle}(\phi_1,\phi_2)\right) & \equiv & \mathsf{R}^{\downarrow}_{\langle l/\delta,u/\delta\rangle}(\Omega_\delta(\phi_1),\Omega_\delta(\phi_2)) \\
\Omega_\delta\left(\mathsf{T}_{\langle l,u\rangle}(\phi_1,\phi_2)\right) & \equiv & \mathsf{T}^{\downarrow}_{\langle l/\delta,u/\delta\rangle}(\Omega_\delta(\phi_1),\Omega_\delta(\phi_2))
\end{array}
$$

Over-approximation. The approximation function $\mathrm{O}_\delta(\cdot)$ maps dense-time MTL formulas to discrete-time MTL formulas such that the validity of the latter implies the validity of the former, over behaviors in $\mathcal{B}^\delta_\chi$. More precisely, for MTL formulas such that the chosen sampling period δ is in $\mathcal{D}_\phi$, $\mathrm{O}_\delta(\cdot)$ is defined as follows.

$$
\begin{array}{lcl}
\mathrm{O}_\delta(\beta) & \equiv & \beta \\
\mathrm{O}_\delta(\phi_1 \vee \phi_2) & \equiv & \mathrm{O}_\delta(\phi_1) \vee \mathrm{O}_\delta(\phi_2) \\
\mathrm{O}_\delta(\phi_1 \wedge \phi_2) & \equiv & \mathrm{O}_\delta(\phi_1) \wedge \mathrm{O}_\delta(\phi_2) \\
\mathrm{O}_\delta\left(\mathsf{U}_{\langle l,u\rangle}(\phi_1,\phi_2)\right) & \equiv & \mathsf{U}_{[l/\delta+1,u/\delta-1]}(\mathrm{O}_\delta(\phi_1),\mathrm{O}_\delta(\phi_2)) \\
\mathrm{O}_\delta\left(\mathsf{S}_{\langle l,u\rangle}(\phi_1,\phi_2)\right) & \equiv & \mathsf{S}_{[l/\delta+1,u/\delta-1]}(\mathrm{O}_\delta(\phi_1),\mathrm{O}_\delta(\phi_2)) \\
\mathrm{O}_\delta\left(\mathsf{R}_{\langle l,u\rangle}(\phi_1,\phi_2)\right) & \equiv & \mathsf{R}_{[l/\delta-1,u/\delta+1]}(\mathrm{O}_\delta(\phi_1),\mathrm{O}_\delta(\phi_2)) \\
\mathrm{O}_\delta\left(\mathsf{T}_{\langle l,u\rangle}(\phi_1,\phi_2)\right) & \equiv & \mathsf{T}_{[l/\delta-1,u/\delta+1]}(\mathrm{O}_\delta(\phi_1),\mathrm{O}_\delta(\phi_2))
\end{array}
$$

2.4.2 System Verification through Approximation

We have the following fundamental verification result from [16], which provides a justification for the TA verification technique discussed in this paper.

Proposition 1 (Approximations [16])**.** *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.*

2.4.3 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 we cannot infer anything about the validity of the property in the original system over dense time.

Let us now provide some evidence about why different, but equivalent, dense-time formulas can yield dramatically different — in terms of usefulness — approximated discrete-time formulas. We provide one in-the-small example for over-approximations and one for under-approximations. More concrete examples will appear in Section 4 when building approximations of TA's axiomatic description.

Let us consider dense-time MTL formula $\theta_1 = \Box_{(0,\delta)}(\mathsf{p})$ 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 most 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 as simply $\Omega_\delta(\theta_1') = \mathsf{p}$, 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.

Let us now consider dense-time MTL formula $\theta_2 = \Diamond_{[0,2\delta]}(\mathsf{p})$, which describes a proposition p which is false for no longer than 2δ time units. If we compute its over-approximation, we get $\mathrm{O}_\delta(\theta_2) = \Diamond_{=1}(\mathsf{p})$ which, under the global satisfiability semantics, entails that p is always true. Although the actual assessment depends on the role θ plays in the overall specification, it is likely that this over-approximation is too coarse, as it basically adds "too strong" information to the discrete-time representation. So, if we build formula (2) from Proposition 1 with $\mathrm{O}_\delta(\theta_2)$ in it, it is very likely that the antecedent will be unsatisfiable (because $\mathrm{O}_\delta(\theta_2)$ introduces a very strong constraint) and hence formula (2) will be valid, yielding no information to the verification process. On the contrary, if we simply modify θ_2 into the *equivalent* $\theta_2' = \mathsf{p} \vee \theta_2$ we get an over-approximation which can be written as $\mathrm{O}_\delta(\theta_2') = \Diamond_{[0,1]}(\mathsf{p})$, i.e., p is false no more than every two time steps. This looks like a much better approximation, one which better preserves the original "meaning" of θ_2.

3 Formalizing Timed Automata in MTL

Let us consider a timed automaton $A = \langle \Sigma, S, S_0, \alpha, C, E \rangle$ and let us formalize its runs over non-Berkeley behaviors for some $\delta > 0$. In other words, we are going to provide 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 $j = 1, \ldots, 6$.

Translating clock constraints. We associate an MTL formula $\Xi(\xi)$ to every clock constraint ξ such that $b(t) \models \xi$ iff $b(t) \models \Xi(\xi)$ at all transition points t.

$\Xi(\xi)$ can be defined inductively as:

$$\begin{array}{lcl}
\Xi\,(c < k) & \equiv & \overleftarrow{\bigcirc}(\mathsf{rs}_c) \wedge \overleftarrow{\Diamond}_{(0,k)}(\neg\mathsf{rs}_c) \quad \vee \quad \overleftarrow{\bigcirc}(\neg\mathsf{rs}_c) \wedge \overleftarrow{\Diamond}_{(0,k)}(\mathsf{rs}_c) \\
\Xi\,(c \geq k) & \equiv & \overleftarrow{\bigcirc}(\mathsf{rs}_c) \wedge \overleftarrow{\Box}_{(0,k)}(\mathsf{rs}_c) \quad \vee \quad \overleftarrow{\bigcirc}(\neg\mathsf{rs}_c) \wedge \overleftarrow{\Box}_{(0,k)}(\neg\mathsf{rs}_c) \\
\xi_1 \wedge \xi_2 & \equiv & \Xi_1 \wedge \Xi_2 \\
\xi_1 \vee \xi_2 & \equiv & \Xi_1 \vee \Xi_2
\end{array}$$

Basically, Ξ translates the guard ξ by comparing the current time to the last time a reset for the clock c happened, where a reset is signaled by a switching of item rs_c. Notice that this assumes the existence of a "first reset" of all clocks, as specified in the formal semantics of TA, and as will be postulated in Formula (5) below. Also notice that, when computing the approximations of the clock-constraint formulas, we will have to require that every constant k used in the definition of the TA is 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 axiom:

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

Complementarily, we introduce an axiom to assert 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 σ, Λ, ξ, i.e., for any pair of states that are not connected by any edge:

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

Sufficient conditions for state change. We have 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 $s \in S$ we add the axiom:

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

Then, for each reset of a clock $c \in C$, let us consider all edges of the form $\langle s_i^k, s_j^k, \Lambda^k, \xi^k\rangle \in E$, such that $c \in \Lambda^k$. Hence, we introduce the pair of axioms:

$$\begin{array}{rcl}
\triangle(\neg\mathsf{rs}_c, \mathsf{rs}_c) & \Rightarrow & \bigvee_k \triangle\big(\mathsf{st}, s_i^k, s_j^k\big) \\
\triangle(\mathsf{rs}_c, \neg\mathsf{rs}_c) & \Rightarrow & \bigvee_k \triangle\big(\mathsf{st}, s_i^k, s_j^k\big) \vee \bigvee_{s_0\in S_0} \overleftarrow{\Box}_{(0,+\infty)} \left(\bigwedge_{c\in C} \mathsf{rs}_c \wedge \mathsf{st} = s_0\right)
\end{array} \tag{4}$$

Note that the second axiom has an additional part that takes into account the instants before the first reset (which must occur somewhere as shown in (5), and which corresponds to the instants before t_{start} in the formal semantics), whereas the first one is not applicable before such a first reset.

Initialization and liveness condition. We complete our axiomatization by first describing the system initialization.

We remark that the following axiom is only evaluated at 0. Notice that, under the global satisfiability semantics and with a mono-infinite time domain, a formula ϕ^0 that should be only evaluated at 0 can be expressed as $\overleftarrow{\Box}(\bot) \Rightarrow \phi^0$, as $\overleftarrow{\Box}(\bot)$ holds only where there is no past, i.e., at 0.

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

Notice that we make the axiomatization slightly more "deterministic" than the formal semantics, in that we require that t_{start}, when the first reset of the clocks occurs, is between 0 and 2δ; this, combined with the non-Berkeleyness requirement, says that it actually occurs between δ and 2δ. All in all, (5) pictures the following initialization:

- rs_c holds over $[0, \delta]$ for all $c \in C$;
- rs_c switches to false at some $t_{\text{start}} \in (\delta, 2\delta]$ for all $c \in C$ (clearly, this transition point is the same for all $c \in C$, still because of the non-Berkeleyness assumption);
- $\mathsf{st} = s_0$ holds for some $s_0 \in S_0$ over $[0, \delta]$;
- because of the non-Berkeleyness assumption, if st changes in $(\delta, 2\delta]$ it does so together with the resets at t_{start};
- $\triangle(\mathsf{rs}_c, \neg \mathsf{rs}_c)$ holds at t_{start} for all $c \in C$; the consequent of (4) is true because of the disjunct $\overleftarrow{\Box}_{(0,+\infty)} \big(\bigwedge_{c \in C} \mathsf{rs}_c \wedge \mathsf{st} = s_0 \big)$ which holds at t_{start}.

Finally, often we introduce a "liveness" condition which states that we eventually have to move out of every state, corresponding to the fact that all states are accepting *à la* Büchi. Thus, for every state $s \in S$, let $S'_s \subset S$ be the set of states that are directly reachable from s through a single transition; then we consider the axiom:

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

3.1 About the Correctness and Completeness of the Axiomatization

We omit a proof of the completeness and correctness of the axiomatization; we refer the reader to [19, App. D.6] where a proof for a similar axiomatization is sketched. Here, we just add a few remarks that can help justify the correctness and appropriateness of the present axiomatization.

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 $b \models A$ for some $t_{\text{start}} \in (\delta, 2\delta)$[3] if and only if $b \models \bigwedge_{1 \le j \le 6} \phi_j^A$.*

[3] This additional condition is introduced to take into account the particular form of the initialization axiom (5).

State changes can occur right- or left-continuously. It should be clear that the above axiomatization with the *becomes* operators does not force any item to transition either right- or left-continuously; in fact, the operator allows both possibilities. Over dense time, however, it would have been possible to force transitions to occur either always right- or always left-continuously. For instance, right-continuity can be achieved in one of the following ways:

- add formulas such as $\widetilde{\bigcirc}(\mathsf{st} = s_i) \Rightarrow s_i$;
- add formulas such as $\neg\left(\overleftarrow{\widetilde{\bigcirc}}(\mathsf{st} = s_i) \wedge \widetilde{\bigcirc}(\mathsf{st} = s_j)\right)$.

Correspondingly, the whole formalization could have been simplified a bit taking into account this new property.

Unfortunately, however, it is not difficult to see that all solutions would yield very poor discrete-time over-approximations, where by very poor we mean comprising only very trivial behaviors, and thus offering a very weak support to verification. For instance, the over- and under-approximations of $\bigcirc(\mathsf{st} = s_i) \Rightarrow s_i$ would require st to stay equal to s_i forever once it takes such value. Intuitively, this is due to the fact that a fine-grained information such as the edge of items at transition points is lost with a finite-precision sampling. There may be work-arounds for this, but it seems that they are overly complex. On the other hand, forgetting about characterizing transitions as right- or left-continuous allows us to get a much more straightforward axiomatization while still getting our approximations to work reasonably well.

4 Discrete-Time Approximations of Timed Automata

Let us show how to compute the under- and over-approximation of formulas (1–6) in a suitable way.

4.1 Under-approximation

The particular form of formulas (1–2),(4) is unsuitable to produce under-approximations that are strong enough to be useful.

Let us first of all notice that $\Omega_\delta\left(\widetilde{\bigcirc}(\beta)\right) = \Diamond_{[0,1]}(\beta)$ and $\Omega_\delta\left(\overleftarrow{\widetilde{\bigcirc}}(\beta)\right) = \overleftarrow{\Diamond}_{[0,1]}(\beta)$. In fact, over dense time, the definition of the *nowon* operator can be rewritten equivalently as: $\beta \wedge \mathsf{U}_{(0,+\infty)}(\beta, \top) \vee \neg\beta \wedge \mathsf{R}_{(0,+\infty)}(\beta, \bot)$, whose under-approximation is: $\beta \wedge \mathsf{U}^{\uparrow}(\beta, \top) \vee \neg\beta \wedge \mathsf{R}^{\downarrow}(\beta, \bot)$. Over discrete time, the latter is equivalent to $\beta \vee \neg\beta \wedge \Diamond_{=1}(\beta) = \Diamond_{[0,1]}(\beta)$. Correspondingly, $\Omega_\delta\left(\triangle(\beta_1, \beta_2)\right) = \Diamond_{[0,1]}(\beta_1) \wedge \Diamond_{[0,1]}(\beta_2)$. Then, for β_1, β_2 that cannot hold at the same instant (i.e., $\neg(\beta_1 \wedge \beta_2)$), this approximation is a suitable discrete-time representation of a transition from β_1 to β_2. However, consider $\Omega_\delta\left(\neg\triangle(\beta_1, \beta_2)\right) = \Omega_\delta\left(\overleftarrow{\widetilde{\bigcirc}}(\neg\beta_1) \vee \neg\beta_2 \wedge \widetilde{\bigcirc}(\neg\beta_2)\right) = \overleftarrow{\Diamond}_{[0,1]}(\neg\beta_1) \vee \neg\beta_2 \wedge \Diamond_{[0,1]}(\neg\beta_2) = \overleftarrow{\Diamond}_{[0,1]}(\neg\beta_1) \vee \neg\beta_2 = \neg(\overleftarrow{\Box}_{[0,1]}(\beta_1) \wedge \beta_2)$. There are two problems with this result. First, $\Omega_\delta\left(\neg\triangle(\beta_1, \beta_2)\right) \neq \neg\Omega_\delta\left(\triangle(\beta_1, \beta_2)\right)$; since we use $\triangle(\beta_1, \beta_2)$ to describe transitions, there are discrete-time behaviors where such a transition both occurs and

does not occur, i.e., $\Omega_\delta\,(\triangle(\beta_1,\beta_2))$ and $\Omega_\delta\,(\neg\triangle(\beta_1,\beta_2))$ are both true. Second, $\Omega_\delta\,(\neg\triangle(\beta_1,\beta_2))$ is very weak, in that it is true, in particular, whenever β_1 or β_2 are false; since $\triangle(\beta_1,\beta_2)$ is often used as antecedent of implications in our axiomatization, such implications are trivially true because $\neg\beta_1 \vee \neg\beta_2$ is an identity when β_1,β_2 cannot hold at the same instant.

This demands a thorough revision of the axiomatization, in order to make it amenable to under-approximations.

4.1.1 A New Axiomatization

The new axiomatization basically replaces every occurrence of $\triangle(\beta_1,\beta_2)$ with $\blacktriangle(\beta_1,\beta_2)$. Hence, formulas (1–2),(4) are changed as follows (notice that also Ξ is changed into $\overrightarrow{\Xi}$, as we are explaining shortly).

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

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

$$\begin{aligned} \blacktriangle(\neg\mathsf{rs}_c,\mathsf{rs}_c) \quad &\Rightarrow \quad \bigvee_k \blacktriangle\big(\mathsf{st},s_i^k,s_j^k\big) \\ \blacktriangle(\mathsf{rs}_c,\neg\mathsf{rs}_c) \quad &\Rightarrow \quad \bigvee_k \blacktriangle\big(\mathsf{st},s_i^k,s_j^k\big) \vee \bigvee_{s_0\in S_0} \overleftarrow{\Box}_{[0,+\infty)}(\mathsf{rs}_c \wedge \mathsf{st}=s_0) \end{aligned} \tag{9}$$

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

Let us now show that the new axiomatization — where formulas (1–2),(4) are replaced by the new formulas (7–9) — is indeed equivalent to the old one.

Proof that (1) iff (7). 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. □

Proof that (2) iff (8). Let $\triangle(\mathsf{st}, s_i, s_j)$ holds at t; we prove that $\blacktriangle(\mathsf{st}, s_i, s_j)$ at some t'. If the transition of st at t is right-continuous let $t' = t + \delta$, else let $t' = t$. From the non-Berkeleyness assumption we have that $\mathsf{st} = s_j$ at $t+\delta$ and $\mathsf{st} = s_i$ at $t - \delta$. Correspondingly, $\blacktriangle(\mathsf{st}, s_i, s_j)$ holds at t' because $\mathsf{st} = s_i$ at t' and $\mathsf{st} = s_j$ at $t' + \delta$.

For the converse, let $\blacktriangle(\mathsf{st}, s_i, s_j)$ holds at t; we prove that $\triangle(\mathsf{st}, s_i, s_j)$ at some t'. This is immediate because $\mathsf{st} = s_i$ at t and $\mathsf{st} = s_j$ at $t + \delta$ entail that there exists a transition instant $t' \in [t, t+\delta]$ where $\triangle(\mathsf{st}, s_i, s_j)$ holds. □

Proof that (4) iff (9). The proof of this part is along the same lines as for the proof that (1) iff (7). □

In the following sub-sections we are going to compute under-approximations of these new equivalent axiomatization, thus showing that the results are indeed much more satisfactory than with the original axioms. In fact, we can already see that $\Omega_\delta\left(\blacktriangle(\beta_1,\beta_2)\right) = \beta_1 \wedge \Diamond_{=1}(\beta_2) = \neg(\neg\beta_1 \vee \Diamond_{=1}(\neg\beta_2)) = \neg\Omega_\delta\left(\neg\blacktriangle(\beta_1,\beta_2)\right)$, thus solving the fundamental problem with the previous axiomatization.

4.1.2 Clock Constraints

Let us consider the under-approximations of clock constraints; they are both straightforward.

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

4.1.3 Formulas (1–2)

From the preliminaries, it is straightforward to re-write (7) in normal form, compute the under approximation, and re-write the resulting discrete-time formula as:

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

The under-approximation of (8) is also straightforward:

$$\neg\blacktriangle(\mathsf{st}, s_i, s_j) \tag{11}$$

4.1.4 Formulas (3–4)

Formula (9) has a structure similar to formula (7); so we immediately compute its under-approximations as:

$$\begin{aligned}
\blacktriangle(\neg\mathsf{rs}_c, \mathsf{rs}_c) \quad &\Rightarrow \quad \bigvee_k \blacktriangle\big(\mathsf{st}, s_i^k, s_j^k\big) \\
\blacktriangle(\mathsf{rs}_c, \neg\mathsf{rs}_c) \quad &\Rightarrow \quad \bigvee_k \blacktriangle\big(\mathsf{st}, s_i^k, s_j^k\big) \vee \bigvee_{s_0\in S_0} \overleftarrow{\Box}_{[0,+\infty)}(\mathsf{rs}_c \wedge \mathsf{st} = s_0)
\end{aligned} \tag{12}$$

Also, simply $\Omega_\delta\left((3)\right) = (3)$.

4.1.5 Formulas (5–6)

Formula (6) is unchanged under under-approximation (after noticing that $\Diamond_{[0,\infty)}(\phi)$ is equivalent to $\Diamond(\phi)$ when the antecedent of (6) holds), so $\Omega_\delta\left((6)\right) = (6)$. Formulas (5–6) are straightforward to under-approximate, and they produce discrete-time formulas that are perfectly adequate.

Next, let us consider (5) instead. Since $\Omega_\delta\left(\neg\overleftarrow{\Box}(\bot)\right) = \top$, we first re-write it as:

$$\overleftarrow{\Box}_{[\delta,+\infty]}(\bot) \Rightarrow \bigwedge_{c\in C} \mathsf{rs}_c \wedge \Diamond_{[0,2\delta]}\left(\bigwedge_{c\in C} \neg\mathsf{rs}_c\right) \wedge \bigvee_{s_0\in S_0} \mathsf{st} = s_0 \tag{13}$$

Let us discuss why (13) and (5) are equivalent, when considered together with the other axioms. $\overleftarrow{\Box}_{[\delta,+\infty]}(\bot)$ holds precisely over $[0,\delta)$, thus (13) asserts that:

- rs_c holds over $[0,\delta)$ for all $c \in C$;
- rs_c switch to false at some $t_{\text{start}} \in [\delta, 2\delta]$ for all $c \in C$;
- $\mathsf{st} = s_0$ holds for some $s_0 \in S_0$ over $[0,\delta)$; note that is must be the same s_0 throughout the interval, still because of the non-Berkeley assumption;
- because of the non-Berkeleyness assumption, if st changes in $(\delta, 2\delta]$ it does so together with the resets at t_{start};
- $\blacktriangle(\mathsf{rs}_c, \neg\mathsf{rs}_c)$ holds over $[t_{\text{start}} - \delta, t_{\text{start}})$ for all $c \in C$; the consequent of (9) is true because of the disjunct $\overleftarrow{\Box}_{[0,+\infty)}\left(\bigwedge_{c\in C} \mathsf{rs}_c \wedge \mathsf{st} = s_0\right)$ which holds throughout $[0, t_{\text{start}})$.

All in all the new initialization formula forces a behavior which is the same as in the original one. Then, given that $\Omega_\delta\left(\neg\overleftarrow{\Box}_{[\delta,+\infty)}(\bot)\right) = \Diamond(\top)$ which holds everywhere except at 0, we compute Ω_δ ((13)):

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

where $\Diamond_{[0,2]}\left(\bigwedge_{c\in C} \neg\mathsf{rs}_c\right)$ has been rewritten as $\Diamond_{[1,2]}\left(\bigwedge_{c\in C} \neg\mathsf{rs}_c\right)$ because $\bigwedge_{c\in C} \mathsf{rs}_c$ holds at 0.

In addition, we notice the following fact. Assume that $\bigwedge_{c\in C} \neg\mathsf{rs}_c$ holds at 1; then (4) can require a state transition only for instants ≥ 1. Otherwise, assume that $\bigwedge_{c\in C} \neg\mathsf{rs}_c$ holds at 2 at that *some* resets switch at 1, i.e., there exists a $D \subset C$ such that: (a) $\bigwedge_{c\in C} \mathsf{rs}_c$ at 0, (b) $\bigwedge_{c\in D} \mathsf{rs}_c$ at 1, and (c) $\bigwedge_{c\in C} \neg\mathsf{rs}_c$ at 2. Then, (4) requires a state transition at 1. All in all, (12) can be rewritten equivalently without the $\bigvee_{s_0\in S_0} \overleftarrow{\Box}_{[0,+\infty)}(\mathsf{rs}_c \wedge \mathsf{st} = s_0)$ part if it is evaluated only at instants ≥ 1.

4.2 Over-approximation

Formulas (1–6) are in a form which is unsuitable to compute useful over-approximation. Hence, we follow the same path as for the under-approximation: we introduce a different, albeit equivalent, continuous-time axiomatization, which is then amenable to over-approximation.

4.2.1 Preliminaries

Let us consider a generic Boolean combination β and let us compute the following over-approximations (clearly, the justifications for those with past operators are the same as for the future operators, so they are omitted for brevity):

- $\mathrm{O}_\delta\left(\widetilde{\bigcirc}(\beta)\right) = \Box_{[0,1]}(\beta)$.
  From the definition of the *nowon* operator, we have: $\mathsf{U}_{\geq 1}(\beta, \top) \vee (\neg\beta \wedge \mathsf{R}_{\geq -1}(\beta, \bot))$. Over discrete time, it is easy to check that $\mathsf{U}_{\geq 1}(\beta, \top)$ is equivalent to $\Box_{[0,1]}(\beta)$; on the other hand, the second disjunct $\neg\beta \wedge \mathsf{R}_{\geq -1}(\beta, \bot)$ is equivalent to $\bot$, as when $d \leq 0$ the interval $[0, d)$ is empty.

- $\mathrm{O}_\delta\left(\Diamond_{[0,2\delta]}(\beta)\right) = \Diamond_{=1}(\beta)$.
- $\mathrm{O}_\delta\left(\bigcirc(\beta)\right) = \Box_{[0,1]}(\beta)$.
- $\mathrm{O}_\delta\left(\widetilde{\overleftarrow{\bigcirc}}(\beta)\right) = \mathrm{O}_\delta\left(\overleftarrow{\bigcirc}(\beta)\right) = \overleftarrow{\Box}_{[0,1]}(\beta)$.
- $\mathrm{O}_\delta\left(\neg\triangle(\beta_1,\beta_2)\right) = \neg(\triangle(\beta_1,\beta_2) \vee \blacktriangle(\beta_1,\beta_2))$, assuming β_1, β_2 cannot hold at the same instant.
  Recall the definition of $\triangle(\beta_1,\beta_2)$, so $\neg\triangle(\beta_1,\beta_2) = \widetilde{\overleftarrow{\bigcirc}}(\neg\beta_1) \vee (\neg\beta_2 \wedge \widetilde{\bigcirc}(\neg\beta_2))$. Thus, $\mathrm{O}_\delta\left(\neg\triangle(\beta_1,\beta_2)\right) = \overleftarrow{\Box}_{[0,1]}(\neg\beta_1) \vee (\neg\beta_2 \wedge \Box_{[0,1]}(\neg\beta_2)) = \overleftarrow{\Box}_{[0,1]}(\neg\beta_1) \vee \Box_{[0,1]}(\neg\beta_2)$. By pushing negations outward in the latter, we get: $\neg(\overleftarrow{\Diamond}_{[0,1]}(\beta_1) \wedge \Diamond_{[0,1]}(\beta_2))$, which is equivalent to $\neg\left(\triangle(\beta_1,\beta_2) \vee \blacktriangle(\beta_1,\beta_2)\right)$ if β_1, β_2 cannot hold at the same instant.

4.2.2 Clock Constraints

It is not difficult to compute the over-approximations of the "existential" clock constraint. In fact, we have:

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

On the contrary, we have to "massage" the "universal" clock constraints into a more suitable form; otherwise, e.g., $\mathrm{O}_\delta\left(\overleftarrow{\Box}_{(0,k)}(\mathsf{rs}_c)\right) = \overleftarrow{\Box}_{[-1,k/\delta+1]}(\mathsf{rs}_c)$ but the latter is never satisfiable if c is both checked and reset when a transition is taken. We can, however, perform a transformation where $\Xi(c \geq k)$ becomes:

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

which is seen to be equivalent for non-Berkeley behaviors at transition points (when clock constraints are evaluated). Hence, we have:

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

4.2.3 Attempting Formula (1)

It is not difficult to see that formula (1) yields a very poor over-approximation. In particular, the portions in the consequent corresponding to the clock resets: $\triangle(\neg\mathsf{rs}_c, \mathsf{rs}_c) \vee \triangle(\mathsf{rs}_c, \neg\mathsf{rs}_c)$ become, when over-approximated:

$$\begin{aligned}\mathrm{O}_\delta\Big(&\widetilde{\overleftarrow{\bigcirc}}(\mathsf{rs}_c) \wedge (\neg\mathsf{rs}_c \vee \widetilde{\bigcirc}(\neg\mathsf{rs}_c)) \;\vee\; \widetilde{\overleftarrow{\bigcirc}}(\neg\mathsf{rs}_c) \wedge (\mathsf{rs}_c \vee \widetilde{\bigcirc}(\mathsf{rs}_c))\Big) \;=\\ &\overleftarrow{\Box}_{[0,1]}(\mathsf{rs}_c) \wedge (\neg\mathsf{rs}_c \vee \Box_{[0,1]}(\neg\mathsf{rs}_c)) \;\vee\; \overleftarrow{\Box}_{[0,1]}(\neg\mathsf{rs}_c) \wedge (\mathsf{rs}_c \vee \Box_{[0,1]}(\mathsf{rs}_c)) \quad (15)\end{aligned}$$

Clearly, the above discrete-time formula is unsatisfiable, as, for instance, $\overleftarrow{\Box}_{[0,1]}(\mathsf{rs}_c)$ is in contradiction with $\neg\mathsf{rs}_c \vee \Box_{[0,1]}(\neg\mathsf{rs}_c)$. Similar problems arise with the over-approximations of formula (4).

As a consequence, the over-approximation axioms would only be satisfiable with behaviors where the antecedents are identically false. It is not difficult to

realize that such behaviors would be the trivial ones, where no transition ever happens. This in turn would contradict (the over-approximation of) formula (6). So, overall, we end up with a set of over-approximated axioms which are unsatisfiable; clearly, this is of little interest for checking non-validity, as an unsatisfiable set of axioms entails any property.

4.2.4 A New Axiomatization

However, we can rewrite our axioms in a form which is equivalent but which yields much better discrete-time over-approximations.

Let us rewrite formulas (1),(4) as follows (formulas (2–3) are instead unchanged).

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

$$\begin{aligned} \triangle(\neg \mathsf{rs}_c, \mathsf{rs}_c) \quad &\Rightarrow \quad \bigvee_k \left(\widetilde{\overleftarrow{\bigcirc}}\big(\mathsf{st} = s_i^k\big) \wedge \Box_{=\delta}\big(\mathsf{rs}_c \Rightarrow \mathsf{st} = s_j^k\big) \right) \\ \triangle(\mathsf{rs}_c, \neg \mathsf{rs}_c) \quad &\Rightarrow \quad \bigvee_k \left(\widetilde{\overleftarrow{\bigcirc}}\big(\mathsf{st} = s_i^k\big) \wedge \Box_{=\delta}\big(\neg \mathsf{rs}_c \Rightarrow \mathsf{st} = s_j^k\big) \right) \\ &\qquad \vee \bigvee_{s_0 \in S_0} \overleftarrow{\Box}_{[\delta,+\infty)}(\mathsf{rs}_c \wedge \mathsf{st} = s_0) \end{aligned} \tag{17}$$

We claim that these new axioms describe the *same* behaviors as the original axioms (1–4).

Proof that (1) iff (16). Since the antecedents of (1) and (16) are the same, we just have to prove that the consequents are equivalent, assuming that the antecedents hold. So let $\triangle(\mathsf{st}, s_i, s_j)$ hold at the current instant t; this means that item st transitions from s_i to s_j. In particular, notice that the non-Berkeleyness requirement for δ entails that s_j holds at least over the interval $(t, t+\delta]$.

Now, let $d \in C$. Note that $\widetilde{\bigcirc}(\mathsf{rs}_d)$ at t iff $\mathsf{st} = s_j \Rightarrow \mathsf{rs}_d$ at $t+\delta$, because t is a transition point, so the non-Berkeleyness requirement entails that rs_d holds throughout $(t, t+\delta]$. Hence, $\triangle(\neg \mathsf{rs}_d, \mathsf{rs}_d)$ iff $\widetilde{\overleftarrow{\bigcirc}}(\neg \mathsf{rs}_d) \wedge \Box_{=\delta}(\mathsf{st} = s_j \Rightarrow \mathsf{rs}_d)$, at t. Since the reasoning holds for a generic clock, and also for the converse transition from rs_d to $\neg \mathsf{rs}_d$, and $\Xi(\xi^k)$ is the same in both (1) and (16) we have proved that (1) iff (16). □

Proof that (4) iff (17). Proofs along the very same lines can be provided for formulas formulas (4) and (17). We only notice that the term $\overleftarrow{\Box}_{(0,+\infty)}(\mathsf{rs}_c \wedge \mathsf{st} = s_0)$ has been equivalently changed to $\overleftarrow{\Box}_{[\delta,+\infty)}(\mathsf{rs}_c \wedge \mathsf{st} = s_0)$. In fact, (5) entails that $\mathsf{st} = s_0$ holds throughout $[0, \delta]$, hence $\triangle(\mathsf{rs}_c, \neg \mathsf{rs}_c)$ is false over $[0, \delta)$. We omit all other details for brevity. □

4.2.5 Formulas (1–2)

The newly built formula (16) is now amenable to over-approximation. In fact, we have the following discrete-time formula.

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

Notice instead that the over-approximation of (2) is simply:

$$\neg(\triangle(\mathsf{st}, s_i, s_j) \vee \blacktriangle(\mathsf{st}, s_i, s_j)) \tag{19}$$

4.2.6 Formula (4)

Formula (17) has a structure similar to formula (16); so we immediately compute its over-approximations as:

$$\begin{aligned}\triangle(\neg \mathsf{rs}_c, \mathsf{rs}_c) \vee \blacktriangle(\neg \mathsf{rs}_c, \mathsf{rs}_c) \quad &\Rightarrow \quad \bigvee_k \left(\overleftarrow{\Box}_{[0,1]}\left(\mathsf{st} = s_i^k\right) \wedge \Box_{[0,2]}\left(\mathsf{rs}_c \Rightarrow \mathsf{st} = s_j^k\right) \right) \\ \triangle(\mathsf{rs}_c, \neg \mathsf{rs}_c) \vee \blacktriangle(\mathsf{rs}_c, \neg \mathsf{rs}_c) \quad &\Rightarrow \quad \bigvee_k \left(\overleftarrow{\Box}_{[0,1]}\left(\mathsf{st} = s_i^k\right) \wedge \Box_{[0,2]}\left(\neg \mathsf{rs}_c \Rightarrow \mathsf{st} = s_j^k\right) \right) \\ &\qquad \vee \bigvee_{s_0 \in S_0} \overleftarrow{\Box}_{[0,+\infty)}(\mathsf{rs}_c \wedge \mathsf{st} = s_0) \end{aligned} \tag{20}$$

4.2.7 Some Simplifications

In this section we show how to re-write discrete-time formulas (18–20) above in a simpler but equivalent form.

Let us start by noting that the formulas have a similar structure, and in particular have antecedents that are structurally identical, the only difference being the items they predicate about. In fact, these antecedent describe a transition of an item from a value to another value; so (18) describes a transition of item st from s_i to s_j, (20) a transition of some rs_c, etc.

Let us consider a generic current instant h where the antecedent of (18) holds and let us spell out what form the transition of st can take. $\triangle(\mathsf{st}, s_i, s_j) \vee \blacktriangle(\mathsf{st}, s_i, s_j)$ holds precisely in the following three cases:

1. $\mathsf{st} = s_i$ holds at $h-1$ and $\mathsf{st} = s_j$ holds at h;
2. $\mathsf{st} = s_i$ holds at $h-1$ and $\mathsf{st} = s_j$ holds at $h+1$;
3. $\mathsf{st} = s_i$ holds at h and $\mathsf{st} = s_j$ holds at $h+1$.

We are going to show that case 1 is in contradiction with the other axioms, and therefore can be removed from the axiomatization.

So, assume that $\mathsf{st} = s_i$ holds at $h-1$ and $\mathsf{st} = s_j$ holds at h. The consequent of (18) is then contradictory: $\overleftarrow{\Box}_{[0,1]}(\neg \mathsf{rs}_c)$ implies that rs_c is false at h, but $\Box_{[0,2]}(\mathsf{st} = s_j \Rightarrow \mathsf{rs}_c)$ implies that rs_c is true at h because $\mathsf{st} = s_j$ is the case. All similarly if $\overleftarrow{\Box}_{[0,1]}(\mathsf{rs}_c)$ holds.

It is simple to see that similar contradictions arise if we consider a transition for rs_c from false to true (or true to false) for some $c \in C$. We conclude that we should never consider transitions as in case 1.

Now, notice that if case 1 never holds, case 2 reduces to case 3. In fact, it cannot be $\mathsf{st} = s_j$ at h or we would have case 1, so it must be $\mathsf{st} = s_i$ at h. All in all, every antecedent in formulas (18–20) can be simplified into just $\blacktriangle(\mathsf{st}, s_i, s_j) \equiv \mathsf{st} = s_i \wedge \Diamond_{=1}(\mathsf{st} = s_j)$ and similar ones.

Finally, notice that also formula (19) can be simplified into just $\neg\blacktriangle(\mathsf{st}, s_i, s_j)$. To see this, assume to the contrary that $\mathsf{st} = s_i$ holds at $h-1$ and $\mathsf{st} = s_j$ holds at h, for some pair of states s_i, s_j which do not belong to any transition. In this case, $\blacktriangle(\mathsf{st}, s_i, s_j)$ holds at $h-1$, thus the new formula is false, which shows that such a transition cannot occur even with the new, weaker formula.

All in all, we have formulas (18–20) simplified as follows.

$$\blacktriangle(\mathsf{st}, s_i, s_j) \quad \Rightarrow$$
$$\bigvee_k \left(\mathrm{O}_\delta\left(\Xi(\xi^k)\right) \wedge \bigwedge\nolimits_{c\in\Lambda^k} \left(\begin{array}{c} \overleftarrow{\Box}_{[0,1]}(\neg\mathsf{rs}_c) \wedge \Box_{[0,2]}(\mathsf{st} = s_j \Rightarrow \mathsf{rs}_c) \\ \vee \\ \overleftarrow{\Box}_{[0,1]}(\mathsf{rs}_c) \wedge \Box_{[0,2]}(\mathsf{st} = s_j \Rightarrow \neg\mathsf{rs}_c) \end{array} \right) \right) \tag{21}$$
$$\neg\blacktriangle(\mathsf{st}, s_i, s_j) \tag{22}$$

$$\begin{aligned} \blacktriangle(\neg\mathsf{rs}_c, \mathsf{rs}_c) \quad &\Rightarrow \quad \bigvee_k \left(\overleftarrow{\Box}_{[0,1]}\left(\mathsf{st} = s_i^k\right) \wedge \Box_{[0,2]}\left(\mathsf{rs}_c \Rightarrow \mathsf{st} = s_j^k\right) \right) \\ \blacktriangle(\mathsf{rs}_c, \neg\mathsf{rs}_c) \quad &\Rightarrow \quad \bigvee_k \left(\overleftarrow{\Box}_{[0,1]}\left(\mathsf{st} = s_i^k\right) \wedge \Box_{[0,2]}\left(\neg\mathsf{rs}_c \Rightarrow \mathsf{st} = s_j^k\right) \right) \\ &\qquad \vee \bigvee_{s_0\in S_0} \overleftarrow{\Box}_{[0,+\infty)}(\mathsf{rs}_c \wedge \mathsf{st} = s_0) \end{aligned} \tag{23}$$

4.2.8 Formulas (3),(5–6)

Notice that simply $\mathrm{O}_\delta((3)) = (3)$ and $\mathrm{O}_\delta((6)) = (6)$.

For (5) notice that $\mathrm{O}_\delta\left(\neg\overleftarrow{\Box}(\bot)\right) = \overleftarrow{\Diamond}_{[1,+\infty)}(\top)$ which holds everywhere except at 0. Thus, we can write $\mathrm{O}_\delta((5))$ as:

$$\text{at 0:} \quad \bigwedge_{c\in C} \mathsf{rs}_c \wedge \Diamond_{=1}\left(\bigwedge_{c\in C} \neg\mathsf{rs}_c\right) \wedge \bigvee_{s_0\in S_0} \Box_{[0,1]}(\mathsf{st} = s_0) \tag{24}$$

Notice that (24) entails that $\overleftarrow{\Box}_{[0,+\infty)}(\mathsf{rs}_c \wedge \mathsf{st} = s_0)$ holds for some $s_0 \in S_0$ at 0. Correspondingly, (20) can be rewritten equivalently without the $\bigvee_{s_0\in S_0} \overleftarrow{\Box}_{[0,+\infty)}(\mathsf{rs}_c \wedge \mathsf{st} = s_0)$ part if it is evaluated only at instants ≥ 1.

4.3 Summary

The following proposition summarizes the results of the discrete-time approximation formulas.

Proposition 3. *Let S be a real-time system described by timed automaton $A = \langle \Sigma, S, S_0, \alpha, C, E \rangle$ 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:*

$$\mathrm{Alw}\left(\phi^A_{(10)} \wedge \phi^A_{(11)} \wedge \phi^A_{(3)} \wedge \phi^A_{(12)} \wedge \phi^A_{(14)} \wedge \phi^A_{(6)} \wedge \bigwedge_j \Omega_\delta\left(\phi_j^{\mathsf{sys}}\right)\right) \Rightarrow \mathrm{Alw}(\mathrm{O}_\delta\left(\phi^{\mathsf{prop}}\right))$$

  is $\mathbb{N}$-valid, then ϕ^{prop} is satisfied by all non-Berkeley runs $b \in \mathcal{B}^\delta_\chi$ of the system (with $t_{\mathrm{start}} \in (\delta, 2\delta)$);

- *if:*

$$\mathrm{Alw}\left(\phi^A_{(21)} \wedge \phi^A_{(22)} \wedge \phi^A_{(3)}\phi^A_{(23)} \wedge \phi^A_{(24)} \wedge \phi^A_{(6)} \wedge \bigwedge_j \mathrm{O}_\delta\left(\phi_j^{\mathsf{sys}}\right)\right) \Rightarrow \mathrm{Alw}(\Omega_\delta\left(\phi^{\mathsf{prop}}\right))$$

  is not $\mathbb{N}$-valid, then ϕ^{prop} is not satisfied by all non-Berkeley runs $b \in \mathcal{B}^\delta_\chi$ of the system (with $t_{\mathrm{start}} \in (\delta, 2\delta)$).

5 Implementation and Example

This section describes briefly the implementation of the verification technique introduced in the previous section and it discusses an example of system verified with the resulting tool.

5.1 TAZot

We implemented the verification technique of this paper as a plugin to the Zot bounded satisfiability checker [32, 33] named TAZot. The plugin provides a set of primitives by which the user can provide the description of a timed automaton, 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 $\mathbb{N}$ bounded by some user-defined constant; the results of the validity check allows one to infer the validity of the original dense-time models, according to Proposition 3.

More precisely, 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) for the validity checking.

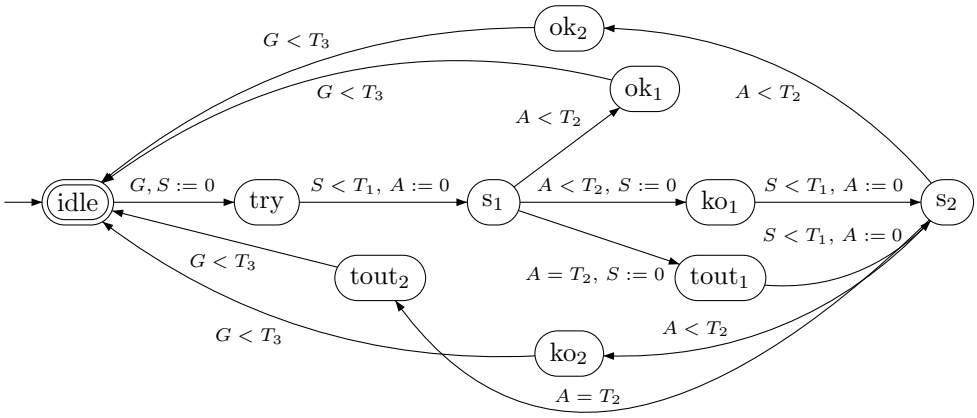

Figure 1: Timed automaton modeling the communication protocol.

5.2 A Communication Protocol Example

We demonstrate the practical feasibility of our verification techniques by means of an example, where we verify certain properties of a communication protocol, modeled through a timed automaton.

5.2.1 Description of the Protocol

Let us 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. Recall that the definition of clock constraints given in Section 2.3 forbids the introduction of exact constraints such as $A = T_2$. Hence, we mean clock constraints in the form $C = T$ as a shorthand for the valid clock constraint $T \leq C < T + \delta$, where δ is the chosen sampling period. In other words, we approximate exact clock constraints to within a tolerance which is given by the time granularity δ.

5.2.2 Properties of the System

Let us describe the properties we verified using our technique. We verified 5 properties of a single instance of the automaton, and 2 other properties of

a concurrent run of two (or more) instances of the automaton, synchronized according to additional MTL axioms described below. We included a *false* property among the former 5, in order to show how the verification technique works at disproving false properties.

Single instance properties.

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, as it is, falls in the incompleteness area of the method. In fact, whether a run is completed in T_3/δ time instants depends sensibly on how the sampling is chosen, so the method cannot conclude anything within its accuracy. 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."
$$\mathrm{ok}_1 \quad \Rightarrow \quad \mathsf{U}_{(0,T_3)}(\neg(\mathrm{ko}_1 \vee \mathrm{ko}_2), \mathrm{idle})$$

Concurrent run properties. Let us now assume that the server runs two concurrent instances of the same protocol. Since the two processes run on the same hardware, it is reasonable to assume that the outcomes of two parallel protocol runs will be correlated. More precisely, we assume that two parallel protocol runs that are initiated concurrently either both terminate successfully, or both terminate unsuccessfully. To formalize this assumption, we augment our operational model with the following MTL axiom, where corresponding states of the two automata instances are differentiated by a superscripted A or B:

$$\begin{aligned}
&\mathrm{try}^A \wedge \mathrm{try}^B \Rightarrow \\
&\mathsf{U}\Big(\neg(\mathrm{tout}_2{}^A \vee \mathrm{ko}_2{}^A), \mathrm{ok}_1{}^A \vee \mathrm{ok}_2{}^A\Big) \wedge \mathsf{U}\Big(\neg(\mathrm{tout}_2{}^B \vee \mathrm{ko}_2{}^B), \mathrm{ok}_1{}^B \vee \mathrm{ok}_2{}^B\Big) \\
&\qquad\qquad \vee \\
&\mathsf{U}\Big(\neg(\mathrm{ok}_1{}^A \vee \mathrm{ok}_2{}^A), \mathrm{tout}_2{}^A \vee \mathrm{ko}_2{}^A\Big) \wedge \mathsf{U}\Big(\neg(\mathrm{ok}_1{}^B \vee \mathrm{ok}_2{}^B), \mathrm{tout}_2{}^B \vee \mathrm{ko}_2{}^B\Big)
\end{aligned} \tag{25}$$

It is also simple to conceive a generalization of (25) to $N \geq 2$ concurrent runs, where we re-state the same property for every pair of instances, that is:

$$\begin{aligned}\forall 1 \leq i < j \leq N: \quad & \mathrm{try}^i \wedge \mathrm{try}^j \Rightarrow \\ & \mathsf{U}\big(\neg(\mathrm{tout_2}^i \vee \mathrm{ko_2}^i), \mathrm{ok_1}^i \vee \mathrm{ok_2}^i\big) \wedge \mathsf{U}\big(\neg(\mathrm{tout_2}^j \vee \mathrm{ko_2}^j), \mathrm{ok_1}^j \vee \mathrm{ok_2}^j\big) \\ & \qquad \vee \\ & \mathsf{U}\big(\neg(\mathrm{ok_1}^i \vee \mathrm{ok_2}^i), \mathrm{tout_2}^i \vee \mathrm{ko_2}^i\big) \wedge \mathsf{U}\big(\neg(\mathrm{ok_1}^j \vee \mathrm{ok_2}^j), \mathrm{tout_2}^j \vee \mathrm{ko_2}^j\big) \quad (26)\end{aligned}$$

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."

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

5.3 Experimental Evaluation

Tables 2 shows some results obtained in tests with TAZot 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 size k of the explored state space (as ZZot is a bounded satisfiability checker); 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); and the total size (in thousands of clauses) of the propositional formulas that have been checked.

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). TAZot used GNU CLisp v. 2.41 and MiniSat v. 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 claim safely 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)	# KCl.
1	1	3,6,18	30	0.1 min/114.6 Mb	3.9 min	0.3 min/90.2 Mb	520.2
2	1	3,6,18	30	0.1 min/228.6 Mb	7.8 min	0.5 min/180.1 Mb	1037.9
3	1	3,6,18	30	0.2 min/244.3 Mb	9.1 min	0.7 min/195.6 Mb	1112.4
3'	1	3,6,18	30	0.1 min/122.5 Mb	4.6 min	0.4 min/98.0 Mb	557.7
4	1	3,6,18	30	0.1 min/121.4 Mb	4.5 min	0.3 min/97.4 Mb	553.2
5	1	3,6,18	30	0.1 min/122.6 Mb	4.6 min	0.4 min/97.9 Mb	557.3
1	1	3,6,24	36	0.1 min/146.8 Mb	6.3 min	0.5 min/117.9 Mb	669.1
2	1	3,6,24	36	0.2 min/292.9 Mb	12.5 min	0.9 min/235.4 Mb	1335.2
3	1	3,6,24	36	0.2 min/319.0 Mb	15.4 min	1.2 min/258.6 Mb	1459.0
3'	1	3,6,24	36	0.1 min/159.9 Mb	7.6 min	0.7 min/129.3 Mb	731.3
4	1	3,6,24	36	0.1 min/155.0 Mb	7.2 min	0.5 min/126.4 Mb	708.5
5	1	3,6,24	36	0.1 min/160.3 Mb	7.8 min	0.9 min/129.8 Mb	731.3
1	1	4,8,24	40	0.1 min/171.9 Mb	8.5 min	0.7 min/136.2 Mb	785.5
2	1	4,8,24	40	0.2 min/343.1 Mb	17.2 min	1.2 min/271.9 Mb	1567.7
3	1	4,8,24	40	0.3 min/372.1 Mb	21.0 min	1.7 min/297.3 Mb	1705.1
3'	1	4,8,24	40	0.1 min/186.5 Mb	10.2 min	0.9 min/148.9 Mb	854.6
4	1	4,8,24	40	0.1 min/184.6 Mb	10.3 min	0.8 min/148.3 Mb	846.6
5	1	4,8,24	40	0.1 min/186.9 Mb	10.4 min	1.1 min/148.9 Mb	854.5
1	1	3,15,90	105	2.2 min/819.6 Mb	203.8 min	20.0 min/674.7 Mb	3826.9
2	1	3,15,90	105	4.4 min/1637.3 Mb	389.2 min	31.3 min/1352.5 Mb	7645.2
3	1	3,15,90	105	5.6 min/1945.7 Mb	561.2 min	61.1 min/821.2 Mb	9103.8
3'	1	3,15,90	105	2.9 min/974.0 Mb	286.7 min	61.1 min/410.9 Mb	4557.2
4	1	3,15,90	105	2.3 min/864.5 Mb	224.8 min	14.4 min/381.0 Mb	4042.8
5	1	3,15,90	105	3.2 min/981.1 Mb	291.4 min	342.5 min/463.4 Mb	4571.0
6	2	3,6,18	30	0.2 min/241.6 Mb	16.7 min	1.6 min/192.4 Mb	1098.9
7	2	3,6,18	30	0.2 min/244.9 Mb	17.3 min	1.8 min/194.4 Mb	1114.4
6	2	3,6,24	36	0.2 min/313.7 Mb	28.7 min	2.4 min/254.5 Mb	1432.0
7	2	3,6,24	36	0.2 min/317.6 Mb	31.0 min	2.7 min/257.5 Mb	1450.5
6	2	4,8,24	40	0.3 min/366.3 Mb	39.5 min	3.5 min/294.1 Mb	1675.3
7	2	4,8,24	40	0.3 min/371.5 Mb	38.2 min	3.8 min/297.0 Mb	1700.1
6	4	3,6,18	30	0.3 min/472.3 Mb	61.4 min	5.0 min/377.3 Mb	2145.6
7	4	3,6,18	30	0.3 min/475.5 Mb	62.3 min	5.3 min/379.3 Mb	2161.1
6	4	3,6,24	36	0.5 min/609.3 Mb	101.6 min	8.7 min/483.6 Mb	2777.7
7	4	3,6,24	36	0.5 min/613.2 Mb	103.1 min	9.2 min/486.2 Mb	2796.2
6	4	4,8,24	40	0.5 min/712.3 Mb	139.2 min	12.1 min/577.0 Mb	3254.6
7	4	4,8,24	40	0.6 min/717.5 Mb	141.0 min	12.6 min/580.3 Mb	3279.5

Table 2: Checking properties of the communication protocol.

6 Conclusion

In this paper, we introduced a verification technique to perform a partial verification of real-time systems modeled under a dense-time model and using mixed operational and descriptive components. The technique relies on discretization techniques introduced in previous work [16]. It is fully automated and implemented on top of a discrete-time bounded satisfiability checker. We experimented with a significant example based on the description of a communication protocol, where concurrent runs of the protocol are synchronized by means of additional MTL formulas, hence building a mixed model. Verification tests showed consistent results and significantly good performances.

References

[1] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[2] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, 1996.

[3] Rajeev Alur and Thomas A. Henzinger. Logics and models of real time: A survey. In J. W. de Bakker, Cornelis Huizing, and Willem P. de Roever, editors, *Proceedings of the Real-Time: Theory in Practice, REX Workshop*, volume 600 of *Lecture Notes in Computer Science*, pages 74–106. Springer-Verlag, 1992.

[4] Rajeev Alur and Thomas A. Henzinger. Real-time logics: Complexity and expressiveness. *Information and Computation*, 104(1):35–77, 1993.

[5] Dirk Beyer, Claus Lewerentz, and Andreas Noack. Rabbit: A tool for BDD-based verification of real-time systems. In Warren A. Hunt Jr. and Fabio Somenzi, editors, *Proceedings of the 15th International Conference on Computer Aided Verification (CAV'03)*, volume 2725 of *Lecture Notes in Computer Science*, pages 122–125. Springer-Verlag, 2003.

[6] Dragan Bošnački. Digitization of timed automata. In *Proceedings of the 4th International Workshop on Formal Methods for Industrial Critical Systems (FMICS'99)*, pages 283–302, 1999.

[7] Ahmed Bouajjani, Rachid Echahed, and Riadh Robbana. Verifying invariance properties of timed systems with duration variables. In *Proceedings of the 3rd International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'94)*, volume 863 of *Lecture Notes in Computer Science*, pages 193–210. Springer-Verlag, 1994.

[8] Marius Bozga, Oded Maler, and Stavros Tripakis. Efficient verification of timed automata using dense and discrete time semantics. In Laurence Pierre and Thomas Kropf, editors, *Proceedings of the 10th Correct Hardware Design and Verification Methods Advanced Research Working Conference (CHARME'99)*, volume 1703 of *Lecture Notes in Computer Science*, pages 125–141. Springer-Verlag, 1999.

[9] Gaurav Chakravorty and Paritosh K. Pandya. Digiziting interval duration logic. In Warren A. Hunt, Jr. and Fabio Somenzi, editors, *Proceedings of the 15th International Conference on Computer Aided Verification (CAV'03)*, volume 2725 of *Lecture Notes in Computer Science*, pages 167–179. Springer-Verlag, 2003.

[10] Edmund M. Clarke, Flavio Lerda, and Muralidhar Talupur. An abstraction technique for real-time verification. In *Proceedings of the GM R&D Workshop on Next Generation Design and Verification Methodologies for Distributed Embedded Control System*, 2007.

[11] Martin De Wulf, Laurent Doyen, and Jean-François Raskin. Almost ASAP semantics: from timed models to timed implementations. *Formal Aspects of Computing*, 17(3):319–341, 2005.

[12] D. D'Souza, R. Mohan M., and P. Prabhakar. Eliminating past operators in metric temporal logic. Technical Report IISc-CSA-TR-2006-11, 2006.

[13] Georgios E. Fainekos and George J. Pappas. Robust sampling for MITL specifications. In *Proc. of FORMATS'07*, volume 4763 of *LNCS*, 2007.

[14] Goran Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. In *Proceedings of the 5th International Workshop on Hybrid Systems: Computation and Control (HSCC'05)*, volume 3414 of *Lecture Notes in Computer Science*, pages 258–273. Springer-Verlag, 2005.

[15] Carlo A. Furia, Dino Mandrioli, Angelo Morzenti, and Matteo Rossi. Modeling time in computing: a taxonomy and a comparative survey. Technical Report 2007.22, Dipartimento di Elettronica e Informazione, Politecnico di Milano, January 2007.

[16] Carlo A. Furia, Matteo Pradella, and Matteo Rossi. Automated verification of dense-time MTL specifications via discrete-time approximation. In Jorge Cuéllar and Tom Maibaum, editors, *Proceedings of the 15th International Symposium on Formal Methods (FM'08)*, volume 5014 of *Lecture Notes in Computer Science*, pages 132–147. Springer-Verlag, May 2008.

[17] Carlo A. Furia and Matteo Rossi. Integrating discrete- and continuous-time metric temporal logics through sampling. In Eugene Asarin and Patricia Bouyer, editors, *Proceedings of the 4th International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS'06)*, volume 4202 of *Lecture Notes in Computer Science*, pages 215–229. Springer-Verlag, September 2006.

[18] Carlo A. Furia and Matteo Rossi. On the expressiveness of MTL variants over dense time. In Jean-François Raskin and P. S. Thiagarajan, editors, *Proceedings of the 5th International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS'07)*, volume 4763 of *Lecture Notes in Computer Science*, pages 163–178. Springer-Verlag, October 2007.

[19] Carlo Alberto Furia. *Scaling up the formal analysis of real-time systems*. PhD thesis, Dipartimento di Elettronica e Informazione, Politecnico di Milano, May 2007.

[20] Aleks Göllü, Anuj Puri, and Pravin Varaiya. Discretization of timed automata. In *Proceedings of the 33rd Conference on Decision and Control*, pages 957–958, 1994.

[21] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. HYTECH: A model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer*, 1(1–2), 1997.

[22] Thomas A. Henzinger, Zohar Manna, and Amir Pnueli. What good are digital clocks? In Werner Kuich, editor, *Proceedings of the 19th International Colloquium on Automata, Languages and Programming (ICALP'92)*, volume 623 of *Lecture Notes in Computer Science*, pages 545–558. Springer-Verlag, 1992.

[23] Thomas A. Henzinger, Jean-François Raskin, and Pierre-Yves Schobbens. The regular real-time languages. In Kim Guldstrand Larsen, Sven Skyum, and Glynn Winskel, editors, *Proceedings of the 25th International Colloquium on Automata, Languages and Programming (ICALP'98)*, volume 1443 of *Lecture Notes in Computer Science*, pages 580–591. Springer-Verlag, 1998.

[24] Dang Van Hung and Phan Hong Giang. Sampling semantics of Duration Calculus. In Joachim Parrow Bengt Jonsson, editor, *Proceedings of the 4th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'96)*, volume 1135 of *Lecture Notes in Computer Science*, pages 188–207. Springer-Verlag, 1996.

[25] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[26] Pavel Krčál and Radek Pelánek. On sampled semantics of timed systems. In R. Ramanujam and Sandeep Sen, editors, *Proceedings of the 25th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'05)*, volume 3821 of *Lecture Notes in Computer Science*, pages 310–321. Springer-Verlag, 2005.

[27] Kim G. Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1–2), 1997.

[28] Oded Maler, Dejan Nickovic, and Amir Pnueli. From MITL to timed automata. In Eugene Asarin and Patricia Bouyer, editors, *Proceedings of the 4th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'06)*, volume 4202 of *Lecture Notes in Computer Science*, pages 274–289. Springer-Verlag, 2006.

[29] Oded Maler and Amir Pnueli. Timing analysis of asynchronous circuits using timed automata. In Paolo Camurati and Hans Eveking, editors, *Proceedings of the Advanced Research Working Conference on Correct Hardware Design and Verification Methods*, volume 987 of *Lecture Notes in Computer Science*, pages 189–205. Springer-Verlag, 1995.

[30] Joël Ouaknine. Digitisation and full abstraction for dense-time model checking. In Joost-Pieter Katoen and Perdita Stevens, editors, *Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'02)*, volume 2280 of *Lecture Notes in Computer Science*, pages 37–51. Springer-Verlag, 2002.

[31] Joël Ouaknine and James Worrell. Revisiting digitization, robustness, and decidability for timed automata. In *Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science (LICS'03)*, pages 198–207. IEEE Computer Society Press, 2003.

[32] Matteo Pradella. ℤot. `http://home.dei.polimi.it/pradella`, March 2007.

[33] Matteo Pradella, Angelo Morzenti, and Pierluigi 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.

[34] Babita Sharma, Paritosh K. Pandya, and Supratik Chakraborty. Bounded validity checking of interval duration logic. In Nicolas Halbwachs and Lenore D. Zuck, editors, *Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'05)*, volume 3440 of *Lecture Notes in Computer Science*, pages 301–316. Springer-Verlag, 2005.

[35] Sergio Yovine. Kronos: A verification tool for real-time systems. *International Journal on Software Tools for Technology Transfer*, 1(1–2):123–133, 1997.