Science of Computer Programming 99 (2015) 128–192

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Sound and complete timed CTL model checking of timed Kripke structures and real-time rewrite theories

Daniela Lepri [a,*], Erika Ábrahám [b], Peter Csaba Ölveczky [a]

[a] University of Oslo, Norway
[b] RWTH Aachen University, Germany

HIGHLIGHTS

- TCTL model checker for (dense-)timed Kripke structures in a pointwise semantics.
- Reduce TCTL model checking from continuous semantics to pointwise semantics.
- Sound and complete TCTL model checker for time-robust Real-Time Maude models.

ARTICLE INFO

Article history:
Received 23 January 2013
Received in revised form 3 April 2014
Accepted 3 June 2014
Available online 24 June 2014

Keywords:
Timed Kripke structures
Rewriting logic
Timed CTL
Model checking
Real-Time Maude

ABSTRACT

In this paper we show that the satisfaction of timed CTL (TCTL) formulas under the natural *continuous* semantics for both discrete-time and dense-time timed Kripke structures can be reduced to a model-checking problem in the *pointwise* semantics for a large class of timed Kripke structures, which includes many discrete-event systems. We then present a TCTL model checking algorithm for the pointwise case. An important consequence of our results is that they together describe a sound and complete TCTL model checking procedure for time-robust real-time rewrite theories also for dense time domains. We have implemented such a TCTL model checker for Real-Time Maude. Our model checker provides for free a sound and complete TCTL model checker for subsets of modeling languages, such as Ptolemy II and (Synchronous) AADL, which have Real-Time Maude analysis integrated into their tool environments.

1. Introduction

Many formal models of real-time systems assume a dense or continuous time domain. For the well-known timed automata [1], such a dense-time model has a discrete abstraction which preserves key properties. However, such an abstraction may not be available for more expressive formalisms, such as, e.g., real-time rewrite theories [2], calendar automata [3], and Ptolemy II discrete-event models [4]. Many such models are typically executed as discrete-event systems: time is advanced until the next time point when an event is scheduled to take place. For *untimed* properties, it is usually reasonably straightforward to prove that this abstraction/execution strategy provides a sound and complete model checking procedure for the underlying dense-time system (see, e.g., [5]).

However, for the important class of *timed* properties—we focus on *timed CTL* (TCTL) properties [6] in this paper—this is typically not the case: only visiting those points in time when some discrete event takes place does *not* provide a sound and complete abstraction when we interpret TCTL formulas in a *continuous semantics*. It is enough to consider the following (dense-time or discrete-time) example: Assume that the first event in the system takes place after three time units. Now

* Corresponding author.

http://dx.doi.org/10.1016/j.scico.2014.06.006
0167-6423/

consider the property $A\ F_{[1,2]}\ true$ (or $E\ F_{[1,2]}\ true$): in each (resp., some) path from the current state we will reach a state in between 1 and 2 time units where *true* holds. In a *continuous interpretation* of validity (even if the time domain is discrete), the above properties should hold. Nevertheless, in an abstraction that always advances time until the next discrete event takes place (corresponding to a *pointwise interpretation* of validity even if the time domain is continuous), the (pointwise) model checking will not see any state in the time interval $[1, 2]$ and hence will say that $A\ F_{[1,2]}\ true$ does *not* hold.

In this paper, we study *timed Kripke structures*, which are Kripke structures where each transition has a duration; the duration is typically 0 for "real events" and non-zero for "tick transitions" that advance time. For finite timed Kripke structures, there exist TCTL model checking algorithms in the *pointwise semantics*; this paper presents one that refines the one in [7]. The main, until now unanswered, question is whether we can use such pointwise model checking to obtain a sound and complete model checking procedure for TCTL formulas interpreted in the *continuous semantics*. In this paper we answer this question affirmatively, under certain reasonable conditions. The idea is to take the given (finite) timed Kripke structure and "split" all timed transitions $s \xrightarrow{t} s'$ into a sequence of timed transitions $s \xrightarrow{\gamma} s_1 \xrightarrow{\gamma} s_2 \xrightarrow{\gamma} \cdots s_k \xrightarrow{\gamma} s'$ with the same total duration, and where each s_i is some state that satisfies the same atomic propositions that s does. The duration γ is the half of the greatest common divisor of all the following time values: all non-zero durations occurring in the timed Kripke structure and all non-zero finite time values that appear as bounds in the TCTL formula under consideration. For example, in dense time,[1] if all transitions in the timed Kripke structure have the form $s \xrightarrow{20} s'$ or $s \xrightarrow{12} s'$ or $s \xrightarrow{0} s'$, and the TCTL property under consideration is $A\ F_{[12,24]}\ p$, then we split up all transitions with duration > 0 into transitions with duration $\frac{gcd(\{12,20,24\})}{2} = 2$. We can now apply the pointwise model checking procedure on the resulting, still finite, timed Kripke structure. Our main result is that this provides a sound and complete model checking procedure in the continuous semantics under certain conditions that are satisfied by interesting classes of systems. In particular, (discrete-event) systems in which the time when the next "event" takes place is given deterministically, and where the elapse of time does not change the valuation of the atomic propositions satisfy these conditions. The latter condition almost always holds in practice, since time elapse only affects timers and clocks, whose values rarely influence the validity of an atomic proposition.

Since our soundness and completeness results—which reduce the model checking of a TCTL formula under the more natural continuous semantics to a TCTL model checking problem in the pointwise semantics, for which there are reasonably straightforward model checking algorithms—are given for timed Kripke structures which are parametric in the (discrete or dense) time domain, our results are fairly general. In particular, our results are independent of the formalism used to define the timed Kripke structure.

One important instance of our setting is the maximal-time-sampling-based execution of time-robust *real-time rewrite theories* [2,5], whose specification and analysis are supported by the Real-Time Maude tool [8]. Real-Time Maude extends Maude [9] to support the formal modeling and analysis of real-time systems in rewriting logic. Real-Time Maude is characterized by its expressiveness and generality, natural model of object-based distributed real-time systems, that supports multiple inheritance as well as dynamic creation and deletion of both objects and messages, the possibility to define any computable data type, and a range of automated formal analysis such as simulation, reachability and temporal logic model checking. This has made it possible to successfully apply the tool to a wide range of real-time systems, including advanced state-of-the-art wireless sensor network algorithms [10,11], multicast protocols [12,13], scheduling algorithms requiring unbounded queues [14], routing protocols [15], and large-scale transactional data stores [16,17].

Real-Time Maude's expressiveness and generality also make it a suitable semantic framework and analysis tool for modeling languages for real-time systems [18]. For example, the tool has been used to formalize (subsets of) the industrial avionics modeling standard AADL [19], a synchronous version of AADL [20,21], Ptolemy II discrete-event (DE) models [4], the web orchestration language Orc [22], different EMF-based timed model transformation frameworks [23,24], the actor-based language Timed Rebeca [25], etc. Real-Time Maude's formal analysis has been integrated into the tool environment of many of these languages, enabling a model engineering process that combines the convenience of an intuitive modeling language with formal analysis.

In Real-Time Maude, the data types of the system are defined by an algebraic equational specification, and the system's instantaneous transitions are modeled by (instantaneous) rewrite rules. Time advance is modeled explicitly by so-called *tick (rewrite) rules* of the form $\{t\}$ `=>` $\{t'\}$ `in time` u `if` *cond*, where `{_}` is an operator that encloses the entire global state, and the term u denotes the *duration* of the rewrite. Real-Time Maude is parametric in the time domain, which may be discrete or dense. For dense time (in particular), tick rules typically have the form $\{t\}$ `=>` $\{t'\}$ `in time` x `if` x `<=` d `/\` *cond*, where x is a new variable not occurring in t, d, or *cond*. This form of the tick rules ensures that any moment in time (within time d) *can* be visited, also for a dense time domain.

For dense time, it is of course not possible to execute all possible rewrite sequences. Instead, the general approach taken in Real-Time Maude is to use *time sampling strategies* to instantiate the new variable x in the tick rules, and to analyze the resulting specification. One such strategy advances time by a fixed amount Δ in each application of any tick rule. The *maximal* time sampling strategy advances time as much as possible in each application of a tick rule. Although the fixed-increment strategy can cover all possible behaviors in the original system when the time domain is discrete, the maximal time sampling typically only analyzes a *subset* of all the possible behaviors. However, in [5], it is shown that for a

[1] We do not fix the time domains, but approach them abstractly, since they are supposed to be user-defined.

fairly large set of *time-robust* real-time systems appearing in practice, the maximal time sampling strategy yields sound and complete analyses for *untimed* LTL properties.

Until recently, Real-Time Maude could only analyze *untimed* temporal logic properties, but not quantitative properties such as "the airbag must deploy within 5 ms of a crash," or "the ventilator machine cannot be turned off more than once every 10 minutes." As a consequence of our main result, we now have a sound and complete TCTL model checking procedure for time-robust real-time rewrite theories if the reachable state space is finite under the maximal time sampling strategy: from the given initial state, we construct the timed Kripke structure corresponding to the maximal time sampling strategy execution and then apply our sound and complete model checking procedure.

We have implemented such a TCTL model checker for Real-Time Maude. An important benefit of our work is that a TCTL model checker for Real-Time Maude also gives us a TCTL model checker *for free* for Ptolemy II DE models, Synchronous AADL models, Timed Rebeca models, and other modeling languages for which Real-Time Maude models can be generated. As shown in Section 8, our model checker has already been integrated into the Ptolemy II tool, allowing the user to model check TCTL properties of Ptolemy II models from within Ptolemy II.

This paper also describes our model checker, its semantic foundations, and its implementation in Maude. We illustrate the use of the model checker on three case studies: a Real-Time Maude model of a simple network of embedded medical devices, a Ptolemy II model of a railroad crossing system, and a Ptolemy II model of a fault-tolerant traffic light system.

A preliminary report on this work appeared in [26]. In addition to clarifying and extending the exposition and adding proofs and examples, the main additions in this paper are the following:

1. The paper [26] only dealt with the soundness and completeness for *real-time rewrite theories*, whereas we now have lifted our results to the more abstract and general *timed Kripke structure* setting.
2. In [26] we only had results for formulas where the time intervals were *closed* time intervals of the form $[a, b]$ with $a \leq b$, or $[a, \infty)$. We have now extended our soundness and completeness results to cover also formulas with *open* time intervals.
3. We now compare the performance of our model checker with other timed CTL model checkers on a well-known benchmark.

Section 2 gives our definitions of timed Kripke structures and satisfaction of TCTL formulas in the pointwise and in the continuous semantics. In Section 3, we first introduce our discrete TCTL-preserving abstraction of timed Kripke structures, the so-called *gcd*-transformation. We then prove that the *gcd*-transformation is a sound and complete abstraction for timed Kripke structures w.r.t. model checking TCTL formulas whose time intervals are *closed* time intervals of the form $[a, b]$, or $[a, \infty)$. In particular, model checking the abstraction in the pointwise semantics is equivalent to model checking the original timed Kripke structure in the continuous semantics. We then define an extended transformation that achieves soundness and completeness for the whole TCTL logic (including *open* time intervals in the temporal operators). Section 4 describes our model checking procedure for TCTL formulas over timed Kripke structures in the pointwise semantics, and proves its complexity and correctness. Section 5 gives a brief background to real-time rewrite theories and Real-Time Maude. Section 6 shows how our sound and complete TCTL abstraction for timed Kripke structures can be used to achieve a sound and complete TCTL abstraction for a large class of real-time rewrite theories. Section 7 discusses the implementation of our TCTL model checker for Real-Time Maude. Section 8 demonstrates the applicability of our TCTL model checker for Real-Time Maude on some case studies. Section 9 compares the performance of our TCTL model checker with the tools RED and TSMV. Section 10 discusses the related works. Finally, concluding remarks are given in Section 11.

2. Timed Kripke structures and timed computation tree logic

2.1. Time domains

A time domain can be either discrete, such as the natural numbers $\mathbb{N}$, or dense, such as the non-negative rational numbers $\mathbb{Q}_{\geq 0}$. We do not restrict to a particular time domain, but give a general algebraic abstract specification of time and require a time domain to be a valid interpretation of this specification.

Time is defined abstractly as a linearly ordered commutative monoid $(\mathit{Time}, +, 0, <)$ with additional operators, such as $\dot{-}$, where $\tau \dot{-} \tau'$ denotes $\tau - \tau'$ if $\tau' < \tau$, and 0 otherwise, and *max*, where $\mathit{max}(\tau, \tau') = \tau'$ if $\tau < \tau'$ and τ otherwise. The theory *TIME*, given in Appendix A, axiomatizes the time domain. We assume that $\mathcal{T}$ is a time domain satisfying the theory *TIME*. To simplify the notation, we write $\mathcal{T}$ for both the algebra and the carrier of the algebra, and write $0, +, \ldots$ for the interpretations $0_{\mathcal{T}}, +_{\mathcal{T}}, \ldots$ of $0, +, \ldots$ in $\mathcal{T}$. We use τ, τ', τ_1, ... to denote time values, we write $\tau_1 < \tau < \tau_2$ when $\tau_1 < \tau \wedge \tau < \tau_2$, by $\tau_1 \leq \tau_2$ we abbreviate $\tau_1 < \tau_2 \vee \tau_1 = \tau_2$, and write $k\tau$ or $k \cdot \tau$ for $\underbrace{\tau + \tau + \cdots + \tau}_{k \text{ times}}$. Furthermore, we denote $\mathit{Time} \setminus \{0\}$ by $\mathit{Time}_{>0}$.

The theory TIME_∞, also shown in Appendix A, extends *TIME* with the new infinity element $\infty \notin \mathit{Time}$, such that $\mathit{Time}_\infty = \mathit{Time} \cup \{\infty\}$ and the operators $<$, $+$, $\dot{-}$, *max* and *min* are extended to Time_∞ in the expected way. We denote by $\mathcal{T}_\infty = \mathcal{T} \uplus \{\infty\}$ the time domain $\mathcal{T}$ extended with the infinity element, so that $\mathcal{T}_\infty$ satisfies the theory TIME_∞.

A *time interval* is a non-empty interval of the form $[a, b]$, $(a, b]$, $[a, b_\infty)$ or (a, b_∞), where $a, b \in \mathcal{T}$ and $b_\infty \in \mathcal{T}_\infty$. We denote by $\mathit{Intervals}(\mathcal{T})$ the set of all time intervals in $\mathcal{T}$. Given a time interval $I \in \mathit{Intervals}(\mathcal{T})$, we define:

Table 1
Axioms for the theory $TIME^{gcd}$, where $\tau_1, \tau_2, \tau_3 \in Time_{>0}$.

2.1.	$\tau_1 \mid \tau_2 \wedge \tau_2 \mid \tau_3 \Longrightarrow \tau_1 \mid \tau_3$	2.7.	$gcd(\tau_1, \tau_2) = gcd(\tau_2, \tau_1)$
2.2.	$\tau_1 \mid \tau_2 \wedge \tau_2 \mid \tau_1 \Longrightarrow \tau_1 = \tau_2$	2.8.	$gcd(gcd(\tau_1, \tau_2), \tau_3) = gcd(\tau_1, gcd(\tau_2, \tau_3))$
2.3.	$\tau_1 \mid \tau_1$	2.9.	$gcd(\tau_1, \tau_2) \mid \tau_1$
2.4.	$\tau_1 \mid \tau_2 \wedge \tau_1 \mid \tau_3 \Longrightarrow \tau_1 \mid (\tau_2 + \tau_3)$	2.10.	$(\tau_3 \mid \tau_1 \wedge \tau_3 \mid \tau_2) \Longrightarrow \tau_3 \mid gcd(\tau_1, \tau_2)$
2.5.	$\tau_2 < \tau_1 \Longrightarrow \neg(\tau_1 \mid \tau_2)$	2.11.	$half(\tau_1) + half(\tau_1) = \tau_1$
2.6.	$\tau_1 \mid (\tau_1 + \tau_2) \Longrightarrow \tau_1 \mid \tau_2$		

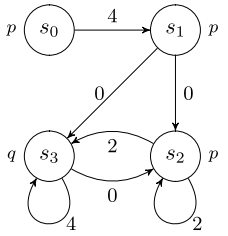


Fig. 1. A timed Kripke structure.

$$inf(I) = max\{\tau \in \mathcal{T}_\infty \mid \forall \tau' \in I.\ \tau \leq \tau'\} \quad \text{to be the } \textit{infimum} \text{ of } I;$$

$$sup(I) = min\{\tau \in \mathcal{T}_\infty \mid \forall \tau' \in I.\ \tau' \leq \tau\} \quad \text{to be the } \textit{supremum} \text{ of } I.$$

Notice that an interval $I \in Intervals(\mathcal{T})$ satisfies $(\forall \tau \in \mathcal{T}.\ inf(I) < \tau < sup(I) \Longrightarrow \tau \in I)$.

The theory $TIME^{gcd}$ extends $TIME$ with the functions

$$\mid : Time_{>0} \times Time_{>0} \to Bool$$

$$gcd : Time_{>0} \times Time_{>0} \to Time_{>0}$$

$$half : Time_{>0} \to Time_{>0}$$

satisfying the axioms in Table 1 for each $\tau_1, \tau_2, \tau_3 \in Time_{>0}$, where $Bool$ is the usual boolean domain. Intuitively, $\tau_1 \mid \tau_2$ ("τ_1 divides τ_2" or, equivalently, "τ_1 is a divisor of τ_2") is true if adding up τ_1 for a finite number of times gives τ_2 (see Axiom 2.6 in Table 1); $gcd(\tau_1, \tau_2)$ denotes the greatest common divisor of τ_1 and τ_2;[2] and $half(\tau_1)$ denotes the half of τ_1, that is $2 \cdot half(\tau_1) = \tau_1$. For example, $\mathbb{Q}_{\geq 0}$ satisfies this theory with the standard interpretation of divisors and the greatest common divisor operator. The theory $TIME^{gcd}_\infty$ extends $TIME^{gcd}$ with the infinity element ∞.

2.2. Timed Kripke structures

There are a number of different timed extensions of Kripke structures (KSs) in the literature (e.g., [27–31,7]). In this paper we use *timed Kripke structures*, which are Kripke structures whose transitions are annotated with *durations*.

Definition 2.1 *(Timed Kripke structure).* Given a set AP of atomic propositions, a *timed Kripke structure over AP* is a tuple $\mathcal{TK} = (S, \mathcal{T}, \longrightarrow, L)$, where S is a set of *states*, $\mathcal{T}$ is a *time domain* satisfying the theory $TIME$, $\longrightarrow \subseteq S \times \mathcal{T} \times S$ is a *total transition relation with duration*[3] (i.e., for each $s \in S$ there exist τ and s' such that $(s, \tau, s') \in \longrightarrow$), and L is a *labeling function* $L : S \to \mathcal{P}(AP)$. We write $s \xrightarrow{\tau} s'$ if $(s, \tau, s') \in \longrightarrow$. We call $s \xrightarrow{0} s'$ an *instantaneous* transition and $s \xrightarrow{\tau} s'$ with $\tau > 0$ a *tick* transition. $\mathcal{TK}$ is *finite* iff $\longrightarrow$ is finite.

Intuitively, our definition of timed Kripke structure generalizes the *tight durational transition graphs* presented in [7]—where the time domain is $\mathbb{N}$—to any time domain as described in Section 2.1.

Example 2.1. A timed Kripke structure is shown in Fig. 1, where circles represent states and arrows represent transitions between states. Each state is labeled with the atomic propositions holding in that state (i.e., p holds in the states s_0, s_1 and s_2, whereas q holds in the state s_3), and transitions are annotated with their duration.

In the rest of this section, let AP be a set of atomic propositions and $\mathcal{TK} = (S, \mathcal{T}, \longrightarrow, L)$ a timed Kripke structure over AP.

[2] Notice that the greatest common divisor of two time values $\tau_1, \tau_2 > 0$ always exists.

[3] A transition relation $\longrightarrow$ can be made *total* by defining $(\longrightarrow)^\bullet = \longrightarrow \cup \{(s, 0, s) \in S \times \mathcal{T} \times S \mid \forall (s', \tau, s'') \in \longrightarrow .\ s \neq s'\}$.

Intuitively, a transition $s_0 \xrightarrow{\tau} s_1$ means that it takes τ time units for the system to move from state s_0 to s_1. Consider the property "it is possible to reach time 2", or, more precisely, "it is possible to reach a state in time 2 where *true* holds." Does this property hold in state s_0 in the timed Kripke structure shown in Fig. 1? Furthermore, what about the property "in *each* path there is a state at time 2"?

There are at least two possible ways in which a transition $s_0 \xrightarrow{\tau} s_1$, with $\tau > 0$, can be interpreted in a timed Kripke structure: it can be seen as an "atomic tick step" of duration τ that *jumps directly* from s_0 to s_1 without "visiting" any intermediate time in between (we refer to this interpretation as the *pointwise* interpretation); or we can see it as a *continuous* time advance from state s_0 to s_1 (we refer to this interpretation as the *continuous* interpretation). To formalize these concepts we introduce the notion of *configuration* of a timed Kripke structure.

Definition 2.2. A *configuration* $\langle s, \delta \rangle \in S \times \mathcal{T}$ of $\mathcal{TK}$ specifies that the system has been in state s for time δ.

We can now interpret the behaviors of timed Kripke structures over possible configurations.

Definition 2.3 *(Interpretations of a timed Kripke structure).* The *continuous interpretation* of $\mathcal{TK}$ is given by the one-step transition relation $\hookrightarrow \subseteq (S \times \mathcal{T}) \times \mathcal{T} \times (S \times \mathcal{T})$ defined by the following rules:

$$\frac{s \xrightarrow{\tau} s' \quad \delta + \tau' < \tau}{\langle s, \delta \rangle \overset{\tau'}{\hookrightarrow} \langle s, \delta + \tau' \rangle}\ \mathtt{Rule}_{\mathtt{tick1}} \qquad \frac{s \xrightarrow{\tau} s' \quad \delta + \tau' = \tau}{\langle s, \delta \rangle \overset{\tau'}{\hookrightarrow} \langle s', 0 \rangle}\ \mathtt{Rule}_{\mathtt{tick2}} \qquad \frac{s \xrightarrow{0} s'}{\langle s, 0 \rangle \overset{0}{\hookrightarrow} \langle s', 0 \rangle}\ \mathtt{Rule}_{\mathtt{inst}}$$

where $\tau, \tau' > 0$. The *pointwise interpretation* of $\mathcal{TK}$ is given by the operational semantics consisting of the rules $\mathtt{Rule}_{\mathtt{tick2}}$ and $\mathtt{Rule}_{\mathtt{inst}}$.

The rules $\mathtt{Rule}_{\mathtt{tick1}}$ and $\mathtt{Rule}_{\mathtt{tick2}}$ define *tick steps* which let time progress, whereas rule $\mathtt{Rule}_{\mathtt{inst}}$ specifies *instantaneous steps*. We say that tick steps defined by rule $\mathtt{Rule}_{\mathtt{tick2}}$ are *maximal*, since they let time elapse by the maximal possible duration of the transition taken. In contrast, $\mathtt{Rule}_{\mathtt{tick1}}$ lets time elapse in a state $s \in S$ by a value less than the duration of the longest tick step from s. The *pointwise interpretation* of a timed Kripke structure allows instantaneous and maximal tick steps only, therefore all reachable configurations $\langle s, \delta \rangle$ have $\delta = 0$.

Definition 2.4. Given a state $s \in S$, we denote by $\mathcal{T}_{\mathcal{TK}}(s) \subset \mathcal{T}$ the set $\{\tau \in \mathcal{T} \mid \exists s' \in S.\ s \xrightarrow{\tau} s' \wedge 0 < \tau\}$ of positive tick step durations from s, and we define the sets $\mathcal{C}^p_{\mathcal{TK}} \subseteq S \times \mathcal{T}$ resp. $\mathcal{C}^c_{\mathcal{TK}} \subseteq S \times \mathcal{T}$ of *valid configurations* of $\mathcal{TK}$ in the *pointwise* resp. *continuous* interpretation as follows:

$$\mathcal{C}^p_{\mathcal{TK}} = \{\langle s, 0 \rangle \mid s \in S\}$$
$$\mathcal{C}^c_{\mathcal{TK}} = \mathcal{C}^p_{\mathcal{TK}} \cup \{\langle s, \delta \rangle \mid \exists \tau \in \mathcal{T}_{\mathcal{TK}}(s).\ \delta < \tau\}.$$

Example 2.2. If $\mathbb{Q}_{\geq 0}$ is the time domain, valid configurations of the timed Kripke structure in Fig. 1, when interpreted continuously (i.e., configurations in $\mathcal{C}^c_{\mathcal{TK}}$), are, for example, $\langle s_0, 0 \rangle$, $\langle s_0, 1/2 \rangle$, $\langle s_0, 3 \rangle$, $\langle s_1, 0 \rangle$, etc., while $\langle s_0, 4 \rangle$ is not a valid configuration. The set of valid configurations in the pointwise interpretation is $\mathcal{C}^p_{\mathcal{TK}} = \{\langle s_0, 0 \rangle, \langle s_1, 0 \rangle, \langle s_2, 0 \rangle, \langle s_3, 0 \rangle\}$.

Definition 2.5. A *path* π of $\mathcal{TK}$ in the *pointwise* interpretation is an infinite sequence of steps

$$\langle s_0, 0 \rangle \overset{\tau_0}{\hookrightarrow} \langle s_1, 0 \rangle \overset{\tau_1}{\hookrightarrow} \langle s_2, 0 \rangle \overset{\tau_2}{\hookrightarrow} \cdots,$$

where $\langle s_0, 0 \rangle \in \mathcal{C}^p_{\mathcal{TK}}$ and each $\langle s_i, 0 \rangle \overset{\tau_i}{\hookrightarrow} \langle s_{i+1}, 0 \rangle$ is a step allowed in the pointwise interpretation of $\mathcal{TK}$. We define $s^\pi_i = s_i$, and $c^\pi_i = \sum_{j=0}^{i-1} \tau_j$.

Definition 2.6. A *path* π of $\mathcal{TK}$ in the *continuous* interpretation is an infinite sequence of steps

$$\langle s_0, \delta_0 \rangle \overset{\tau_0}{\hookrightarrow} \langle s_1, \delta_1 \rangle \overset{\tau_1}{\hookrightarrow} \langle s_2, \delta_2 \rangle \overset{\tau_2}{\hookrightarrow} \cdots,$$

where $\langle s_0, \delta_0 \rangle \in \mathcal{C}^c_{\mathcal{TK}}$ and each $\langle s_i, \delta_i \rangle \overset{\tau_i}{\hookrightarrow} \langle s_{i+1}, \delta_{i+1} \rangle$ is a step allowed in the continuous interpretation of $\mathcal{TK}$. We define s^π_i and c^π_i as in Definition 2.5, and define $\delta^\pi_i = \delta_i$.

We write $Paths^p_{\mathcal{TK}}(\langle s, 0 \rangle)$ to denote the set of all paths in the pointwise interpretation of $\mathcal{TK}$ starting in a valid configuration $\langle s, 0 \rangle \in \mathcal{C}^p_{\mathcal{TK}}$. Similarly, we write $Paths^c_{\mathcal{TK}}(\langle s, \delta \rangle)$ to denote the set of all paths in the continuous interpretation of $\mathcal{TK}$ starting in $\langle s, \delta \rangle \in \mathcal{C}^c_{\mathcal{TK}}$.

Example 2.3. For the time domain $\mathbb{Q}_{\geq 0}$, in state s_0 of the timed Kripke structure in Fig. 1, the above property "it is possible to reach a state in time 2 where *true* holds" is true in the continuous interpretation, but does not hold in the pointwise one, since time 2 is never "reached" in this semantics.

Notice that timed Kripke structures have an "eager" semantics, in the sense that the system can not idle in a state s_i (unless there is a transition $s_i \xrightarrow{\tau} s_i$).

In general, the continuous interpretation of a timed Kripke structure allows also *Zeno* paths.

Definition 2.7. A path π of $\mathcal{TK}$ is *time-divergent* if for each $\tau \in \mathcal{T}$ there is an $i \in \mathbb{N}$ such that $c_i^\pi > \tau$, and is *time-convergent* or *Zeno* otherwise.

Example 2.4. Consider the timed Kripke structure in Fig. 1 with the time domain $\mathbb{Q}_{\geq 0}$. When interpreted continuously, the following path is possible:

$$\langle s_0, 0\rangle \overset{1}{\hookrightarrow} \langle s_0, 1\rangle \overset{1/2}{\hookrightarrow} \langle s_0, 1+1/2\rangle \overset{1/4}{\hookrightarrow} \langle s_0, 1+1/2+1/4\rangle \overset{1/8}{\hookrightarrow} \cdots.$$

This path is Zeno, i.e., it has a finite duration such that time 2 is never reached.

Although Zeno paths model unrealizable behavior, paths like the one in the above example cannot be excluded by proper modeling. However, paths that are not only Zeno but perform even infinitely many *instantaneous* steps in finite time should be considered as a modeling flaw. In the following we will restrict to so-called *Zeno-free* timed Kripke structures that do not have such paths, i.e., whose Zeno paths all execute finitely many instantaneous steps in finite time.

Fact 2.1. *A finite timed Kripke structure is Zeno-free if and only if it does not contain any loops consisting of instantaneous transitions only (zero-loops).*

The notion of reachability (see below) is restricted to configurations that are reachable via time-divergent paths. We write $tdPaths^p_{\mathcal{TK}}(\langle s, 0\rangle)$ to denote the set of all time-divergent paths in the pointwise interpretation of $\mathcal{TK}$ starting in a valid configuration $\langle s, 0\rangle \in \mathcal{C}^p_{\mathcal{TK}}$. Similarly, we write $tdPaths^c_{\mathcal{TK}}(\langle s, \delta\rangle)$ to denote the set of all time-divergent paths in the continuous interpretation of $\mathcal{TK}$ starting in $\langle s, \delta\rangle \in \mathcal{C}^c_{\mathcal{TK}}$.

A configuration $\langle s', 0\rangle \in \mathcal{C}^p_{\mathcal{TK}}$ is *reachable* from $\langle s, 0\rangle \in \mathcal{C}^p_{\mathcal{TK}}$ in time τ in the pointwise interpretation iff there exists a path $\pi \in tdPaths^p_{\mathcal{TK}}(\langle s, 0\rangle)$ and some $i \in \mathbb{N}$ such that $s_i^\pi = s'$ and $c_i^\pi = \tau$. Similarly, a configuration $\langle s', \delta'\rangle \in \mathcal{C}^c_{\mathcal{TK}}$ is *reachable* from $\langle s, \delta\rangle \in \mathcal{C}^c_{\mathcal{TK}}$ in time τ in the continuous interpretation iff there exists a path $\pi \in tdPaths^c_{\mathcal{TK}}(\langle s, \delta\rangle)$ and some $i \in \mathbb{N}$ such that $s_i^\pi = s'$, $\delta_i^\pi = \delta'$ and $c_i^\pi = \tau$. A state $s' \in S$ is reachable from $s \in S$ iff there exist two time values $\delta', \tau \in \mathcal{T}$ such that the configuration $\langle s', \delta'\rangle$ is valid and reachable from $\langle s, 0\rangle$ in time τ.

Fact 2.2. $Paths^p_{\mathcal{TK}}(\langle s, 0\rangle) \subseteq Paths^c_{\mathcal{TK}}(\langle s, 0\rangle)$ *and* $tdPaths^p_{\mathcal{TK}}(\langle s, 0\rangle) \subseteq tdPaths^c_{\mathcal{TK}}(\langle s, 0\rangle)$.

In the rest of this paper, we sometimes omit the superscripts "c" or "p" and write $\mathcal{C}_{\mathcal{TK}}$, $Paths_{\mathcal{TK}}$, etc., when we intend a definition or statement to hold for *both* interpretations. We also omit the subscript $\mathcal{TK}$ when it is clear from the context.

Configurations and paths can be annotated with the global time, resulting in *timed configurations* and *timed paths*.

Definition 2.8. The set $\mathcal{TC}_{\mathcal{TK}} \subseteq S \times \mathcal{T} \times \mathcal{T}$ of *timed configurations* of $\mathcal{TK}$ is defined as $\mathcal{TC}_{\mathcal{TK}} = \{(s, \delta, c) \mid \langle s, \delta\rangle \in \mathcal{C}_{\mathcal{TK}} \text{ and } c \in \mathcal{T}\}$. A timed configuration (s, δ, c) is denoted by $\langle s, \delta\rangle @c$. Let $\pi \in Paths_{\mathcal{TK}}(\langle s_0, \delta_0\rangle)$, for some $\langle s_0, \delta_0\rangle \in \mathcal{C}_{\mathcal{TK}}$, be the path $\langle s_0, \delta_0\rangle \overset{\tau_0}{\hookrightarrow} \langle s_1, \delta_1\rangle \overset{\tau_1}{\hookrightarrow} \ldots$. Then the corresponding *timed path* $@\pi$ is denoted by

$$\langle s_0, \delta_0\rangle @c_0 \overset{\tau_0}{\hookrightarrow} \langle s_1, \delta_1\rangle @c_1 \overset{\tau_1}{\hookrightarrow} \ldots$$

with $c_i = c_i^\pi$ for all $i \in \mathbb{N}$. We use notions defined for configurations analogously for timed configurations, e.g., we say that a timed path $@\pi$ is *time-divergent (time-convergent)* if its corresponding path π is time-divergent (time-convergent).

By definition, timed paths start with the global time equal to 0, i.e., $c_0 = 0$. As for configurations, in the pointwise interpretation of a timed Kripke structure, all reachable timed configurations $\langle s, \delta\rangle @c$ have $\delta = 0$.

Example 2.5. The timed path corresponding to the path

$$\langle s_0, 0\rangle \overset{1/2}{\hookrightarrow} \langle s_0, 1/2\rangle \overset{7/2}{\hookrightarrow} \langle s_1, 0\rangle \overset{0}{\hookrightarrow} \langle s_2, 0\rangle \overset{1}{\hookrightarrow} \langle s_2, 1\rangle \overset{1}{\hookrightarrow} \cdots$$

is the following path of timed configurations:

$$\langle s_0, 0\rangle @0 \overset{1/2}{\hookrightarrow} \langle s_0, 1/2\rangle @1/2 \overset{7/2}{\hookrightarrow} \langle s_1, 0\rangle @4 \overset{0}{\hookrightarrow} \langle s_2, 0\rangle @4 \overset{1}{\hookrightarrow} \langle s_2, 1\rangle @5 \overset{1}{\hookrightarrow} \cdots.$$

A *position* in a path is a configuration on the path together with the time of its occurrence, i.e., it is a timed configuration on the corresponding timed path. According to the different interpretations, we distinguish between pointwise and continuous positions.

Definition 2.9. Assume a path $\pi = \langle s_0, \delta_0\rangle \overset{\tau_0}{\hookrightarrow} \langle s_1, \delta_1\rangle \overset{\tau_1}{\hookrightarrow} \ldots \in Paths_{\mathcal{TK}}(\langle s_0, \delta_0\rangle)$ starting in some configuration $\langle s_0, \delta_0\rangle \in \mathcal{C}_{\mathcal{TK}}$. A *pointwise position* in π is any timed configuration $\langle s_i, \delta_i\rangle @c_i^\pi$ with $i \in \mathbb{N}$. For each pointwise position $\langle s_i, \delta_i\rangle @c_i^\pi$ in π we define the set

$$pre^p_\pi(\langle s_i, \delta_i\rangle @c_i^\pi) = \{\langle s_j, \delta_j\rangle @c_j^\pi \mid j \in \mathbb{N} \text{ and } 0 \le j < i\}$$

of its *pointwise predecessor positions*. A *continuous position* in π is any timed configuration $\langle s_i, \delta_i + \tau\rangle @c_i^\pi + \tau$ with $i \in \mathbb{N}$ and $0 \le \tau < \tau_i$. For each continuous position $\langle s_i, \delta_i + \tau\rangle @c_i^\pi + \tau$ in π we define the set

$$pre^c_\pi(\langle s_i, \delta_i + \tau\rangle @c_i^\pi + \tau) = \{\langle s_j, \delta_j + \tau'\rangle @c_j^\pi + \tau' \mid (i = j \text{ and } 0 \le \tau' < \tau) \text{ or } (0 \le j < i \text{ and } 0 \le \tau' < \tau_j)\}$$

of its *continuous predecessor positions*. The *pointwise* (resp., *continuous*) positions in $@\pi$ are the pointwise (resp., continuous) positions in π.

Example 2.6. Consider the path π from Example 2.5. The pointwise positions in π are $\langle s_i, \delta_i\rangle @c_i$. Continuous positions in π include the timed configurations $\langle s_0, 1/4\rangle @1/4$ and $\langle s_0, 3/2\rangle @3/2$, the first one being a (continuous) predecessor position of the second one.

2.3. Timed computation tree logic

Several timed extensions of temporal logics have been proposed (see [32,33] for an overview). We use *timed CTL* (*TCTL*) [6], an extension of CTL [34] with interval time constraints on temporal operators, to specify properties of timed Kripke structures.

Definition 2.10. Given a set *AP* of atomic propositions and a time domain $\mathcal{T}$, TCTL formulas φ are built using the following grammar:

$$\begin{aligned} \varphi &::= true \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid E\,\psi \mid A\,\psi \\ \psi &::= \varphi\, U_I\, \varphi \end{aligned}$$

where $p \in AP$ and $I \in Intervals(\mathcal{T})$.

Intuitively, the *path formula* $\psi = \varphi_1\, U_I\, \varphi_2$ holds for a given path π if φ_2 holds at a position $\langle s, \delta\rangle @c$ in π occurring at time $c \in I$ and φ_1 holds for all predecessor positions. The existentially quantified *state formula* $E\,\psi$ holds in a given configuration if there exists some path that starts from that configuration and satisfies ψ, while the universally quantified state formula $A\,\psi$ holds in a given configuration if ψ holds in each path that starts from that configuration.

We omit the bound $[0, \infty)$ as subscript and we write $=a$, $\le b$, $<b$, $\ge a$ and $>a$ for $[a, a]$, $[0, b]$, $[0, b)$, $[a, \infty)$ and (a, ∞), respectively. As syntactic sugar we use the following common abbreviations for boolean and modal operators:

$$\begin{aligned} false &\overset{def}{=} \neg true \\ \varphi_1 \vee \varphi_2 &\overset{def}{=} \neg(\neg\varphi_1 \wedge \neg\varphi_2) \\ \varphi_1 \Longrightarrow \varphi_2 &\overset{def}{=} \neg\varphi_1 \vee \varphi_2 \\ \varphi_1 \Longleftrightarrow \varphi_2 &\overset{def}{=} (\varphi_1 \Longrightarrow \varphi_2) \wedge (\varphi_2 \Longrightarrow \varphi_1) \\ F_I\,\varphi &\overset{def}{=} true\, U_I\, \varphi \\ E\,G_I\,\varphi &\overset{def}{=} \neg A\,F_I\,\neg\varphi \\ A\,G_I\,\varphi &\overset{def}{=} \neg E\,F_I\,\neg\varphi \end{aligned}$$

We denote by $TCTL_{cb}$ the fragment of TCTL without open finite bounds, i.e., where all time intervals are of the form $[a, b]$ with $a \le b$, or $[a, \infty)$. The *universal* and the *existential* fragments, TACTL and TECTL, respectively, of TCTL [35] are defined by allowing negation only in front of atomic propositions and by restricting quantification to the universal quantifier and to the existential quantifier, respectively.

We can now define both a *pointwise* and a *continuous* semantics for TCTL formulas.

Definition 2.11 *(TCTL pointwise semantics).* For a timed Kripke structure $\mathcal{TK} = (S, \mathcal{T}, \longrightarrow, L)$, a valid configuration of $\mathcal{TK}$ in the pointwise interpretation $\langle s, 0\rangle \in \mathcal{C}^p_{\mathcal{TK}}$, and a TCTL formula φ, the *pointwise satisfaction relation* $\mathcal{TK}, \langle s, 0\rangle \models_p \varphi$ is defined inductively as follows:

$\mathcal{TK}, \langle s, 0\rangle \models_p true$		always holds
$\mathcal{TK}, \langle s, 0\rangle \models_p p$	iff	$p \in L(s)$
$\mathcal{TK}, \langle s, 0\rangle \models_p \neg\varphi_1$	iff	$\mathcal{TK}, \langle s, 0\rangle \not\models_p \varphi_1$
$\mathcal{TK}, \langle s, 0\rangle \models_p \varphi_1 \wedge \varphi_2$	iff	$\mathcal{TK}, \langle s, 0\rangle \models_p \varphi_1$ and $\mathcal{TK}, \langle s, 0\rangle \models_p \varphi_2$
$\mathcal{TK}, \langle s, 0\rangle \models_p E\ \psi$	iff	$\mathcal{TK}, \pi \models_p \psi$ for some $\pi \in tdPaths^p_{\mathcal{TK}}(\langle s, 0\rangle)$
$\mathcal{TK}, \langle s, 0\rangle \models_p A\ \psi$	iff	$\mathcal{TK}, \pi \models_p \psi$ for all $\pi \in tdPaths^p_{\mathcal{TK}}(\langle s, 0\rangle)$
$\mathcal{TK}, \pi \models_p \varphi_1\ U_I\ \varphi_2$	iff	there is a pointwise position $\langle s'', 0\rangle @ c''$ in π s.t. $c'' \in I$, $\mathcal{TK}, \langle s'', 0\rangle \models_p \varphi_2$, and $\mathcal{TK}, \langle s', 0\rangle \models_p \varphi_1$ for all pointwise positions $\langle s', 0\rangle @ c'$ in $pre^p_\pi(\langle s'', 0\rangle @ c'')$.

We define $\mathcal{TK}, s \models_p \varphi$ iff $\mathcal{TK}, \langle s, 0\rangle \models_p \varphi$.

Definition 2.12 *(TCTL continuous semantics).* Given $\mathcal{TK}$ as above, $\langle s, \delta\rangle \in \mathcal{C}^c_{\mathcal{TK}}$ a valid configuration in the continuous interpretation of $\mathcal{TK}$, and a TCTL formula φ. We define the *continuous satisfaction relation* $\models_c$ similarly to $\models_p$, but using the notion of continuous position instead of pointwise position, and $tdPaths^c_{\mathcal{TK}}(\langle s, d\rangle)$ instead of $tdPaths^p_{\mathcal{TK}}(\langle s, d\rangle)$:

$\mathcal{TK}, \langle s, \delta\rangle \models_c true$		always holds
$\mathcal{TK}, \langle s, \delta\rangle \models_c p$	iff	$p \in L(s)$
$\mathcal{TK}, \langle s, \delta\rangle \models_c \neg\varphi_1$	iff	$\mathcal{TK}, \langle s, \delta\rangle \not\models_c \varphi_1$
$\mathcal{TK}, \langle s, \delta\rangle \models_c \varphi_1 \wedge \varphi_2$	iff	$\mathcal{TK}, \langle s, \delta\rangle \models_c \varphi_1$ and $\mathcal{TK}, \langle s, \delta\rangle \models_c \varphi_2$
$\mathcal{TK}, \langle s, \delta\rangle \models_c E\ \psi$	iff	$\mathcal{TK}, \pi \models_c \psi$ for some $\pi \in tdPaths^c_{\mathcal{TK}}(\langle s, \delta\rangle)$
$\mathcal{TK}, \langle s, \delta\rangle \models_c A\ \psi$	iff	$\mathcal{TK}, \pi \models_c \psi$ for all $\pi \in tdPaths^c_{\mathcal{TK}}(\langle s, \delta\rangle)$
$\mathcal{TK}, \pi \models_c \varphi_1\ U_I\ \varphi_2$	iff	there is a continuous position $\langle s'', \delta''\rangle @ c''$ in π s.t. $c'' \in I$, $\mathcal{TK}, \langle s'', \delta''\rangle \models_c \varphi_2$, and $\mathcal{TK}, \langle s', \delta'\rangle \models_c \varphi_1$ for all continuous positions $\langle s', \delta'\rangle @ c'$ in $pre^c_\pi(\langle s'', \delta''\rangle @ c'')$.

We define $\mathcal{TK}, s \models_c \varphi$ iff $\mathcal{TK}, \langle s, 0\rangle \models_c \varphi$.

Our definitions of TCTL pointwise and continuous semantics intuitively correspond to the pointwise and continuous semantics for timed automata as defined in [36].

Intuitively, in the pointwise semantics, the validity of a formula is checked only right after a discrete event in the system, while, in the continuous semantics, the validity is checked continuously through a tick step at all possible times. In particular, in the continuous semantics, a tick transition $s \xrightarrow{\tau} s'$ requires us to consider any possible "splitting" $\langle s, 0\rangle \overset{\tau_0}{\hookrightarrow} \langle s, \tau_0\rangle \overset{\tau_1}{\hookrightarrow} \cdots \overset{\tau_k}{\hookrightarrow} \langle s, \tau_0 + \cdots + \tau_k\rangle \overset{\tau_{k+1}}{\hookrightarrow} \langle s', 0\rangle$ of $s \xrightarrow{\tau} s'$, with $\tau_0 + \tau_1 \cdots + \tau_k + \tau_{k+1} = \tau$. Furthermore, a path π requires us to consider any possible continuous position in the path.

Example 2.7. Consider the tick transition $s_0 \xrightarrow{4} s_1$ of the timed Kripke structure of Fig. 1 with time domain $\mathbb{Q}_{\geq 0}$. The only possible pointwise interpretation of this transition corresponds to the step $\langle s_0, 0\rangle \overset{4}{\hookrightarrow} \langle s_1, 0\rangle$, while there are infinitely many possible ways of interpreting it in the continuous one, e.g., $\langle s_0, 0\rangle \overset{1}{\hookrightarrow} \langle s_0, 1\rangle \overset{3}{\hookrightarrow} \langle s_1, 0\rangle$, or $\langle s_0, 0\rangle \overset{1/2}{\hookrightarrow} \langle s_0, 1/2\rangle \overset{1}{\hookrightarrow} \langle s_0, 3/2\rangle \overset{5/2}{\hookrightarrow} \langle s_1, 4\rangle$, and so on.

A TCTL formula $\varphi = A\ G_{[1,2]}\ \varphi_1$ is trivially satisfied in state s_0 in the pointwise semantics, since there is no configuration at a time in the interval $[1, 2]$. In order to evaluate the same formula in the continuous semantics, it must be checked that the infinitely many continuous positions in the interval $[1, 2]$ satisfy φ_1.

Example 2.8. Consider the timed Kripke structure in Fig. 1 with time domain $\mathbb{Q}_{\geq 0}$. In the pointwise semantics, we have that $\mathcal{TK}, s_0 \models_p E\ p\ U_{<6}\ q$ due to the (pointwise) path

$$\langle s_0, 0\rangle \overset{4}{\hookrightarrow} \langle s_1, 0\rangle \overset{0}{\hookrightarrow} \langle s_3, 0\rangle \overset{4}{\hookrightarrow} \cdots.$$

However, $\mathcal{TK}, s_0 \not\models_p A\ p\ U_{<6}\ q$ due to, e.g., the path

$$\langle s_0, 0\rangle \overset{4}{\hookrightarrow} \langle s_1, 0\rangle \overset{0}{\hookrightarrow} \langle s_2, 0\rangle \overset{2}{\hookrightarrow} \langle s_2, 0\rangle \overset{2}{\hookrightarrow} \langle s_2, 0\rangle \overset{2}{\hookrightarrow} \cdots.$$

For the property $E\ F_{=2}\ true$ we have that $\mathcal{TK}, s_0 \models_c E\ F_{=2}\ true$ and $\mathcal{TK}, s_0 \not\models_p E\ F_{=2}\ true$, since there exists a continuous path, e.g., $\langle s_0, 0\rangle \overset{2}{\hookrightarrow} \langle s_0, 2\rangle \overset{2}{\hookrightarrow} \langle s_1, 0\rangle \overset{0}{\hookrightarrow} \cdots$, that reaches a configuration in exactly 2 time units from s_0, while, in the pointwise interpretation of $\mathcal{TK}$ this is not the case. The property $A\ F_{=2}\ true$ also holds in the continuous semantics, since for each continuous path starting in s_0, we encounter a continuous position at time 2.

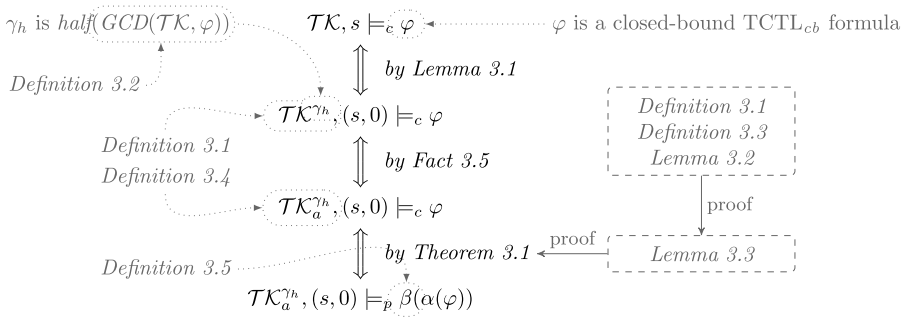


Fig. 2. Equivalences used to reduce model checking TCTL_{cb} formulas in the continuous semantics to a model checking problem in the pointwise semantics (as proven in Theorem 3.2).

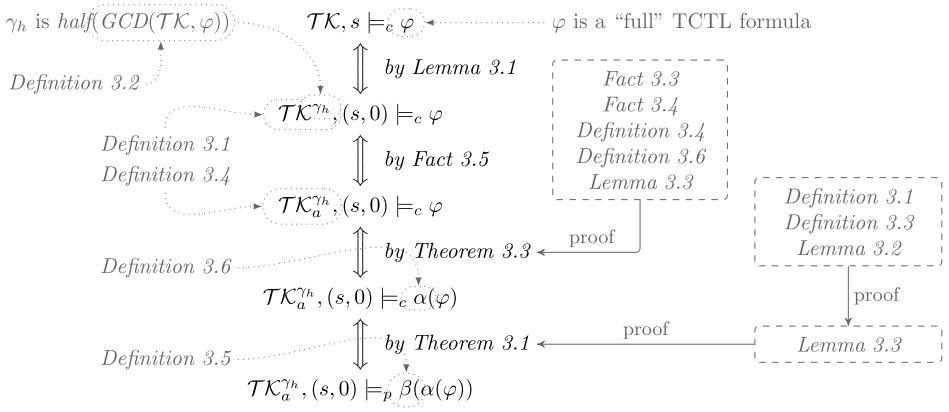


Fig. 3. Equivalences reducing model checking (all) TCTL formulas in the continuous semantics to model checking in the pointwise semantics (as proven in Theorem 3.4).

3. Reducing model checking in the continuous semantics to model checking in the pointwise semantics

In this section we explain how TCTL model checking in the continuous semantics can be reduced to TCTL model checking in the pointwise semantics, which is amenable to *explicit-state* model checking. In particular, we describe a transformation of a timed Kripke structure $\mathcal{TK}$, so that model checking the resulting timed Kripke structure in the pointwise semantics is equivalent to model checking $\mathcal{TK}$ in the continuous semantics. The reduction from model checking in the continuous semantics to model checking in the pointwise semantics consists of the following main steps:

1. A transformation (called, the *gcd*-transformation) mapping a timed Kripke structure $\mathcal{TK}$ to another timed Kripke structure $\mathcal{TK}_a^{half(GCD(\mathcal{TK},\varphi))}$, so that
$$\mathcal{TK}, s \models_c \varphi \iff \mathcal{TK}_a^{half(GCD(\mathcal{TK},\varphi))}, (s,0) \models_p \beta(\varphi),$$
for a TCTL_{cb} formula φ (i.e., all temporal operators in φ are annotated with closed intervals $[a,b]$ or $[a,\infty)$), where $(s,0)$ is the state in $\mathcal{TK}_a^{half(GCD(\mathcal{TK},\varphi))}$ corresponding to s, and $\beta(\varphi)$ is a transformation of φ.
2. We then extend this soundness and completeness result to the entire TCTL logic, by transforming a given TCTL formula (with possibly open intervals) φ to a TCTL_{cb} formula $\alpha(\varphi)$, so that
$$\mathcal{TK}_a^{half(GCD(\mathcal{TK},\varphi))}, (s,0) \models_c \varphi \iff \mathcal{TK}_a^{half(GCD(\mathcal{TK},\varphi))}, (s,0) \models_c \alpha(\varphi).$$

By combining the above results together with the fact that the *gcd*-transformation does not change the continuous semantics of timed Kripke structures, i.e. $\mathcal{TK}, s \models_c \varphi \iff \mathcal{TK}_a^{half(GCD(\mathcal{TK},\varphi))}, (s,0) \models_c \varphi$, we get the desired equivalence which reduces the model checking problem in the continuous semantics to a model checking problem in the pointwise semantics:

$$\mathcal{TK}, s \models_c \varphi \iff \mathcal{TK}_a^{half(GCD(\mathcal{TK},\varphi))}, (s,0) \models_p \beta(\alpha(\varphi)),$$

for any TCTL formula φ.

Fig. 2 gives an overview of the lemmas, theorems, and definitions used in Section 3.2 to reduce model checking in the continuous semantics to model checking in the pointwise semantics for formulas with closed interval bounds. Dotted arrows

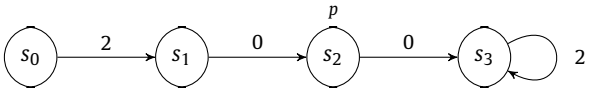
indicate where the element in the pointed circle is defined, while arrows labeled by "proof" indicate proof dependencies (e.g., the arrow from Lemma 3.3 to Theorem 3.1 indicates that the result proven in Lemma 3.3 is necessary to prove the result stated in Theorem 3.1). Intuitively, $\mathcal{TK}_a^{\gamma_h}$ represents the *gcd*-transformation of the timed Kripke structure $\mathcal{TK}$.

Fig. 3 extends the diagram in Fig. 2 with the additional steps for *full* TCTL formulas used in Section 3.3. Besides φ now being a general TCTL formula, the main difference between this diagram and the one in Fig. 3 is the transformation of φ into the TCTL_{cb} formula $\alpha(\varphi)$, which, as shown in Theorem 3.3, does not change the continuous semantics of the *gcd*-transformation $\mathcal{TK}_a^{\gamma_h}$.

Section 3.4 discusses discrete time and how our reduction can be used to speed up the model checking of some discrete-time systems.

One part of our solution is to make sure that time progress "stops" at any time point when a time bound in the formula can be reached. This can be achieved if we "split" any tick transition into a sequence of tick transitions of a smaller duration τ that divides the duration of each tick transition and each finite non-zero time bound in the formula. The following example shows that it is not sufficient to always advance time by the *greatest* common divisor (gcd) γ of these values to obtain a sound and complete model checking in the continuous semantics.

Example 3.1. Consider the following timed Kripke structure $\mathcal{TK}$:



$\mathcal{TK}$ has only one behavior from s_0 in the pointwise interpretation, which we show here in terms of validity of the atomic proposition p in the corresponding states:

$$\pi = \neg p \xrightarrow{2} \neg p \xrightarrow{0} p \xrightarrow{0} \neg p \xrightarrow{2} \neg p \xrightarrow{2} \cdots (\neg p \text{ forever})$$

That is, the only p-state is reachable in exactly time 2. All tick transitions have duration 2. Consider the formula $\varphi = E\ \varphi_1\ U_{=2}\ \mathit{true}$, where φ_1 is the formula $E\ F_{=2}\ p$. The formula φ says that φ_1 must hold in all positions until we reach 2 time units.[4] The greatest common divisor of all tick transition durations and all time values in φ is 2, so transforming $\mathcal{TK}$ by "splitting" each tick transition into a "sequence of transitions", each of duration 2, leaves it as it was. In the pointwise semantics this transformation (i.e., the above behavior) satisfies φ w.r.t. the initial state s_0. However, $\mathcal{TK}, s_0 \models_c \varphi$ does *not* hold, since φ_1 does not hold in the continuous positions along the first tick step; for example, it does not hold at time 1.

Our approach is therefore to capture these "intermediate" states by further splitting the tick transitions of duration equal to the gcd into two smaller ones. In essence, we advance time not by the greatest common divisor γ, but by a time γ_h such that $2 \cdot \gamma_h = \gamma$, that is, by "half" the gcd, in each tick step.

3.1. τ-transformation of timed Kripke structures

We first introduce τ-transformations as a generalization of the *gcd*-transformation. We then prove that such transformations do not change the continuous semantics of timed Kripke structures.

In the remainder of this section, unless said otherwise, we assume that AP is a set of atomic propositions, $\mathcal{TK} = (S, \mathcal{T}, \longrightarrow, L)$ is a timed Kripke structure over AP, and φ is a TCTL formula over AP.

Intuitively, a τ-transformation "splits" a tick transition $s \xrightarrow{\tau'} s'$ into k transitions of duration τ followed by a transition of duration $\tau' \dot{-} k\tau$, and requires k new states (s, δ) at times $\delta = \tau, 2\tau, \ldots, k\tau$.

Definition 3.1 *(τ-transformation).* Let $\tau \in \mathcal{T}$ be a non-zero time value. The *τ-transformation* of $\mathcal{TK}$ is the timed Kripke structure $\mathcal{TK}^\tau = (S^\tau, \mathcal{T}, \longrightarrow^\tau, L^\tau)$ with

- $S^\tau = \{(s, \delta) \in S \times \mathcal{T} \mid \delta = 0 \vee (\exists n \in \mathbb{N}, \tau' \in \mathcal{T}_{\mathcal{TK}}(s).\ \delta = n\tau < \tau')\}$,
- $((s_1, \delta_1), \tau_1, (s_2, \delta_2)) \in \longrightarrow^\tau$ if and only if
  - $s_1 \xrightarrow{0} s_2$ and $\tau_1 = \delta_1 = \delta_2 = 0$; or
  - $s_1 \xrightarrow{\tau_1'} s_2'$ and $s_2 = s_1$, $\tau_1 = \tau$, $\delta_2 = \delta_1 + \tau < \tau_1'$; or
  - $s_1 \xrightarrow{\tau_1'} s_2$ and $\tau_1 \leq \tau$, $\delta_1 + \tau_1 = \tau_1'$, $\delta_2 = 0$.
- $L^\tau(s, \delta) = L(s)$.

[4] The interval does not have to be the single point interval $[2, 2]$, e.g., $[2, 4]$ would also work to illustrate the issue.

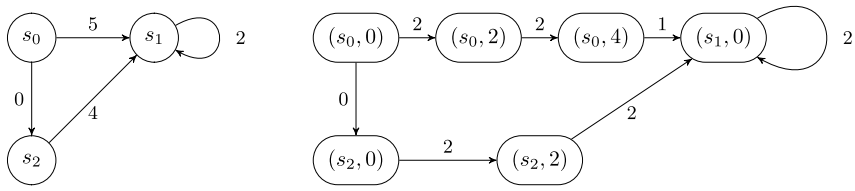


Fig. 4. A timed Kripke structure (on the left) and its 2-transformation (on the right).

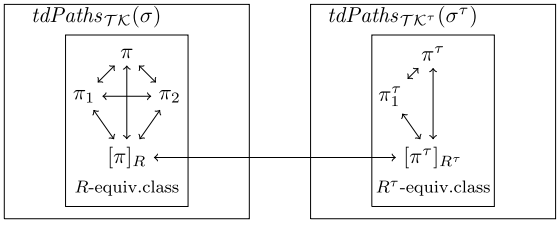


Fig. 5. Illustration of the proof for Lemma 3.1 (arrows state TCTL-equivalence).

Intuitively, a state (s,δ) in $\mathcal{TK}^\tau$ denotes that the system is in state s and that time has advanced by δ since the last $\mathcal{TK}$-transition. Instantaneous $\mathcal{TK}$-transitions are represented as transitions $(s_1,0)\xrightarrow{0}{}^{\tau}(s_2,0)$ in $\mathcal{TK}^\tau$, and a tick $\mathcal{TK}$-transition $s_1 \xrightarrow{k\tau+\tau_1} s_2'$ is represented by the sequence $(s_1,0) \xrightarrow{\tau} (s_1,\tau) \xrightarrow{\tau} \cdots \xrightarrow{\tau} (s_1,k\tau) \xrightarrow{\tau_1} (s_2,0)$.

Example 3.2. Fig. 4 shows a timed Kripke structure on the left and its 2-transformation on the right.

The following lemma states that the τ-transformation does not change the continuous semantics of timed Kripke structures.

Lemma 3.1. *Let $\langle s, n\tau+\delta\rangle \in \mathcal{C}^c_{\mathcal{TK}}$ for some $s \in S$, $n \in \mathbb{N}$, $\tau \in \mathcal{T}$ with $\tau > 0$, and $\delta \in \mathcal{T}$ with $\delta < \tau$. Then*

$$\mathcal{TK}, \langle s, n\tau+\delta\rangle \models_c \varphi \quad \textit{iff} \quad \mathcal{TK}^\tau, \langle (s,n\tau),\delta\rangle \models_c \varphi.$$

Proof. We denote the configurations $\langle s, n\tau+\delta\rangle$ and $\langle (s,n\tau),\delta\rangle$ by, respectively, σ and σ^τ. Fig. 5 illustrates the steps of this proof.

We define an equivalence relation $R \subseteq tdPaths^c_{\mathcal{TK}}(\sigma) \times tdPaths^c_{\mathcal{TK}}(\sigma)$ on the time-divergent paths of $\mathcal{TK}$ starting in the configuration σ in the continuous semantics such that two paths are equivalent iff they have the same set of continuous positions. It is easy to see that equivalent paths satisfy the same TCTL path formulas. For each equivalence class we define as representative element a path of the form

$$\pi = \sigma \overset{\tau_0}{\hookrightarrow} \langle s_1, n_1\tau\rangle \overset{\tau_1}{\hookrightarrow} \langle s_2, n_2\tau\rangle \overset{\tau_2}{\hookrightarrow} \cdots$$

with $\tau_i \leq \tau$ and $n_i \in \mathbb{N}$ for each $i \in \mathbb{N}$. Such a representative element exists for each equivalence class and is unique. $[\pi]_R$ denotes the representative element of the equivalence class of π and $\Pi_R = \{[\pi]_R \mid \pi \in tdPaths^c_{\mathcal{TK}}(\sigma)\}$ denotes the set of all representative elements.

Let an analogous equivalence relation $R^\tau \subseteq tdPaths^c_{\mathcal{TK}^\tau}(\sigma^\tau) \times tdPaths^c_{\mathcal{TK}^\tau}(\sigma^\tau)$ on the time-divergent paths of $\mathcal{TK}^\tau$ starting in the configuration σ^τ in the continuous semantics relate those paths that have the same set of continuous positions. Again, equivalent paths satisfy the same TCTL path formulas. For each equivalence class we define as representative element a path

$$\pi^\tau = \sigma^\tau \overset{\tau_0}{\hookrightarrow} \langle (s_1, n_1\tau), 0\rangle \overset{\tau_1}{\hookrightarrow} \langle (s_2, n_2\tau), 0\rangle \overset{\tau_2}{\hookrightarrow} \cdots$$

such that $\tau_i \leq \tau$ and $n_i \in \mathbb{N}$ for each $i \in \mathbb{N}$. Such a representative element exists for each equivalence class and is unique. We use $[\pi^\tau]_{R^\tau}$ to denote the representative element of the equivalence class of π^τ and $\Pi_{R^\tau} = \{[\pi^\tau]_{R^\tau} \mid \pi^\tau \in tdPaths^c_{\mathcal{TK}^\tau}(\sigma^\tau)\}$ to denote the set of all representative elements.

We define a mapping between representative elements of the two relations in the form of a bijective function $f : \Pi_R \to \Pi_{R^\tau}$, mapping to each $\pi = \sigma \overset{\tau_0}{\hookrightarrow} \langle s_1, n_1\tau\rangle \overset{\tau_1}{\hookrightarrow} \langle s_2, n_2\tau\rangle \overset{\tau_2}{\hookrightarrow} \cdots \in \Pi_R$ the unique path $f(\pi) = \sigma^\tau \overset{\tau_0}{\hookrightarrow} \langle (s_1, n_1\tau), 0\rangle \overset{\tau_1}{\hookrightarrow} \langle (s_2, n_2\tau), 0\rangle \overset{\tau_2}{\hookrightarrow} \cdots \in \Pi_{R^\tau}$. Note that f is bijective and that f maps TCTL-equivalent paths, i.e., for all TCTL path formulas ψ and all $\pi \in \Pi_R$ we have that $\mathcal{TK}, \pi \models_c \psi$ iff $\mathcal{TK}^\tau, f(\pi) \models_c \psi$.

We prove that $\mathcal{TK}, \sigma \models_c \varphi$ iff $\mathcal{TK}^\tau, \sigma^\tau \models_c \varphi$, by induction on the structure of φ. The cases for atomic propositions and the boolean connectives are straightforward.

- For $\varphi = A\ \psi$ we have that

$$\begin{aligned}
\mathcal{TK}, \sigma \models_c A\ \psi &\overset{\text{TCTL semantics}}{\iff} \forall \pi \in tdPaths^c_{\mathcal{TK}}(\sigma).\ \mathcal{TK}, \pi \models_c \psi \\
&\overset{R \text{ is TCTL-preserving}}{\iff} \forall \pi \in \Pi_R.\ \mathcal{TK}, \pi \models_c \psi \\
&\overset{f \text{ is bijective and TCTL-preserving}}{\iff} \forall \pi^\tau \in \Pi_{R^\tau}.\ \mathcal{TK}^\tau, \pi^\tau \models_c \psi \\
&\overset{R^\tau \text{ is TCTL-preserving}}{\iff} \forall \pi^\tau \in tdPaths^c_{\mathcal{TK}^\tau}(\sigma^\tau).\ \mathcal{TK}^\tau, \pi^\tau \models_c \psi \\
&\overset{\text{TCTL semantics}}{\iff} \mathcal{TK}^\tau, \sigma^\tau \models_c A\ \psi.
\end{aligned}$$

- For $\varphi = E\ \psi$ we have that

$$\begin{aligned}
\mathcal{TK}, \sigma \models_c E\ \psi &\overset{\text{TCTL semantics}}{\iff} \exists \pi \in tdPaths^c_{\mathcal{TK}}(\sigma).\ \mathcal{TK}, \pi \models_c \psi \\
&\overset{R \text{ is TCTL-preserving}}{\iff} \exists \pi \in \Pi_R.\ \mathcal{TK}, \pi \models_c \psi \\
&\overset{f \text{ is bijective and TCTL-preserving}}{\iff} \exists \pi^\tau \in \Pi_{R^\tau}.\ \mathcal{TK}^\tau, \pi^\tau \models_c \psi \\
&\overset{R^\tau \text{ is TCTL-preserving}}{\iff} \exists \pi^\tau \in tdPaths^c_{\mathcal{TK}^\tau}(\sigma^\tau).\ \mathcal{TK}^\tau, \pi^\tau \models_c \psi \\
&\overset{\text{TCTL semantics}}{\iff} \mathcal{TK}^\tau, \sigma^\tau \models_c E\ \psi. \qquad \square
\end{aligned}$$

We define the time value $gcd(\mathcal{T}_{\mathcal{TK}} \cup \mathcal{T}_\varphi)$ to be the greatest common divisor of the set $\mathcal{T}_{\mathcal{TK}}$ of all possible tick transition durations, and the set $\mathcal{T}_\varphi$ of all possible finite non-zero time bounds in the formula φ.

Definition 3.2. Given a timed Kripke structure $\mathcal{TK} = (S, \mathcal{T}, \longrightarrow, L)$ over a set of atomic propositions AP, where the time domain $\mathcal{T}$ satisfies $TIME^{gcd}$, and a TCTL formula φ over AP, we define

$$\begin{aligned}
\mathcal{T}_{\mathcal{TK}} &= \{\tau \in \mathcal{T} \mid \exists s, s' \in S.\ s \xrightarrow{\tau} s' \wedge 0 < \tau\} \\
\mathcal{T}_\varphi &= \{\tau \in \{inf(I), sup(I)\} \backslash \{0, \infty\} \mid \textit{there exists a subformula } \varphi_1\ U_I\ \varphi_2 \textit{ of } \varphi\} \\
GCD(\mathcal{TK}, \varphi) &= gcd(\mathcal{T}_{\mathcal{TK}} \cup \mathcal{T}_\varphi),
\end{aligned}$$

where, for $\mathcal{T}' \subseteq \mathcal{T}$ a finite non-empty set of time values, $gcd(\mathcal{T}')$ is the greatest common divisor of these time values, and is recursively defined as $gcd(\{\tau\}) = \tau$ and $gcd(\{\tau_1, \tau_2\} \uplus \mathcal{T}') = gcd(\{gcd(\tau_1, \tau_2)\} \cup \mathcal{T}')$.

Fact 3.1. *If $\mathcal{TK}$ is finite and has at least one tick transition then the set of time values $\mathcal{T}_{\mathcal{TK}} \cup \mathcal{T}_\varphi$ is non-empty and finite.*

Example 3.3. In the timed Kripke structure $\mathcal{TK}$ and formula φ in Example 3.1, $\mathcal{T}_{\mathcal{TK}} = \{2\}$, $\mathcal{T}_\varphi = \{2\}$, and therefore $GCD(\mathcal{TK}, \varphi) = 2$.

In the rest of this section, unless otherwise specified, we assume that the time domain of $\mathcal{TK}$ satisfies $TIME^{gcd}$, $\mathcal{TK}$ is finite and Zeno-free and has at least one tick transition, γ denotes the (defined) time value $GCD(\mathcal{TK}, \varphi)$, and γ_h denotes $half(\gamma)$.

3.2. Reducing model checking in the continuous semantics to model checking in the pointwise semantics for $TCTL_{cb}$-formulas

In this section, we explain how model checking in the continuous semantics can be reduced to pointwise model checking for formulas in the restricted logic $TCTL_{cb}$.

Let φ be a $TCTL_{cb}$ formula. A non-Zeno path in $\mathcal{TK}^{\gamma_h}$ starting in state $(s_0, 0)$ with only maximal tick steps consists of an alternating sequence of zero or more instantaneous steps, followed by a sequence of one or more tick steps having total duration multiple of γ:

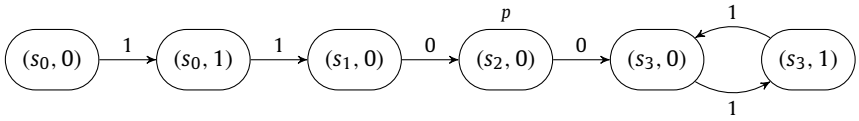
$$\langle (s_0,0),0\rangle \underbrace{\xrightarrow{0} \cdots \xrightarrow{0}}_{k_0 \text{ times}} \langle (s_{k_0},0),0\rangle \quad \underbrace{\xrightarrow{\gamma_h} \cdots \xrightarrow{\gamma_h}}_{\text{total duration } \gamma n_0} \quad \langle (s'_{k_0},0),0\rangle \underbrace{\xrightarrow{0} \cdots \xrightarrow{0}}_{k_1 \text{ times}} \cdots$$

where, for each $i \in \mathbb{N}$, $k_i, n_i \in \mathbb{N}$, and $n_i > 0$, there exists a tick transition of duration γn_i from s_{k_i} to s'_{k_i} in $\mathcal{TK}$. Intuitively, this maximal path in $\mathcal{TK}^{\gamma_h}$ corresponds to the following maximal path in $\mathcal{TK}$:

$$\langle s_0,0\rangle \underbrace{\xrightarrow{0} \cdots \xrightarrow{0}}_{k_0 \text{ times}} \langle s_{k_0},0\rangle \xrightarrow{\gamma n_0} \langle s'_{k_0},0\rangle \underbrace{\xrightarrow{0} \cdots \xrightarrow{0}}_{k_1 \text{ times}} \cdots$$

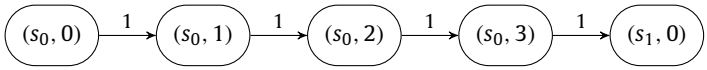
The main fact that we use to prove that our reduction is sound and complete is the following. All configurations that can be reached in a time $\tau \in (0, \gamma)$ from a valid configuration $\langle (s, 2n \cdot \gamma_h), 0\rangle$ (for some $n \in \mathbb{N}$) in $\mathcal{TK}^{\gamma_h}$ by a sequence of tick steps in the continuous interpretation of $\mathcal{TK}^{\gamma_h}$, satisfy the same subformulas of φ. Therefore, each of these sets of configurations will be represented by the respective canonical representative $\langle (s, (2n+1)\gamma_h), 0\rangle$.

Example 3.4. Consider the timed Kripke structure $\mathcal{TK}$ and formula φ of Example 3.1 with a dense time domain. Here $GCD(\mathcal{TK}, \varphi) = 2$. If we split each tick transition into transitions of duration $\gamma_h = 1$, we obtain $\mathcal{TK}^1$:

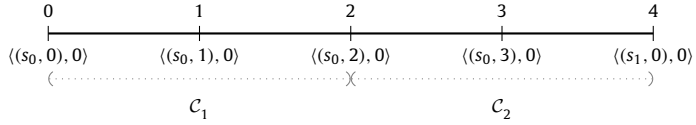


We show below that all (continuous) configurations "strictly between" $\langle (s_0, 0), 0\rangle$ and $\langle (s_1, 0), 0\rangle$ satisfy the same subformulas of φ, and represent these configurations with the canonical representative $\langle (s_0, 1), 0\rangle$.

Example 3.5. Consider the timed Kripke structure $\mathcal{TK}$ of Fig. 1 with a dense time domain, and a formula φ such that $GCD(\mathcal{TK}, \varphi) = 2$. The tick transition $s_0 \xrightarrow{4} s_1$ is split in four tick transitions of duration 1 in $\mathcal{TK}^1$:



In the continuous interpretation, there are two sets of configurations, denoted by $\mathcal{C}_1$ and $\mathcal{C}_2$ in the time segment below, that satisfy the same subformulas of φ.



We represent all configurations in $\mathcal{C}_1$ by $\langle (s_0, 1), 0\rangle$, and all configurations in $\mathcal{C}_2$ by $\langle (s_0, 3), 0\rangle$.

We use $can(\sigma)$ to denote the *canonical representative* of a configuration σ of $\mathcal{TK}^{\gamma_h}$. The definition is generalized for any time τ that is a divisor of γ_h.

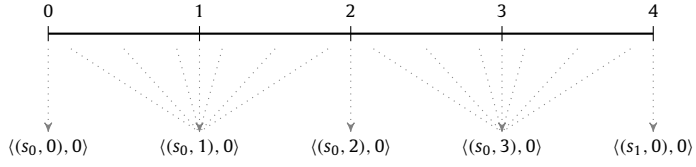
Definition 3.3. Given $0 < \tau \in \mathcal{T}$ such that $GCD(\mathcal{TK}, \varphi)$ is a multiple of 2τ and $\mathcal{TK}^\tau = (S, \mathcal{T}, \longrightarrow, L)$ the τ-transformation of $\mathcal{TK}$. We define the function $can_{\mathcal{TK}^\tau} : \mathcal{C}^c_{\mathcal{TK}^\tau} \to \mathcal{C}^c_{\mathcal{TK}^\tau}$ on valid configurations $\langle (s, n\tau), \delta\rangle$ of $\mathcal{TK}^\tau$ as

$$can_{\mathcal{TK}^\tau}(\langle (s, n\tau), \delta\rangle) = \begin{cases} \langle (s, n\tau), 0\rangle & \text{if } \delta = 0 \text{ or } n \text{ is odd} \\ \langle (s, (n+1)\tau), 0\rangle & \text{otherwise} \end{cases}$$

and call $can_{\mathcal{TK}^\tau}(\langle (s, n\tau), \delta\rangle)$ the *canonical representative of* $\langle (s, n\tau), \delta\rangle$. We omit the subscript $\mathcal{TK}^\tau$ when it is clear from the context.

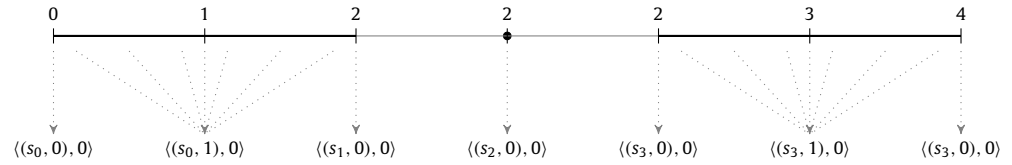
Fact 3.2. *The canonical representative of configurations of the kind* $\langle (s, 0), 0\rangle$ *in* $\mathcal{TK}^{\gamma_h}$*, which corresponds to the state* s *in* $\mathcal{TK}$*, is the identity. The canonical representative of configurations of the kind* $\langle (s, \delta), \delta'\rangle$*, with* $0 < \delta' < \gamma_h$*, is a configuration* $\langle (s, \delta''), 0\rangle$ *such that* δ'' *is an* odd multiple *of* γ_h*, i.e.,* $\delta'' = (2n+1)\gamma_h$ *for some* $n \in \mathbb{N}$*.*

Example 3.6. The following figure shows the mapping of configurations to their canonical representatives on the time segment of Example 3.5.



Dotted arrows map a configuration σ on the top to its canonical representative $can(\sigma)$ on the bottom. Notice that for any $0 < \delta < 1 = \gamma_h$, the configurations $\langle (s_0, 0), \delta \rangle$ and $\langle (s_0, 1), \delta \rangle$ are represented by the configuration $\langle (s_0, 1), 0 \rangle$, and the configurations $\langle (s_0, 2), \delta \rangle$ and $\langle (s_0, 3), \delta \rangle$ are represented by $\langle (s_0, 3), 0 \rangle$, i.e., in both cases the set of configurations are represented by a configuration $\langle (s_0, \delta), 0 \rangle$, where δ is an odd multiple of γ_h.

Example 3.7. The following figure shows the mapping of configurations to their canonical representatives on the time segment corresponding to the path of $\mathcal{TK}$ in Example 3.1.



Again, dotted arrows map a configuration on the top to its canonical representative on the bottom. The continuous positions along tick steps are denoted by a thick black line, and the continuous position after a discrete step (at time 2) is denoted by a black circle.

Fact 3.3. *By definition, γ divides all finite non-zero time bounds in any time interval appearing in φ. Therefore, any finite bound in such time intervals is a multiple of $2\gamma_h$ (e.g., if $\gamma_h = 1/2$ and $I = (2, \infty)$, I is $(2\gamma_h \cdot 2, \infty)$). Furthermore, any time interval I in a TCTL$_{cb}$ formula has the form $[2m_l \cdot \gamma_h, 2m_u \cdot \gamma_h]$ for some $m_l, m_u \in \mathbb{N}$ with $m_l \leq m_u$, or $I = [2m_l \cdot \gamma_h, \infty)$ for some $m_l \in \mathbb{N}$.*

To prove the correctness of our reduction we use the following lemma to argue that if the time duration between two configurations along a path is in a given time interval I (appearing in φ) then the time duration between the corresponding canonical representatives is also in I.

Lemma 3.2. *Let $\tau \in \mathcal{T}$, $\tau > 0$ and I a time interval such that either $I = [2m_l \cdot \tau, 2m_u \cdot \tau]$ for some $m_l, m_u \in \mathbb{N}$ with $m_l \leq m_u$, or $I = [2m_l \cdot \tau, \infty)$ for some $m_l \in \mathbb{N}$. Let furthermore $\delta_i \in \mathcal{T}$ with $\delta_i = 2n_i\tau + \tau_i$ for some $n_i \in \mathbb{N}$ and $\tau_i \in \mathcal{T}$, $0 \leq \tau_i < 2\tau$ for $i = 0, 1$. We define*

$$\delta_i^* = \begin{cases} \delta_i & \text{if } \tau_i = 0 \\ 2n_i\tau + \tau & \text{if } \tau_i > 0 \end{cases}$$

for $i = 0, 1$. Then

$$\delta_1 - \delta_0 \in I \quad \Longrightarrow \quad \delta_1^* - \delta_0^* \in I.$$

Proof. Assume $\delta_1 - \delta_0 \in I$. We distinguish the following cases:

- $\tau_0 = \tau_1 = 0$. In this case $\delta_1 - \delta_0 = \delta_1^* - \delta_0^* = 2(n_1 - n_0)\tau$ and the implication holds.
- $\tau_0 = 0$, $0 < \tau_1 < 2\tau$. In this case

  $$\delta_1 - \delta_0 = (2n_1\tau + \tau_1) - 2n_0\tau = 2(n_1 - n_0)\tau + \tau_1$$
  $$\delta_1^* - \delta_0^* = (2n_1\tau + \tau) - 2n_0\tau = 2(n_1 - n_0)\tau + \tau.$$

  Since $\delta_1 - \delta_0 \in I$ and all finite bounds of I are by assumption closed and multiples of 2τ, it follows that $[2(n_1 - n_0)\tau, 2(n_1 - n_0 + 1)\tau] \subseteq I$ and therefore $\delta_1^* - \delta_0^* \in I$.
- $0 < \tau_0 < 2\tau$, $\tau_1 = 0$. In this case

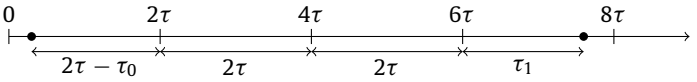
  $$\delta_1 - \delta_0 = 2n_1\tau - (2n_0\tau + \tau_0) = 2(n_1 - n_0)\tau - \tau_0$$
  $$\delta_1^* - \delta_0^* = 2n_1\tau - (2n_0\tau + \tau) = 2(n_1 - n_0)\tau - \tau.$$

  Since $\delta_1 - \delta_0 \in I$ and all finite bounds of I are by assumption closed and multiples of 2τ, it follows that $[2(n_1 - n_0 - 1)\tau, 2(n_1 - n_0)\tau] \subseteq I$ and therefore $\delta_1^* - \delta_0^* \in I$.
- $0 < \tau_0, \tau_1 < 2\tau$

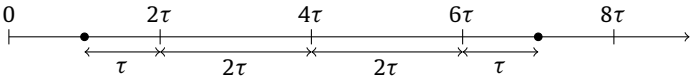
  $$\delta_1 - \delta_0 = (2n_1\tau + \tau_1) - (2n_0\tau + \tau_0) = 2(n_1 - n_0)\tau + (\tau_1 - \tau_0)$$
  $$\delta_1^* - \delta_0^* = (2n_1\tau + \tau) - (2n_0\tau + \tau) = 2(n_1 - n_0)\tau.$$

  Note that $-2\tau < \tau_1 - \tau_0 < 2\tau$, i.e., $2(n_1 - n_0 - 1)\tau < \delta_1 - \delta_0 < 2(n_1 - n_0 + 1)\tau$. Since $\delta_1 - \delta_0 \in I$ and all finite bounds of I are by assumption closed and multiples of 2τ, it follows that $2(n_1 - n_0)\tau \in I$ and therefore $\delta_1^* - \delta_0^* \in I$. □

Example 3.8. Assume a path with two configurations (denoted by circles):



If the time duration between those configurations is in an interval I whose closed finite bounds are multiples of 2τ then the time duration between the following (canonical) configurations is also within I:



Lemma 3.3, whose proof is given in Appendix B, shows that a configuration σ and its canonical representative $can(\sigma)$ satisfy the same subformulas of φ in the continuous semantics. The lemma is more general and shows that the mapping to the canonical representatives preserves the equivalence for any TCTL$_{cb}$ formula with time bounds that are multiples of the computed greatest common divisor γ.

Lemma 3.3. *Let φ be a TCTL$_{cb}$ formula over AP, $0 < \tau \in \mathcal{T}$ such that $GCD(\mathcal{TK}, \varphi)$ is a multiple of 2τ and $\mathcal{TK}^\tau = (S, \mathcal{T}, \longrightarrow, L)$ the τ-transformation of $\mathcal{TK}$. Then for each valid configuration σ of $\mathcal{TK}^\tau$ it holds that*

$$\mathcal{TK}^\tau, \sigma \models_c \varphi \quad \Longleftrightarrow \quad \mathcal{TK}^\tau, can(\sigma) \models_c \varphi.$$

Lemma 3.3 states that, given a tick step of duration γ_h in $\mathcal{TK}^{\gamma_h}$, all (continuous) configurations during this tick step (i.e., all configurations $\langle (s, \delta), \delta' \rangle$ with $0 < \delta' < \gamma_h$) satisfy the same subformulas of φ. Therefore, we can "abstract" these configurations with the respective canonical representative (i.e., $can(\langle (s, \delta), \delta' \rangle)$), which, as noted in Fact 3.2, has the form $\langle (s, (2n+1)\gamma_h), 0 \rangle$, for some $n \in \mathbb{N}$. These canonical configurations play an important role in the proof of our soundness and completeness theorem. Since each of them is the canonical representative for a set of configurations (in contrast to other canonical configurations, i.e. configurations of the form $\langle (s, (2n)\gamma_h), 0 \rangle$, that only represent themselves), in the following, we will refer to them as the *abstract configurations* (in contrast to the other canonical configurations that will be referred as the *concrete configurations*).

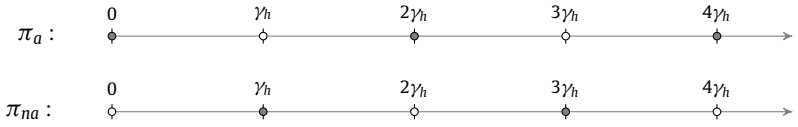
Example 3.9. The abstract configurations in Example 3.6 are $\langle (s_0, 1), 0 \rangle$ and $\langle (s_0, 3), 0 \rangle$, while the concrete ones are $\langle (s_0, 0), 0 \rangle$, $\langle (s_0, 2), 0 \rangle$ and $\langle (s_1, 0), 0 \rangle$. The abstract configurations in Example 3.7 are $\langle (s_0, 1), 0 \rangle$ and $\langle (s_3, 1), 0 \rangle$, while the concrete ones are $\langle (s_0, 0), 0 \rangle$, $\langle (s_1, 0), 0 \rangle$, $\langle (s_2, 0), 0 \rangle$ and $\langle (s_3, 0), 0 \rangle$.

The regularity of the position of abstract configurations on paths of $\mathcal{TK}^{\gamma_h}$ is a key aspect for proving our soundness and completeness result. Intuitively, if a path starts in an abstract configuration, then all the positions at an even multiple of γ_h are also abstract, and all positions at an odd multiple of γ_h are concrete. Similarly, if a path starts in a concrete configuration, then all positions at an even multiple of γ_h are also concrete, and all positions at an odd multiple of γ_h are abstract. Again, in general, the fact holds for any $\mathcal{TK}^\tau$ such that γ_h is a multiple of τ.

Fact 3.4. *Given an abstract configuration (respectively, a concrete configuration) σ of $\mathcal{TK}^\tau$, such that γ_h is a multiple of τ, and a path $\pi \in Paths_{\mathcal{TK}^\tau}(\sigma)$, then both the following facts are true:*

- *For all positions $\sigma'@c$ in π such that c is an* even *multiple of τ, it holds that σ' is an abstract configuration (respectively, a concrete configuration).*
- *For all positions $\sigma'@c$ in π such that c is an* odd *multiple of τ, it holds that σ' is a concrete configuration (respectively, an abstract configuration).*

Example 3.10. The path π_a below is a path (of some γ_h-transformation $\mathcal{TK}^{\gamma_h}$) starting in an abstract configuration, while π_{na} is a path starting in a concrete one. The abstract configurations on each path are denoted by gray circles, while white circles denote the concrete ones.

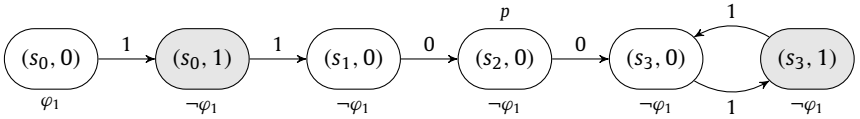


Pointwise model checking of $\mathcal{TK}^{\gamma_h}$ is still not a sound and complete model checking procedure for model checking TCTL$_{cb}$ formulas in $\mathcal{TK}$ in the continuous semantics:

$$\mathcal{TK}^{\gamma_h}, (s,0) \models_p \varphi \quad \not\Longleftrightarrow \quad \mathcal{TK}, s \models_c \varphi.$$

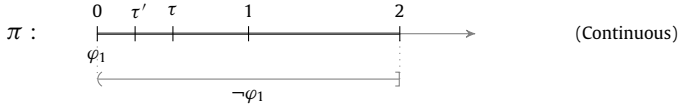
Example 3.11 illustrates why this is the case.

Example 3.11. Let φ_1 and $\mathcal{TK}$ be as in Example 3.1 with time domain $\mathbb{Q}_{\geq 0}$. Consider the formula $\varphi = E\,\varphi_1\,U\,\neg\varphi_1$. Then $GCD(\mathcal{TK}, \varphi) = 2$ and the corresponding 1-transformation $\mathcal{TK}^1$ is



where the abstract configurations are colored in gray, and we labeled states with the satisfaction of φ_1 and $\neg\varphi_1$ in the pointwise semantics.

Clearly, $\mathcal{TK}^1, (s_0, 0) \models_p E\,\varphi_1\,U\,\neg\varphi_1$. However, $\mathcal{TK}, s_0, \not\models_c E\,\varphi_1\,U\,\neg\varphi_1$. In particular, the satisfaction of φ_1 and $\neg\varphi_1$ in the first 2 time units of each path π in $\mathcal{TK}$ is as follows:



In order for π to satisfy $\varphi_1\,U\,\neg\varphi_1$ in the continuous semantics, it must be the case that there exists a position at some time $\tau > 0$ in π where $\neg\varphi_1$ holds, and φ_1 must hold in all positions before. However, since we have dense time, regardless of how small τ is, there always exists a $0 < \tau' < \tau$ such that, at time τ', φ_1 does not hold.

In general, a path π satisfies the formula $\varphi_1\,U_I\,\varphi_2$ in the continuous semantics iff one of the following formulas holds (in both the continuous and pointwise semantics):

1. $\varphi_1\,U_I$ (φ_2 and "concrete"), that is φ_2 holds at a position in I corresponding to some concrete configuration (and φ_1 holds in all positions before), or
2. $\varphi_1\,U_I$ ($\varphi_1 \wedge \varphi_2$ and "abstract"), that is $\varphi_1 \wedge \varphi_2$ holds at a position in I corresponding to some abstract configuration (and φ_1 holds in all positions before), or
3. the interval I includes time 0 and φ_2 holds immediately (in particular, when the configuration at the first position is an abstract one).

The modification $\beta(\varphi)$ of formula φ captures the above conditions, and therefore has the same satisfaction (in the continuous semantics) as φ. In order to achieve an elegant definition of the modified formula $\beta(\varphi)$, we label all states $(s, (2n+1)\gamma_h)$ in $\mathcal{TK}^{\gamma_h}$, corresponding to the abstract configurations, with a new proposition p_a. In this way, configurations corresponding to some abstract state (i.e., p_a-states) can be distinguished from other concrete configurations (i.e., configurations corresponding to some $\neg p_a$-state) directly in a TCTL formula. We define the *gcd-transformation* $\mathcal{TK}_a^{\gamma_h}$ of a timed Kripke structure to be $\mathcal{TK}^{\gamma_h}$, where the labeling function is extended with this new atomic proposition p_a. We first define a generalization of the aforementioned *gcd*-transformation, i.e. the *abstract* τ*-transformation* $\mathcal{TK}_a^{\tau}$, where τ is any divisor of the half of the computed greatest common divisor.

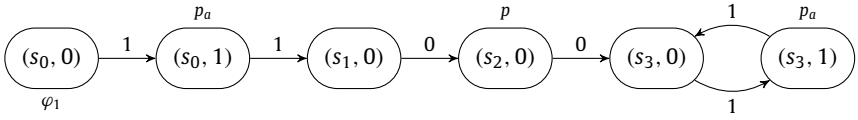
Definition 3.4 *(Abstract τ-transformation).* Let φ be a TCTL formula over AP, $0 < \tau \in \mathcal{T}$ such that $GCD(\mathcal{TK}, \varphi)$ is a multiple of 2τ and $\mathcal{TK}^{\tau} = (S, \mathcal{T}, \longrightarrow, L)$ be the τ-transformation of $\mathcal{TK}$. We define $AP_a = AP \uplus \{p_a\}$ and an extended labeling function $L_a : S \to \mathcal{P}(AP_a)$, such that:

$$\forall (s,\delta) \in S. \quad L_a(s,\delta) = \begin{cases} L(s,\delta) \cup \{p_a\} & \text{if } \exists n \in \mathbb{N} :\ \delta = (2n+1)\tau, \\ L(s,\delta) & \text{otherwise.} \end{cases}$$

We denote the timed Kripke structure $(S, \mathcal{T}, \longrightarrow, L_a)$ by $\mathcal{TK}_a^{\tau}$ and we refer to it as the *abstract* τ*-transformation* of $\mathcal{TK}$. When $2\tau = GCD(\mathcal{TK}, \varphi)$, we denote the abstract τ-transformation of $\mathcal{TK}$ by $\mathcal{TK}_a^{\gamma_h}$ and we refer to it as the *gcd-transformation* of $\mathcal{TK}$.

Fact 3.5. *By definition, the* abstract *τ-transformation does not change the continuous semantics of the τ-transformation, i.e., $\mathcal{TK}^{\tau}, (s,0) \models_c \varphi' \iff \mathcal{TK}_a^{\tau}, (s,0) \models_c \varphi'$, where φ is a TCTL formula over the atomic propositions AP.*

Example 3.12. Let φ, φ_1, and $\mathcal{TK}$ be as in Example 3.1 with time domain $\mathbb{Q}_{\geq 0}$. We have that $GCD(\mathcal{TK}, \varphi) = 2$ and the corresponding *gcd*-transformation $\mathcal{TK}_a^1$ is



where we labeled states with the satisfaction of φ_1 in the pointwise semantics, and with the propositions p and p_a.

Definition 3.5. Let φ be a TCTL$_{cb}$ formula over AP, $0 < \tau \in \mathcal{T}$ such that $GCD(\mathcal{TK}, \varphi)$ is a multiple of 2τ and $\mathcal{TK}_a^\tau = (S, \mathcal{T}, \longrightarrow, L)$ the abstract τ-transformation of $\mathcal{TK}$, so that $L : S \to \mathcal{P}(AP_a)$ and $AP_a = AP \uplus \{p_a\}$ are as given in Definition 3.4. We define the TCTL$_{cb}$ formula $\beta(\varphi)$ inductively as follows:

$$\begin{aligned} \beta(true) &= true; \\ \beta(p) &= p; \\ \beta(\neg\varphi_1) &= \neg\beta(\varphi_1); \\ \beta(\varphi_1 \wedge \varphi_2) &= \beta(\varphi_1) \wedge \beta(\varphi_2); \end{aligned} \qquad \beta(E\ \varphi_1\ U_I\ \varphi_2) = \begin{cases} \beta(\varphi_2) \vee E\ \psi & \text{if } 0 \in I \\ E\ \psi & \text{if } 0 \notin I; \end{cases}$$

$$\beta(A\ \varphi_1\ U_I\ \varphi_2) = \begin{cases} \beta(\varphi_2) \vee A\ \psi & \text{if } 0 \in I \\ A\ \psi & \text{if } 0 \notin I, \end{cases}$$

where $\psi = \beta(\varphi_1)\ U_I\ (\beta(\varphi_2) \wedge (\neg p_a \vee \beta(\varphi_1)))$.

Example 3.13. In Example 3.11, $\mathcal{TK}, s_0 \not\models_c \varphi$ and $\mathcal{TK}_a^1, (s_0, 0) \not\models_p \beta(\varphi)$, since $(s_0, 1)$ is a p_a-state. In particular $\beta(\varphi_1) = E\ F_{=2}\ (p \wedge (\neg p_a \vee true)) = E\ F_{=2}\ p = \varphi_1$ and, therefore, the labeling in the pointwise semantics of $\beta(\varphi_1)$ and $\neg\beta(\varphi_1)$ is the same as the one of, respectively, φ_1 and $\neg\varphi_1$. Furthermore, we have that $\beta(\varphi) = \beta(E\ \varphi_1\ U\ \neg\varphi_1) = \beta(\neg\varphi_1) \vee E\ \beta(\varphi_1)\ U\ (\beta(\neg\varphi_1) \wedge (\neg p_a \vee \beta(\varphi_1)))$, which can be simplified to $\neg\varphi_1 \vee E\ \varphi_1\ U\ (\neg\varphi_1 \wedge \neg p_a)$.

Example 3.14. Consider now φ as in Example 3.12. It is again the case that $\mathcal{TK}, s_0 \not\models_c \varphi$ and $\mathcal{TK}_a^1, (s_0, 0) \not\models_p \beta(\varphi)$. In particular, $\beta(\varphi) = \beta(E\ \varphi_1\ U_{=2}\ true) = E\ \beta(\varphi_1)\ U_{=2}\ (\beta(true) \wedge (\neg p_a \vee \beta(\varphi_1)))$, which can be simplified to $E\ \varphi_1\ U_{=2}\ (\neg p_a \vee \varphi_1)$.

A modification similar to $\beta(\varphi)$ has been used for dense time TCTL model checking of timed automata in [37]. In particular, in [37], in order to prove the lemma that shows the correctness of their labeling algorithm for the region transformation of some timed automata, the authors use a modified formula (i.e., $\varphi_1\ U\ (\varphi_2 \wedge p_{\sim c} \wedge (p_b \vee \varphi_1))$, in our syntax), where the atomic proposition p_b (which, to some extent, is similar to our $\neg p_a$) denotes boundary regions.

Our result holds for any abstract τ-transformation with $GCD(\mathcal{TK}, \varphi)$ a multiple of 2τ, and in particular for the *gcd*-transformation. Theorem 3.1, whose proof is given in Appendix B, shows the equivalence of the pointwise and continuous semantics model checking in $\mathcal{TK}_a^\tau$.

Theorem 3.1. *Let φ be a TCTL$_{cb}$ formula over AP, $0 < \tau \in \mathcal{T}$ such that $GCD(\mathcal{TK}, \varphi)$ is a multiple of 2τ and $\mathcal{TK}_a^\tau = (S, \mathcal{T}, \longrightarrow, L)$ the abstract τ-transformation of $\mathcal{TK}$, so that $L : S \to \mathcal{P}(AP_a)$ and $AP_a = AP \uplus \{p_a\}$ are as given in the definition of $\mathcal{TK}_a^\tau$. Then for each state (s, δ) of $\mathcal{TK}_a^\tau$ and each subformula φ' of φ, it holds that*

$$\mathcal{TK}_a^\tau, (s, \delta) \models_c \varphi' \iff \mathcal{TK}_a^\tau, (s, \delta) \models_p \beta(\varphi').$$

Based on Lemma 3.1, the above transformation of φ and Theorem 3.1, we gain our soundness and completeness result, i.e., model checking $\beta(\varphi)$ in $\mathcal{TK}_a^{\gamma_h}$ in the pointwise semantics is equivalent to model checking the TCTL$_{cb}$ formula φ in $\mathcal{TK}$ in the continuous one. Again, the result is generalized for any $\mathcal{TK}_a^\tau$, such that τ is any divisor of half of the computed greatest common divisor $GCD(\mathcal{TK}, \varphi)$.

Theorem 3.2. *Let AP be a set of atomic propositions and $\mathcal{TK}$ a timed Kripke structure over AP whose time domain satisfies the theory TIME^{gcd}. Let φ be a TCTL$_{cb}$ formula over AP, $0 < \tau \in \mathcal{T}$ such that $GCD(\mathcal{TK}, \varphi)$ is a multiple of 2τ and $\mathcal{TK}_a^\tau = (S, \mathcal{T}, \longrightarrow, L)$ the abstract τ-transformation of $\mathcal{TK}$. Then for each state s of $\mathcal{TK}$ and each subformula φ' of φ it holds that*

$$\mathcal{TK}, s \models_c \varphi' \iff \mathcal{TK}_a^\tau, (s, 0) \models_p \beta(\varphi').$$

Proof. For $\mathcal{TK}$, s, φ' and $\mathcal{TK}_a^{\gamma_h}$ as given in the theorem we have that:

$$\begin{aligned} \mathcal{TK}, s \models_c \varphi' &\overset{\text{by Lemma 3.1}}{\iff} \mathcal{TK}^\tau, (s, 0) \models_c \varphi' \\ &\overset{\text{by Fact 3.5}}{\iff} \mathcal{TK}_a^\tau, (s, 0) \models_c \varphi' \\ &\overset{\text{by Theorem 3.1}}{\iff} \mathcal{TK}_a^\tau, (s, 0) \models_p \beta(\varphi'). \end{aligned}$$

□

3.3. Extending the results to TCTL

In this section we extend the soundness and completeness result of Section 3.2 to the entire TCTL logic by transforming the given TCTL formula φ (with possibly open interval bounds) to a TCTL$_{cb}$ formula $\alpha(\varphi)$, so that $\alpha(\varphi)$ holds in $\mathcal{TK}_a^{\gamma_h}$ in the continuous semantics if and only if φ holds in $\mathcal{TK}_a^{\gamma_h}$ in the continuous semantics:

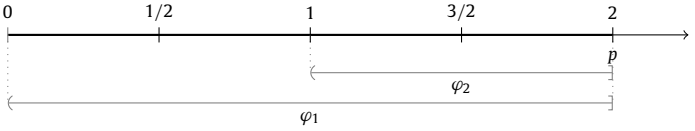
$$\mathcal{TK}_a^{\gamma_h}, (s, \delta) \models_c \varphi \quad \Longleftrightarrow \quad \mathcal{TK}_a^{\gamma_h}, (s, \delta) \models_c \alpha(\varphi).$$

Together with Theorem 3.2, this implies that model checking $\beta(\alpha(\varphi))$ in $\mathcal{TK}_a^{\gamma_h}$ in the pointwise semantics is equivalent to model checking φ in $\mathcal{TK}$ in the continuous one:

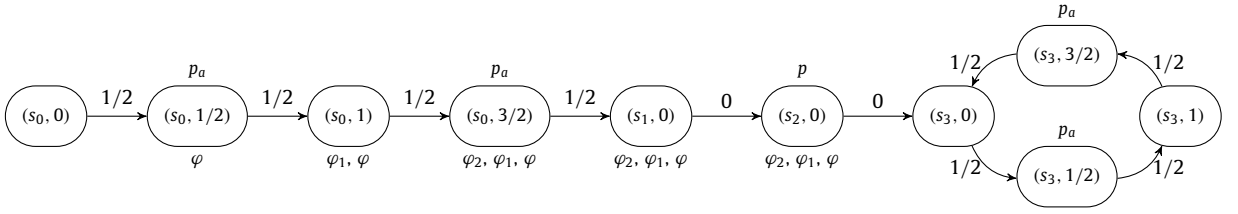
$$\mathcal{TK}, s \models_c \varphi \quad \Longleftrightarrow \quad \mathcal{TK}_a^{\gamma_h}, (s, 0) \models_p \beta(\alpha(\varphi)).$$

We first illustrate by an example why model checking $\beta(\varphi)$ in the pointwise semantics in $\mathcal{TK}_a^{\gamma_h}$ is not enough for a sound and complete model checking of φ in $\mathcal{TK}$ in the continuous semantics, if φ contains open intervals.

Example 3.15. Consider the timed Kripke structure $\mathcal{TK}$ of Example 3.1 with time domain $\mathbb{Q}_{\geq 0}$. Let φ_2 be the formula $E\, F_{<1}\, p$, let φ_1 be $E\, F_{<1}\, \varphi_2$ and let φ be $E\, F_{<1}\, \varphi_1$. The formula φ holds in s_0 in the continuous semantics, since all configurations in the time interval $(1, 2]$ of $\mathcal{TK}$ satisfy φ_2 and all the configurations in the time interval $(0, 2]$ satisfy φ_1, as depicted in the following time line:



However, the formula $\beta(E\, F_{<1}\, E\, F_{<1}\, E\, F_{<1}\, p)$ does not hold in the *gcd*-transformation of $\mathcal{TK}$ in the pointwise semantics. In particular, in the pointwise semantics, the satisfaction of $\beta(\varphi)$ is equivalent to the satisfaction of φ, since, e.g., $\beta(\varphi_2) = p \vee E\, F_{<1}\, p$ which is equivalent to φ_2, and, similarly, $\beta(\varphi_1)$ is equivalent to φ_1. The greatest common divisor of all maximal tick transition durations and all non-zero finite time values in φ is 1, so $\gamma_h = 1/2$, and the transformation $\mathcal{TK}_a^{1/2}$ is the following:



where we also show the labeling of φ, φ_1 and φ_2 in the pointwise semantics. Notice that the configuration at state $(s_0, 1/2)$ is not labeled with φ_1. This leads to the wrong evaluation of φ in $(s_0, 0)$, i.e., we have that $\mathcal{TK}, s_0 \models_c E\, F_{<1}\, \varphi_1$, since, in the continuous semantics there exists a path of $\mathcal{TK}$, starting in s_0 that reaches a φ_1-state in time smaller than 1, while $\mathcal{TK}_a^{1/2}, (s_0, 0) \not\models_p E\, F_{<1}\, \varphi_1$.

Intuitively, for $\mathcal{TK}_a^{1/2}$ to be a sound and complete abstraction for $\mathcal{TK}$ and formula φ, the configuration corresponding to the p_a-state $(s_0, 1/2)$ should be included in the satisfaction set of φ_1.

The key observation (which follows from Facts 3.4 and 3.3) is that a path starting from a concrete configuration will reach the (finite) bounds $inf(I)$ and $sup(I)$ at some concrete configurations. Similarly, a path starting from an abstract configuration will reach the above interval bounds at some abstract configurations. Intuitively, a path π satisfies the TCTL path formula $\varphi_1\, U_{(a,b)}\, \varphi_2$ with the open bound (a, b), if and only if one of the following two conditions holds:

1. The initial configuration is concrete and the formula $\varphi_1\, U_{[a+\gamma_h, b-\gamma_h]}\, \varphi_2$, with the contracted interval $[a + \gamma_h, b - \gamma_h]$ holds in π.
2. The initial configuration is abstract and the formula $\varphi_1\, U_{[a,b]}\, \varphi_2$, with the extended interval $[a, b]$ holds in π.

The above conditions are captured by the modification of a TCTL formula φ into the TCTL$_{cb}$ formula $\alpha(\varphi)$, defined (for a general $\mathcal{TK}_a^{\tau}$ such that τ is a divisor of γ_h) as follows.

Definition 3.6. Let φ be a TCTL formula over AP, $0 < \tau \in \mathcal{T}$ such that $GCD(\mathcal{TK}, \varphi)$ is a multiple of 2τ and $\mathcal{TK}_a^\tau = (S, \mathcal{T}, \longrightarrow, L)$ the abstract τ-transformation of $\mathcal{TK}$, so that $L : S \to \mathcal{P}(AP_a)$ and $AP_a = AP \uplus \{p_a\}$ are as given in Definition 3.4, we define the TCTL$_{cb}$ formula $\alpha(\varphi)$ inductively as follows:

$$\begin{aligned}
\alpha(true) &= true;\\
\alpha(p) &= p;\\
\alpha(\neg\varphi_1) &= \neg\alpha(\varphi_1);\\
\alpha(\varphi_1 \wedge \varphi_2) &= \alpha(\varphi_1) \wedge \alpha(\varphi_2);\\
\alpha(E\ \varphi_1\ U_I\ \varphi_2) &= (\neg p_a \wedge E\ \alpha(\varphi_1)\ U_{I_1}\ \alpha(\varphi_2)) \vee (p_a \wedge E\ \alpha(\varphi_1)\ U_{I_2}\ \alpha(\varphi_2));\\
\alpha(A\ \varphi_1\ U_I\ \varphi_2) &= (\neg p_a \wedge A\ \alpha(\varphi_1)\ U_{I_1}\ \alpha(\varphi_2)) \vee (p_a \wedge A\ \alpha(\varphi_1)\ U_{I_2}\ \alpha(\varphi_2));
\end{aligned}$$

where I_1 is the time interval $[a, b]$, with a and b two time values such that

$$a = \begin{cases} inf(I) & \text{if } inf(I) \in I \\ inf(I) + \tau & \text{otherwise} \end{cases} \qquad b = \begin{cases} sup(I) & \text{if } sup(I) \in I \\ sup(I) \dot{-} \tau & \text{otherwise,} \end{cases}$$

and $I_2 = [inf(I), sup(I)]$.

Fact 3.6. *$\alpha(\varphi)$ preserves the abstract γ_h-transformation for a given timed Kripke structure $\mathcal{TK}$ and TCTL formula φ, since $GCD(\mathcal{TK}, \alpha(\varphi)) = GCD(\mathcal{TK}, \varphi)$.*

Example 3.16. In Example 3.15, $\mathcal{TK}, s_0 \models_c \varphi$ and $\mathcal{TK}_a^{1/2}, (s_0, 0) \models_p \alpha(\varphi)$, since $(s_0, 1/2)$ is a p_a-state. In particular $\alpha(\varphi_2) = (\neg p_a \wedge E\ F_{\leq 1/2}\ p) \vee (p_a \wedge E\ F_{\leq 1}\ p)$, and $\alpha(\varphi_1)$ and $\alpha(\varphi)$ are similar to $\alpha(\varphi_2)$ but with, respectively, φ_2 and φ_1 in place of p. The labeling in the pointwise semantics of $\alpha(\varphi_2)$ is the same as the one of φ_2, while the labeling of $\alpha(\varphi_1)$ includes all the states labeled with φ_1 and, in addition, state $(s_0, 1/2)$.

Theorem 3.3, whose proof is given in Appendix B, shows that, in the continuous semantics, the *closed-bound* formula $\alpha(\varphi)$ holds in $\mathcal{TK}_a^{\gamma_h}$ if and only if the (possibly open-bound) formula φ does so. The result is generalized for any $\mathcal{TK}_a^\tau$ such that τ is a divisor of γ_h.

Theorem 3.3. *Let φ be a TCTL formula over AP, $0 < \tau \in \mathcal{T}$ a time value such that $GCD(\mathcal{TK}, \varphi)$ is a multiple of 2τ and $\mathcal{TK}_a^\tau = (S, \mathcal{T}, \longrightarrow, L)$ the abstract τ-transformation of $\mathcal{TK}$, so that $L : S \to \mathcal{P}(AP_a)$ and $AP_a = AP \uplus \{p_a\}$ are as given in Definition 3.4. Moreover, let $\alpha(\varphi)$ be defined as above, then the following holds for each state (s, δ) in $\mathcal{TK}_a^\tau$ and each subformula φ' of φ:*

$$\mathcal{TK}_a^\tau, (s, \delta) \models_c \varphi' \iff \mathcal{TK}_a^\tau, (s, \delta) \models_c \alpha(\varphi').$$

Based on Lemma 3.1, Theorem 3.1, and the above Theorem 3.3, we gain our soundness and completeness result for the entire TCTL, i.e., model checking $\beta(\alpha(\varphi))$ in $\mathcal{TK}_a^{\gamma_h}$ in the pointwise semantics is equivalent to model checking the TCTL formula φ in $\mathcal{TK}$ in the continuous one. Again, the result is generalized for any $\mathcal{TK}_a^\tau$, such that τ is a divisor of the half of the computed greatest common divisor $GCD(\mathcal{TK}, \varphi)$.

Theorem 3.4. *Let AP be a set of atomic propositions and $\mathcal{TK}$ a timed Kripke structure over AP whose time domain satisfies the theory TIME^{gcd}. Let φ be a TCTL formula over AP, $0 < \tau \in \mathcal{T}$ such that $GCD(\mathcal{TK}, \varphi)$ is a multiple of 2τ and $\mathcal{TK}_a^\tau = (S, \mathcal{T}, \longrightarrow, L)$ the abstract τ-transformation of $\mathcal{TK}$. Then for each state s of $\mathcal{TK}$ and each subformula φ' of φ:*

$$\mathcal{TK}, s \models_c \varphi' \iff \mathcal{TK}_a^\tau, (s, 0) \models_p \beta(\alpha(\varphi')).$$

Proof. For $\mathcal{TK}$, s, φ' and $\mathcal{TK}_a^\tau$ as given in the theorem we have that:

$$\begin{aligned}
\mathcal{TK}, s \models_c \varphi' &\overset{\text{by Lemma 3.1}}{\iff} \mathcal{TK}^\tau, (s, 0) \models_c \varphi'\\
&\overset{\text{by Fact 3.5}}{\iff} \mathcal{TK}_a^\tau, (s, 0) \models_c \varphi'\\
&\overset{\text{by Theorem 3.3}}{\iff} \mathcal{TK}_a^\tau, (s, 0) \models_c \alpha(\varphi')\\
&\overset{\text{by Theorem 3.1}}{\iff} \mathcal{TK}_a^\tau, (s, 0) \models_p \beta(\alpha(\varphi')). \qquad \square
\end{aligned}$$

3.4. Sound and complete model checking for discrete time

When the time domain is the natural numbers $\mathbb{N}$, sound and complete model checking for a TCTL formula φ and a timed Kripke structure $\mathcal{TK}$ can be achieved by exhaustively visiting all time instants, which can be done by splitting each tick transition in $\mathcal{TK}$ with smaller ones of duration one (i.e., by model checking the 1-transformation of $\mathcal{TK}$).[5]

Theorem 3.5. *Let AP be a set of atomic propositions and $\mathcal{TK}$ a timed Kripke structure over AP whose time domain is $\mathbb{N}$. Let φ be a TCTL formula over AP, and $\mathcal{TK}^1 = (S, \mathbb{N}, \longrightarrow, L)$ the 1-transformation of $\mathcal{TK}$. Then for each state s of $\mathcal{TK}$ and each subformula φ' of φ:*

$$\mathcal{TK}, s \models_c \varphi' \quad \Longleftrightarrow \quad \mathcal{TK}^1, (s, 0) \models_p \varphi'.$$

Proof. If the time domain is $\mathbb{N}$, a bijective mapping $f : \mathcal{C}^c_{\mathcal{TK}} \to \mathcal{C}^p_{\mathcal{TK}^1}$ with $f(\langle s, \delta\rangle) = \langle (s, \delta), 0\rangle$ can be defined between the valid configurations of $\mathcal{TK}$ and the valid configurations of $\mathcal{TK}^1$, such that the continuous interpretation of $\mathcal{TK}$ is equivalent to the pointwise interpretation of $\mathcal{TK}^1$. Each instantaneous step $\langle s, 0\rangle \overset{0}{\hookrightarrow} \langle s', 0\rangle$ in $\mathcal{TK}$ corresponds to the instantaneous step $f(\langle s, 0\rangle) \overset{0}{\hookrightarrow} f(\langle s', 0\rangle)$, i.e. $\langle (s, 0), 0\rangle \overset{0}{\hookrightarrow} \langle (s', 0), 0\rangle$, in $\mathcal{TK}^1$. Furthermore, each tick step $\langle s, \delta\rangle \overset{\tau}{\hookrightarrow} \langle s', \delta'\rangle$ in the continuous interpretation $\mathcal{TK}$ can be simulated by the following sequence of τ tick steps, each of duration 1, in the pointwise interpretation of $\mathcal{TK}^1$:

$$\langle (s, \delta), 0\rangle \overset{1}{\hookrightarrow} \langle (s, \delta + 1), 0\rangle \overset{1}{\hookrightarrow} \langle (s, \delta + 2), 0\rangle \overset{1}{\hookrightarrow} \cdots \overset{1}{\hookrightarrow} \langle (s', \delta'), 0\rangle$$

According to the above correspondence between states and transition steps, we can easily define a TCTL-preserving bijective mapping between the time-divergent paths in the continuous interpretation of $\mathcal{TK}$ and the time-divergent paths in the pointwise interpretation of $\mathcal{TK}^1$ such that corresponding paths satisfy the same TCTL path formulas over *AP*. □

Fact 3.7. *The continuous interpretation of $\mathcal{TK}^1$ and the pointwise interpretation of $\mathcal{TK}^1$ are equivalent, since each tick step in $\mathcal{TK}^1$ is a maximal one, and therefore its one-step transition relation uses only rules* $\texttt{Rule}_{\texttt{tick1}}$ *and* $\texttt{Rule}_{\texttt{tick2}}$ *in both interpretations. Consequently, model checking $\mathcal{TK}^1$ in the continuous semantics is equivalent to model checking it in the pointwise semantics, i.e., for each state s in $\mathcal{TK}^1$ and each subformula φ' of a TCTL formula φ over AP:*

$$\mathcal{TK}^1, (s, 0) \models_c \varphi' \quad \Longleftrightarrow \quad \mathcal{TK}^1, (s, 0) \models_p \varphi'.$$

The heavy price of achieving soundness and completeness for discrete time in this way has to do with the fact that visiting all discrete times typically leads to a state space explosion that renders model checking infeasible. For example, this procedure is inefficient if discrete events do not happen for relatively long time intervals, e.g., if each tick transition duration in the system and each finite time bound appearing in the formula is a multiple of 10 000 time units.

A more efficient model checking procedure can be achieved by adapting our *gcd*-transformation to the time domain $\mathbb{N}$. The *gcd*-transformation is directly applicable to timed Kripke structures over the time domain $\mathbb{N}$, if the greatest common divisor γ is an even number, i.e., if $\gamma_h = \mathit{half}(\gamma)$ is defined.

If the greatest common divisor is odd but larger than 1, we can multiply each finite constant appearing in the timed Kripke structure and in the TCTL formula by 2. This transformation does not change the TCTL semantics of the formula, but the greatest common divisor becomes even, such that our approach can be applied. The following fact formalizes that this scaling is satisfiability-preserving.

Fact 3.8. *For each state s in $\mathcal{TK}$ with time domain $\mathbb{N}$ and each subformula φ' of a TCTL formula φ over AP with $GCD(\mathcal{TK}, \varphi) > 1$:*

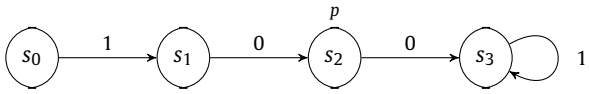
$$\mathcal{TK}, s \models_c \varphi' \quad \Longleftrightarrow \quad (2 \cdot \mathcal{TK})^{\gamma_h}_a, (s, 0) \models_p 2 \cdot \beta(\alpha(\varphi')),$$

where $2 \cdot \mathcal{TK}$ is obtained from $\mathcal{TK}$ by multiplying each finite transition duration by 2, and $2 \cdot \beta(\alpha(\varphi'))$ is obtained from $\beta(\alpha(\varphi'))$ by multiplying each finite time value appearing in some interval bound by 2.

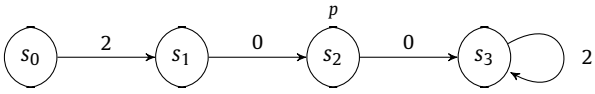
If the greatest common divisor is 1 then the formula might evaluate differently for discrete and for dense time domains, since satisfaction might depend on whether there is any configuration reachable between two successive discrete time points. Scaling up durations in a discrete time domain with greatest common divisor 1 would correspond to inserting such reachable configurations and therefore it could change the evaluation of TCTL formulas in the continuous semantics, as illustrated by Example 3.17 below. Instead, the 1-transformation (without scaling the model) gives us the correct result, as stated in Theorem 3.5.

[5] When the time domain is $\mathbb{N}$, all transitions in $\mathcal{TK}^1$ have duration 0 or 1. Such a timed Kripke structure corresponds to a *small-step* DTGs (a DTG with transitions of duration equal to 0 or 1) in [7], where it is also discussed that the continuous semantics and the pointwise semantics coincide for small-step DTGs.

Example 3.17. Assume the time domain $\mathbb{N}$, the formula $\varphi = E\ \varphi_1\ U_{=1}\ true$ with $\varphi_1 = E\ F_{=1}\ p$ and the following timed Kripke structure $\mathcal{TK}$:



Obviously, γ is 1. Multiplying each constant by 2 gives us the formula $\varphi' = 2 \cdot \varphi = E\ \varphi_1'\ U_{=2}\ true$ with $\varphi_1' = 2 \cdot \varphi_1 = E\ F_{=2}\ p$, and the following timed Kripke structure $\mathcal{TK}' = 2 \cdot \mathcal{TK}$:



That such a scaling is not TCTL-preserving can be demonstrated by $\mathcal{TK}, s_0 \models_c \varphi$ but $\mathcal{TK}', s_0 \not\models_c \varphi'$.

4. TCTL model checking of timed Kripke structures in the pointwise semantics

This section describes our model checking procedure for TCTL formulas over timed Kripke structures in the pointwise semantics. Our procedure is based on the explicit-state CTL model checking approach [38] that recursively computes, for each subformula of the desired TCTL formula, the set of reachable states satisfying the subformula. In particular, we

1. show that any TCTL formula is equivalent to one in *normal form*; and
2. describe a model checking procedure for TCTL formulas in normal form.

Section 4.1 explains how this normal form is defined. Section 4.2 gives an overview of our model checking procedure, whose correctness is proved in Section 4.3.

The correctness of our model checking procedure relies on the assumption that the timed Kripke structures we consider are finite and Zeno-free, implying that in the pointwise interpretation all paths in the timed Kripke structure are *time-divergent*. As stated in Fact 2.1, a finite timed Kripke structure $\mathcal{TK}$ is Zeno-free iff it has no zero-time loops. Hence checking Zeno-freeness of $\mathcal{TK}$ can be done in linear time as follows: Let $\mathcal{TK}_0$ be $\mathcal{TK}$ where all tick transitions have been removed (i.e., only instantaneous transitions are left). Then $\mathcal{TK}$ is Zeno-free iff $\mathcal{TK}_0$ has no loops.

Our model checking procedure adapts the one in [7], which is developed for $\text{TCTL}_{\leq\geq}$ formulas,[6] by extending it to formulas with *interval time bounds* and by simplifying some of the core procedures, which is possible because the assumption of Zeno-freeness excludes zero-time loops.

To simplify the notation, in the following we sometimes write state s for the configuration $\langle s, 0\rangle$, and write $s \xrightarrow{\tau} s'$ for a step $\langle s, 0\rangle \overset{\tau}{\hookrightarrow} \langle s, 0\rangle$.

4.1. Normal form of a TCTL formula

Any TCTL formula can be reduced to one in a *normal form*.

Definition 4.1 *(TCTL normal form).* Given a set AP of atomic propositions and a time domain $\mathcal{T}$, TCTL formulas in *normal form* are built using the following grammar:

$$\varphi ::= true \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid E\ G\ \varphi \mid E\ \varphi\ U_I\ \varphi \mid A\ \varphi\ U_{I_{>0}}\ \varphi$$

where $p \in AP$, $I, I_{>0} \in Intervals(\mathcal{T})$, and $inf(I_{>0}) = 0 \notin I_{>0}$.

Table 2 lists the equivalences that are applied to subformulas of a TCTL formula φ to reduce it to its normal form $\varphi!$. Equivalences might have a validity condition (i.e., a condition under which they are applicable). Additionally, we only apply equivalences to formulas that are not yet in normal form, expressed by the application conditions. The equivalences are applied with a priority that depends on the order in which they are listed in Table 2: when more than one equivalence can be applied for reducing the same formula, the first listed one in the table is chosen.[7]

[6] $\text{TCTL}_{\leq\geq}$ is the TCTL fragment where the time constraints have the form $\sim k$ with $\sim \in \{<, \leq, \geq, >\}$.

[7] The right-hand side of an equivalence is not necessarily in normal form and may need to be further reduced by applying other equivalences.

Table 2
Equivalences for reducing a TCTL formula to its normal form.

Equivalence			Validity condition	Application condition
4.1.	$false$	$= \neg true$		
4.2.	$\varphi_1 \vee \varphi_2$	$= \neg(\neg\varphi_1 \wedge \neg\varphi_2)$		
4.3.	$\varphi_1 \Longrightarrow \varphi_2$	$= \neg\varphi_1 \vee \varphi_2$		
4.4.	$\varphi_1 \Longleftrightarrow \varphi_2$	$= (\varphi_1 \Longrightarrow \varphi_2) \wedge (\varphi_2 \Longrightarrow \varphi_1)$		
4.5.	$E\ F_I\ \varphi$	$= E\ true\ U_I\ \varphi$		
4.6.	$A\ F_I\ \varphi$	$= A\ true\ U_I\ \varphi$		
4.7.	$E\ G_{\leq b}\ \varphi$	$= E\ \varphi\ U_{>b}\ true$		
4.8.	$E\ G_{<b}\ \varphi$	$= E\ \varphi\ U_{\geq b}\ true$	$if\ b \neq \infty$	
4.9.	$E\ G_I\ \varphi$	$= \neg A\ F_I\ \neg\varphi$		$if\ I \neq [0, \infty)$
4.10.	$A\ G_I\ \varphi$	$= \neg E\ F_I\ \neg\varphi$		
4.11.	$A\ \varphi_1\ U\ \varphi_2$	$= (\neg E\ G\ \neg\varphi_2) \wedge \neg E\ (\neg\varphi_2)\ U\ (\neg\varphi_1 \wedge \neg\varphi_2)$		
4.12.	$A\ \varphi_1\ U_{\leq b}\ \varphi_2$	$= (\neg E\ G_{\leq b}\ \neg\varphi_2) \wedge \neg E\ (\neg\varphi_2)\ U\ (\neg\varphi_1 \wedge \neg\varphi_2)$		
4.13.	$A\ \varphi_1\ U_{<b}\ \varphi_2$	$= (\neg E\ G_{<b}\ \neg\varphi_2) \wedge \neg E\ (\neg\varphi_2)\ U\ (\neg\varphi_1 \wedge \neg\varphi_2)$		
4.14.	$A\ \varphi_1\ U_{\geq a}\ \varphi_2$	$= A\ G_{<a}\ (\varphi_1 \wedge A\ \varphi_1\ U_{>0}\ \varphi_2)$	$if\ a \neq 0$	
4.15.	$A\ \varphi_1\ U_{>a}\ \varphi_2$	$= A\ G_{\leq a}\ (\varphi_1 \wedge A\ \varphi_1\ U_{>0}\ \varphi_2)$		$if\ a \neq 0$
4.16.	$A\ \varphi_1\ U_{[a,b]}\ \varphi_2$	$= (\neg E\ F_{=a}\ (E\ (\varphi_1 \wedge \neg\varphi_2)\ U\ (\neg\varphi_1 \wedge \neg\varphi_2))) \wedge$ $(\neg E\ F_{=a}\ (E\ \neg\varphi_2\ U_{>b-a}\ true)) \wedge$ $(\neg E\ F_{<a}\ \neg\varphi_1)$	$if\ a \neq 0$	
4.17.	$A\ \varphi_1\ U_{[a,b)}\ \varphi_2$	$= (\neg E\ F_{=a}\ (E\ (\varphi_1 \wedge \neg\varphi_2)\ U\ (\neg\varphi_1 \wedge \neg\varphi_2))) \wedge$ $(\neg E\ F_{=a}\ (E\ \neg\varphi_2\ U_{\geq b-a}\ true)) \wedge$ $(\neg E\ F_{<a}\ \neg\varphi_1)$	$if\ a \neq 0 \wedge b \neq \infty$	
4.18.	$A\ \varphi_1\ U_{(a,b]}\ \varphi_2$	$= (A\ F_{=a}\ (A\ \varphi_1\ U_{(0,b-a]}\ \varphi_2)) \wedge A\ G_{\leq a}\ \varphi_1$	$if\ b \neq \infty$	$if\ a \neq 0$
4.19.	$A\ \varphi_1\ U_{(a,b)}\ \varphi_2$	$= (A\ F_{=a}\ (A\ \varphi_1\ U_{(0,b-a)}\ \varphi_2)) \wedge A\ G_{\leq a}\ \varphi_1$	$if\ b \neq \infty$	$if\ a \neq 0$

Equivalences 4.1–4.10 are due to the syntactic sugar definitions for TCTL, with the exception of Equivalences 4.7 and 4.8, which hold because of time-divergence. Equivalences 4.11–4.19 cover the reduction of universal until formulas, distinguishing on the type of the time interval on the until operator (except the type $A\ \varphi_1\ U_{I_{>0}}\ \varphi_2$ which is allowed in the normal form and does not need to be reduced). The unbounded case 4.11 is a well-known CTL equivalence. Equivalences 4.12 and 4.13 follow directly from the following Equivalence (E1) in [7]:

$$(E1)\ \ A\ \varphi_1\ U_{\leq c}\ \varphi_2 = (A\ F_{\leq c}\ \varphi_2)\ \wedge\ \neg E\ (\neg\varphi_2)\ U\ (\neg\varphi_1 \wedge \neg\varphi_2)$$

Equivalences 4.14 and 4.15, which hold because of the assumption of time-divergence, follow directly from the following Equivalence (E2) [7]:

$$(E2)\ \ A\ \varphi_1\ U_{\geq c}\ \varphi_2 = A\ G_{<c}\ (\varphi_1 \wedge A\ \varphi_1\ U_{>0}\ \varphi_2) \quad \text{if } c > 0$$

Equivalences 4.16 and 4.17 generalize 4.12 and 4.13 to a "general interval" time bound. The formula on the right-hand side of Equivalence 4.16 is a conjunction, where $\neg E\ F_{<a}\ \neg\varphi_1$ requires φ_1 to hold all the time before bound a is reached in each behavior; $\neg E\ F_{=a}\ (E\ \neg\varphi_2\ U_{>b-a}\ true)$ requires that in each behavior φ_2 will hold somewhere in the time interval $[a, b]$; finally, $\neg E\ F_{=a}\ (E\ (\varphi_1 \wedge \neg\varphi_2)\ U\ (\neg\varphi_1 \wedge \neg\varphi_2))$ requires that, in each behavior, once we are in the interval, φ_1 holds until a φ_2-state is eventually reached.

Equivalences 4.18 and 4.19 also generalize Equivalences 4.14 and 4.15 to a "general interval" time bound. The right-hand side of 4.18 is the conjunction of two formulas, where $A\ G_{\leq a}\ \varphi_1$ requires that φ_1 holds all the time before the interval is reached; while $A\ F_{=a}\ (A\ \varphi_1\ U_{(0,b-a]}\ \varphi_2)$ requires that in any path, at time a, we reach a state where (again in any path) φ_1 holds until a φ_2-state is reached within the time-interval $(0, b-a]$ ("shifting" the interval $(0, b-a]$ of a time units results in the interval $(0+a, b-a+a] = (a, b]$).

Fact 4.1. *Since each equivalence in Table 2 preserves validity in the pointwise semantics, we have*

$$\mathcal{TK}, s \models_p \varphi \quad \Longleftrightarrow \quad \mathcal{TK}, s \models_p \varphi!.$$

4.2. Core model checking procedures

Procedure 1 gives a high-level overview of our TCTL model checking procedure. Given a timed Kripke structure $\mathcal{TK} = (S, \mathcal{T}, \longrightarrow, L)$, a state $s \in S$ and a TCTL formula φ, the model checking procedure $\mathsf{MC}(\mathcal{TK}, s, \varphi)$ establishes whether $\mathcal{TK}, s \models_p \varphi$ by recursively computing the satisfaction set of the normal form $\varphi!$ of φ.

The recursive computation of the satisfaction set $\mathsf{Sat}(\mathcal{TK}, \varphi!)$ is given as Procedure 2. It relies on six *core procedures*, each taking care of a certain normal form modality:

Procedure 1 MC($\mathcal{TK}, s, \varphi$)

Input : $\mathcal{TK} = (S, \mathcal{T}, \longrightarrow, L)$,
TCTL formula φ.

Output : $\mathcal{TK}, s \models_p \varphi$.

Q := Sat($\mathcal{TK}, \varphi$!); // Recursively compute satisfaction set of normalized formula φ!
return $s \in$ Q;

(i). Sat-EU for formulas $E\ \varphi_1\ U\ \varphi_2$;
(ii). Sat-EG for formulas $E\ G\ \varphi$;
(iii). Sat-EU$_b$ for formulas $E\ \varphi_1\ U_{\sim b}\ \varphi_2$, with $\sim \in \{\leq, <\}$, except the modality covered in (i);
(iv). Sat-EU$_a$ for formulas $E\ \varphi_1\ U_{\sim a}\ \varphi_2$, with $\sim \in \{\geq, >\}$, except the modality covered in (i);
(v). Sat-EU$_{a,b}$ for formulas $E\ \varphi_1\ U_I\ \varphi_2$, except the modalities covered in (i), (iii), and (iv); and
(vi). Sat-AU$_0$ for formulas $A\ \varphi_1\ U_{I_{>0}}\ \varphi_2$.

Procedure 2 Sat($\mathcal{TK}, \varphi$)

Input : $\mathcal{TK} = (S, \mathcal{T}, \longrightarrow, L)$,
TCTL formula φ in normal form.

Output : $Sat(\varphi) = \{s \in S \mid \mathcal{TK}, s \models_p \varphi\}$.

// Recursively evaluate all subformulas of φ

switch φ :
- $true$: **return** S;
- p : **return** $\{s \mid p \in L(s)\}$;
- $\neg\varphi_1$: **return** $S \setminus$ Sat($\mathcal{TK}, \varphi_1$);
- $\varphi_1 \wedge \varphi_2$: **return** Sat($\mathcal{TK}, \varphi_1$) $\cap$ Sat($\mathcal{TK}, \varphi_2$);
- $\varphi_1 \vee \varphi_2$: **return** Sat($\mathcal{TK}, \varphi_1$) $\cup$ Sat($\mathcal{TK}, \varphi_2$);
- $E\ \varphi_1\ U\ \varphi_2$: **return** Sat-EU($\mathcal{TK}, \varphi$); // do Procedure 3
- $E\ G\ \varphi_1$: **return** Sat-EG($\mathcal{TK}, \varphi$); // do Procedure 4
- $E\ \varphi_1\ U_{\leq b}\ \varphi_2$:
  $E\ \varphi_1\ U_{<b}\ \varphi_2$: **return** Sat-EU$_b$($\mathcal{TK}, \varphi$); // do Procedure 5
- $E\ \varphi_1\ U_{\geq a}\ \varphi_2$:
  $E\ \varphi_1\ U_{>a}\ \varphi_2$: **return** Sat-EU$_a$($\mathcal{TK}, \varphi$); // do Procedure 6
- $E\ \varphi_1\ U_{[a,b]}\ \varphi_2$:
  $E\ \varphi_1\ U_{(a,b]}\ \varphi_2$:
  $E\ \varphi_1\ U_{[a,b)}\ \varphi_2$:
  $E\ \varphi_1\ U_{(a,b)}\ \varphi_2$: **return** Sat-EU$_{a,b}$($\mathcal{TK}, \varphi$); // do Procedure 7
- $A\ \varphi_1\ U_{(0,\infty)}\ \varphi_2$:
  $A\ \varphi_1\ U_{(0,b]}\ \varphi_2$:
  $A\ \varphi_1\ U_{(0,b)}\ \varphi_2$: **return** Sat-AU$_0$($\mathcal{TK}, \varphi$); // do Procedure 8

endsw

Procedure 3 describes the algorithm Sat-EU($\mathcal{TK}, \varphi$) for computing the satisfaction set of a formula φ of the kind $E\ \varphi_1\ U\ \varphi_2$. We follow the classic explicit model checking algorithm for CTL [38]. All states satisfying φ_2 also satisfy φ. The set of φ_2-states (i.e., Q = $\{s \in Sat(\varphi_2)\}$) is recursively closed with the set $\mathsf{Q}_{\mathsf{pre}}$ of all φ_1-states, which are not already in Q and which can reach some state in Q in one step, until a fixed-point is reached.

Procedure 4 describes the algorithm Sat-EG($\mathcal{TK}, \varphi$) for computing the satisfaction set for a formula φ of the form $E\ G\ \varphi_1$. We again follow a classic explicit model checking algorithm for CTL. Tarjan(Q$_1$, $\longrightarrow$) computes all the states which belong to some non-trivial strongly connected component[8] of φ_1-states ($SCC(\varphi_1)$) over the given transition relation, using the well-known Tarjan algorithm for discovering strongly connected components. From each state in some $SCC(\varphi_1)$ there exists an infinite behavior consisting of φ_1-states, since each SCC contains looping behaviors, therefore φ holds in these states. Then, this initial set of states Q = $\{s \in SCC(\varphi_1)\}$ is recursively closed with the set $\mathsf{Q}_{\mathsf{pre}}$ of all φ_1-states, which are not already in Q and which can reach some state in Q in one step, until a fixed-point is reached.

Procedure 5 describes the algorithm Sat-EU$_b$($\mathcal{TK}, \varphi$) for computing the satisfaction set for a formula φ of the form $E\ \varphi_1\ U_{\sim b}\ \varphi_2$ with $\sim \in \{\leq, <\}$. We can compute the satisfaction set of such a formula using a procedure similar to the one

[8] A *strongly connected component* of a directed graph is a maximal set of nodes such that for each s_1, s_2 from the set there is a directed path from s_1 to s_2 and vice versa. A *non-trivial* strongly connected component has at least one edge.

Procedure 3 Sat-EU($\mathcal{TK}, \varphi$)

Input : $\mathcal{TK} = (S, \mathcal{T}, \longrightarrow, L)$,
$\varphi = E\ \varphi_1\ U\ \varphi_2$;

Output : $Sat(\varphi) = \{s \in S \mid \mathcal{TK}, s \models_p \varphi\}$.

$\mathsf{Q}_1 := \mathsf{Sat}(\mathcal{TK}, \varphi_1)$;
$\mathsf{Q} := \mathsf{Sat}(\mathcal{TK}, \varphi_2)$;
$\mathsf{Q}_{\mathsf{pre}} := \{s' \in \mathsf{Q}_1 \setminus \mathsf{Q} \mid \exists s \in \mathsf{Q}, \tau \in \mathcal{T} : (s', \tau, s) \in \longrightarrow\}$;
while $\mathsf{Q}_{\mathsf{pre}} \neq \emptyset$ **do**
  $\mathsf{Q} := \mathsf{Q} \cup \mathsf{Q}_{\mathsf{pre}}$;
  $\mathsf{Q}_{\mathsf{pre}} := \{s' \in \mathsf{Q}_1 \setminus \mathsf{Q} \mid \exists s \in \mathsf{Q}, \tau \in \mathcal{T} : (s', \tau, s) \in \longrightarrow\}$;
od
return Q;

Procedure 4 Sat-EG($\mathcal{TK}, \varphi$)

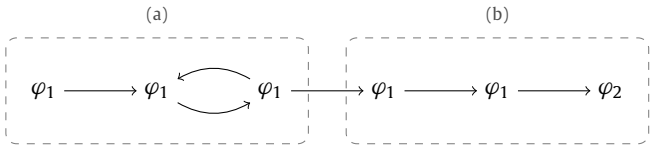
Input : $\mathcal{TK} = (S, \mathcal{T}, \longrightarrow, L)$,
$\varphi = E\ G\ \varphi_1$;

Output : $Sat(\varphi) = \{s \in S \mid \mathcal{TK}, s \models_p \varphi\}$.

$\mathsf{Q}_1 := \mathsf{Sat}(\mathcal{TK}, \varphi_1)$;
$\mathsf{Q} := \mathsf{Tarjan}(\mathsf{Q}_1, \longrightarrow)$; // set of all states in any $SCC(\varphi_1)$
$\mathsf{Q}_{\mathsf{pre}} := \{s' \in \mathsf{Q}_1 \setminus \mathsf{Q} \mid \exists s \in \mathsf{Q}, \tau \in \mathcal{T} : (s', \tau, s) \in \longrightarrow\}$;
while $\mathsf{Q}_{\mathsf{pre}} \neq \emptyset$ **do**
  $\mathsf{Q} := \mathsf{Q} \cup \mathsf{Q}_{\mathsf{pre}}$;
  $\mathsf{Q}_{\mathsf{pre}} := \{s' \in \mathsf{Q}_1 \setminus \mathsf{Q} \mid \exists s \in \mathsf{Q}, \tau \in \mathcal{T} : (s', \tau, s) \in \longrightarrow\}$;
od
return Q;

in [7]. The procedure computes the shortest distances from each φ_1-state to a φ_2-state along a $(\varphi_1\ U\ \varphi_2)$-path. First the satisfaction set Q_u of the untimed formula $\varphi_u = E\ \varphi_1\ U\ \varphi_2$ is computed, and the shortest path computation is restricted to the graph $G = (\mathsf{Q}_u, \mathsf{TR})$, where TR is the set of transitions between φ_u-states, excluding outgoing transitions from φ_2-states, since these transitions would not contribute to finding such shortest paths. Then, using a procedure inspired by Dijkstra's shortest path algorithm, the shortest distance from each φ_1-state to a φ_2-state along the paths in TR is computed. This is achieved by initially putting in the set T of (*state*, *distance*) pairs the φ_2-states in Q_u with distance zero, and then iteratively picking the pair $(s, \tau) \in \mathsf{T}$ with the minimum distance and adding to T the set of pairs $\mathsf{T}_{\mathsf{pre}}$, of predecessors of s that can reach a φ_2-state through a transition in TR "within the bound b". The procedure returns the states having shortest distance within the bound, which are stored in Q.

Procedure 6 describes the algorithm $\mathsf{Sat\text{-}EU}_{\mathsf{a}}(\mathcal{TK}, \varphi)$ for computing the satisfaction set for a formula φ of the form $E\ \varphi_1\ U_{\sim a}\ \varphi_2$ with $\sim \in \{\geq, >\}$ and either $\sim$ is equal to $>$ or $a \neq 0$. This procedure is again an adaptation of the one in [7]. In general, a state satisfying φ appears on a path of the kind



The procedure will compute the two sets of φ-states (intuitively grouped by dashed boxes in the above figure):

(a) the set of states that satisfy φ due to a looping sequence of φ_1-states leading to some φ_2-state; having a finite set of states, looping sequences of φ_1-states can be detected by finding $SCC(\varphi_1)$. Because of the assumption of time-divergence, any non-trivial strongly connected component in $\mathcal{TK}$ has at least one tick transition with duration greater than 0;
(b) the set of states that satisfy φ due to a simple (non-looping) $(\varphi_1\ U\ \varphi_2)$-path, whose duration is higher than the required time bound.

For computing the states described in (a), first, the sets Q_1 and Q_u are computed as in Procedure 5. Then the set $\mathsf{Q}_{\mathsf{SCC}}$ of states belonging to some non-trivial strongly connected components of φ_1 states in $\mathcal{TK}$ is computed, by calling $\mathsf{Tarjan}(\mathsf{Q}_1, \longrightarrow)$. Q initially contains states which are both in $\mathsf{Q}_{\mathsf{SCC}}$ and satisfying $E\ \varphi_1\ U\ \varphi_2$, and hence satisfy φ. In

Procedure 5 Sat-EU$_b(\mathcal{TK}, \varphi)$

Input : $\mathcal{TK} = (S, \mathcal{T}, \longrightarrow, L)$,
$\varphi = E\ \varphi_1\ U_I\ \varphi_2$, with $inf(I) = 0 \in I$, $b = sup(I)$;
$\sim = (\text{if } b \in I \text{ then } \leq \text{ else } <)$.

Output : $Sat(\varphi) = \{s \in S \mid \mathcal{TK}, s \models_p \varphi\}$.

$\mathsf{Q}_1 := \mathsf{Sat}(\mathcal{TK}, \varphi_1)$;
$\mathsf{Q}_2 := \mathsf{Sat}(\mathcal{TK}, \varphi_2)$;
$\mathsf{Q}_u := \mathsf{Sat}(\mathcal{TK}, E\ \varphi_1\ U\ \varphi_2)$;
$\mathsf{T} := \{(s, 0) \mid s \in \mathsf{Q}_2\}$;
$\mathsf{TR} := \{(s, \tau, s') \in \longrightarrow \mid s, s' \in \mathsf{Q}_u \wedge s \in \mathsf{Q}_1 \setminus \mathsf{Q}_2\}$; // $(\varphi_1\ U\ \varphi_2)$-paths
$\mathsf{Q} := \emptyset$;
/* compute shortest distances (up to b) to a φ_2-state in TR */
while $\mathsf{T} \neq \emptyset$ **do**
  let $(s, \tau) \in \mathsf{T}\ s.t.\ \forall (s', \tau') \in \mathsf{T}.\ \tau \leq \tau'$; // s has minimal distance
  $\mathsf{T}_{pre} := \{(s', \tau + \tau') \mid (s', \tau', s) \in \mathsf{TR} \wedge (\tau + \tau' \sim b)\}$;
  $\mathsf{T} := (\mathsf{T} \setminus \{(s, \tau)\}) \cup \mathsf{T}_{pre}$;
  $\mathsf{TR} := \{(s', \tau', s'') \in \mathsf{TR} \mid s'' \neq s\}$;
  $\mathsf{Q} := \mathsf{Q} \cup \{s\}$;
od
return Q;

addition, the set Q is recursively closed with the set Q_{pre} of all φ_1-states, which are not already in Q and which can reach some state in Q in one step, until a fixed-point is reached.

For states described in (b), we first restrict to the graph $\mathsf{DAG} = (\mathsf{Q}_{DAG}, \mathsf{TR}_{DAG})$, where Q_{DAG} is the set of $(E\ \varphi_1\ U\ \varphi_2)$-states, which do not belong to the set of states computed in point (a), and TR_{DAG} is the set of outgoing transitions from φ_1-states in Q_{DAG}. Outgoing transitions from $(\varphi_2 \wedge \neg\varphi_1)$-states are not included in TR_{DAG}, since they are irrelevant for our aim of computing the maximal durations from each φ_1-state from Q_{DAG} to any φ_2-state along $(\varphi_1\ U\ \varphi_2)$-paths. Having removed all cycles, DAG is a directed acyclic graph,[9] and the longest paths from a φ_1-state to some φ_2 state in it can be computed by traversing DAG in a backward-reachability fashion, such that the discovery of the longest distances follows an inverted topological ordering.[10] During the computation, T^* contains those states whose longest distance is established, i.e., those states which have no outgoing transitions left to visit; T contains the maximal currently known distances for those states which have been discovered so far, but still have some outgoing transition left to visit. The computation begins by setting in T^* the distance of states without outgoing edges (that are φ_2-states by construction) to zero,[11] while T will be initially empty. The algorithm continues by picking and removing a longest-distance pair (s, τ) in T^*, adding s to Q if τ is greater than the time bound and iteratively backward visiting its incoming edges. For each discovered predecessor s' of s: if s' is discovered for the first time, then it is inserted together with its current distance in T; if s' was already in T then, if needed, its distance is updated to the current one; in any case, if all outgoing edges from s' have been visited, then we know its longest distance and we store this information in T^*. The procedure stops when T^* becomes empty. At the end Q contains all the states satisfying φ.

Procedure 7 describes the algorithm Sat-EU$_{a,b}(\mathcal{TK}, \varphi)$ for computing the satisfaction set for a formula φ of the form $E\ \varphi_1\ U_I\ \varphi_2$ with finite interval bounds. This procedure is similar to Procedure 5, except that instead of recursively computing the shortest paths from φ_1-states to φ_2-states along $(\varphi_1\ U\ \varphi_2)$-paths in $\mathcal{TK}$, it computes all the durations of such paths within the upper time bound. It first computes the satisfaction set Q_u of the untimed formula $\varphi_u = E\ \varphi_1\ U\ \varphi_2$, and restricts the path duration computation to the graph $G = (\mathsf{Q}_u, \mathsf{TR})$, where TR is the set of transitions between φ_u-states, excluding outgoing transitions from $\neg\varphi_1$-states, since these transitions do not belong to any $(\varphi_1\ U\ \varphi_2)$-path. Then, it recursively computes distances from each φ_1-state to some φ_2-state along the paths in TR. This is achieved by initially putting in the set T of (*state*, *distance*) pairs the φ_2-states in Q_u with distance zero, and then iteratively picking any pair $(s, \tau) \in \mathsf{T}$ and adding to T the set of pairs T_{pre}, of (not already visited) predecessors of s that can reach a φ_2-state through a transition in TR within the upper time bound b. The procedure returns the states having the shortest distance within the interval I, which are stored in Q. Because of the assumption of time-divergence, which guarantees the absence of zero-time loops, the above backward computation of the distances will eventually terminate, since the distance will eventually reach the time bound b.

[9] DAG does not contain states in some $SCC(\varphi_1)$ and there cannot be an SCC in DAG containing some $(\varphi_2 \wedge \neg\varphi_1)$-state, since, when constructing $\mathcal{TK}'$, all outgoing transitions from such states have been removed.

[10] A topological ordering of a directed graph is a linear ordering of its nodes such that for every directed edge (u, v) from node u to node v, u comes before v in the ordering.

[11] The longest distance to these states is zero.

Procedure 6 Sat-EU$_a(\mathcal{TK}, \varphi)$

Input : $\mathcal{TK} = (S, \mathcal{T}, \longrightarrow, L)$,
$\varphi = E\ \varphi_1\ U_I\ \varphi_2$, with $a = inf(I)$ and $sup(I) = \infty$
$\sim = (\text{if } a \in I \text{ then } \geq \text{ else } >)$.

Output : $Sat(\varphi) = \{s \in S \mid \mathcal{TK}, s \models_p \varphi\}$.

$\mathsf{Q}_1 := \mathsf{Sat}(\mathcal{TK}, \varphi_1)$;
$\mathsf{Q}_u := \mathsf{Sat}(\mathcal{TK}, E\ \varphi_1\ U\ \varphi_2)$;
$\mathsf{Q}_{SCC} := \mathsf{Tarjan}(\mathsf{Q}_1, \longrightarrow)$;
$\mathsf{Q} := \mathsf{Q}_u \cap \mathsf{Q}_{SCC}$; // $s \in (\mathsf{Q}_u \cap \mathsf{Q}_{SCC}) \Longrightarrow s \in Sat(\varphi)$
$\mathsf{Q}_{pre} := \{s' \in \mathsf{Q}_1 \setminus \mathsf{Q} \mid \exists s \in \mathsf{Q}, \tau \in \mathcal{T} : (s', \tau, s) \in \longrightarrow\}$;

/* recursively compute $\mathsf{Q} = Sat(E\ \varphi_1\ U$ ("$SCC(\varphi_1)$-state" $\wedge\ E\ \varphi_1\ U\ \varphi_2$)) */
while $\mathsf{Q}_{pre} \neq \emptyset$ **do**
  $\mathsf{Q} := \mathsf{Q} \cup \mathsf{Q}_{pre}$;
  $\mathsf{Q}_{pre} := \{s' \in \mathsf{Q}_1 \setminus \mathsf{Q} \mid \exists s \in \mathsf{Q}, \tau \in \mathcal{T} : (s', \tau, s) \in \longrightarrow\}$;
od

$\mathsf{Q}_{DAG} := \mathsf{Q}_u \setminus \mathsf{Q}$; // the states in $\mathsf{Q}_u \setminus \mathsf{Q}$ induce a *DAG* in $\mathcal{TK}$
$\mathsf{TR}_{DAG} := \{(s, \tau, s') \in \longrightarrow \mid s, s' \in \mathsf{Q}_{DAG} \wedge s \in \mathsf{Q}_1\}$; // $(\varphi_1\ U\ \varphi_2)$-paths in Q_{DAG}

/* Backward traverse $\mathsf{DAG} = (\mathsf{Q}_{DAG}, \mathsf{TR}_{DAG})$ to find longest paths from φ_1-states to φ_2-states;
T^* : {(Q_{DAG}-*state without outgoing edges in* TR_{DAG}, *distance*)}, yet to visit;
T : {(Q_{DAG}-*state*, *current longest distance*)}. */
$\mathsf{T}^* := \{(s, 0) \mid s \in \mathsf{Q}_{DAG} \wedge (\forall (s', \tau, s'') \in \mathsf{TR}_{DAG}.\ s' \neq s)\}$;
$\mathsf{T} := \emptyset$;
while $\mathsf{T}^* \neq \emptyset$ **do**
  let $(s, \tau) \in \mathsf{T}^*$;
  $\mathsf{T}^* := \mathsf{T}^* \setminus \{(s, \tau)\}$;
  if $\tau \sim a$ **then** $\mathsf{Q} := \mathsf{Q} \cup \{s\}$ **fi**;
  foreach $(s', \tau', s) \in \mathsf{TR}_{DAG}$ **do**
    $\mathsf{TR}_{DAG} := \mathsf{TR}_{DAG} \setminus \{(s', \tau', s)\}$;
    if $(\exists \tau'' \in \mathcal{T} : (s', \tau'') \in \mathsf{T})$ **then**
      if $\tau + \tau' > \tau''$ **then**
        $\tau_{max} := \tau + \tau'$; $\mathsf{T} := (\mathsf{T} \setminus \{(s', \tau'')\}) \cup \{(s', \tau_{max})\}$;
      else
        $\tau_{max} := \tau''$;
      fi
    else
      $\tau_{max} := \tau + \tau'$; $\mathsf{T} := \mathsf{T} \cup \{(s', \tau_{max})\}$;
    fi
    /* if s' has no more outgoing transitions in TR_{DAG} */
    if $\{s'' \xrightarrow{\tau''} s''' \in \mathsf{TR}_{DAG} \mid s'' = s'\} = \emptyset$ **then**
      $\mathsf{T}^* := \mathsf{T}^* \cup \{(s', \tau_{max})\}$;
    fi
  endeach
od
return Q;

Procedure 8 describes the algorithm Sat-AU$_0(\mathcal{TK}, \varphi)$ for computing the satisfaction set for a formula φ of the form $A\ \varphi_1\ U_I\ \varphi_2$ with $inf(I) = 0 \notin I$. First the satisfaction set Q_u of the formula $\varphi_u = A\ \varphi_1\ U_{I_0}\ \varphi_2$ where $I_0 = I \cup \{0\}$ is computed; this formula can be reduced to its normal form using Equivalences 4.11, 4.12, 4.13. We restrict the remaining computation to the graph $G = (\mathsf{Q}_u, \mathsf{TR})$, where TR is the set of transitions between φ_u-states, excluding outgoing transitions from $\neg\varphi_1$-states, since these transitions do not belong to any $(\varphi_1\ U\ \varphi_2)$-path. Then the set of states which do not satisfy φ is computed by recursively closing the initial set of states

$$\mathsf{Q} = \{s \in Sat(\varphi_u) \mid \text{“}s \text{ has no outgoing transitions to } \varphi_u\text{-states”}\}$$

with those states that can reach a state in Q in zero-time. Because of the time-divergent assumption, which excludes the presence of zero-time loops in G, this can be done by a backward closure over the zero-time transitions that reach a state in Q. The satisfaction set of φ is hence given by excluding from the set of states satisfying φ_u the states in Q.

4.3. Correctness of the model checking procedure

We first discuss the termination and time complexity of the model checking procedure, which is equivalent to the time complexity of the computation of the satisfaction sets above. By τ_{min} (τ_{max}) we denote the smallest (largest) non-zero

Procedure 7 Sat-EU$_{a,b}(\mathcal{TK}, \varphi)$

Input : $\mathcal{TK} = (S, \mathcal{T}, \longrightarrow, L)$,
$\varphi = E\ \varphi_1\ U_I\ \varphi_2$, with $a = inf(I)$ and $b = sup(I) < \infty$;
$\sim_a = (\text{if } a \in I \text{ then } \geq \text{ else } >)$;
$\sim_b = (\text{if } b \in I \text{ then } \leq \text{ else } <)$.

Output : $Sat(\varphi) = \{s \in S \mid \mathcal{TK}, s \models_p \varphi\}$.

$\mathsf{Q}_1 := \mathsf{Sat}(\mathcal{TK}, \varphi_1)$;
$\mathsf{Q}_2 := \mathsf{Sat}(\mathcal{TK}, \varphi_2)$;
$\mathsf{Q}_u := \mathsf{Sat}(\mathcal{TK}, E\ \varphi_1\ U\ \varphi_2)$;
$\mathsf{T} := \{(s, 0) \mid s \in \mathsf{Q}_2\}$;
$\mathsf{TR} := \{(s, \tau, s') \in \longrightarrow \mid s, s' \in \mathsf{Q}_u \wedge s \in \mathsf{Q}_1\}$; // $(\varphi_1\ U\ \varphi_2)$-paths
$\mathsf{Q} := \emptyset$; $\mathsf{T}_v := \emptyset$;
/* compute distances (up to b) from φ_1-states to a φ_2-state in TR */
while $\mathsf{T} \neq \emptyset$ **do**
  let $(s, \tau) \in \mathsf{T}$;
  $\mathsf{T}_v := \mathsf{T}_v \cup \{(s, r)\}$;
  $\mathsf{T}_{pre} := \{(s', \tau + \tau') \mid (s', \tau', s) \in \mathsf{TR} \wedge (\tau + \tau' \sim_b b)\}$;
  $\mathsf{T} := (\mathsf{T} \setminus \{(s, \tau)\}) \cup \{\mathsf{T}_{pre} \setminus \mathsf{T}_v\}$;
  if $\tau \sim_a a$ **then** $\mathsf{Q} := \mathsf{Q} \cup \{s\}$ **fi**;
od
return Q;

Procedure 8 Sat-AU$_0(\mathcal{TK}, \varphi)$

Input : $\mathcal{TK} = (S, \mathcal{T}, \longrightarrow, L)$,
$\varphi = A\ \varphi_1\ U_I\ \varphi_2$, with $inf(I) = 0 \notin I$.

Output : $Sat(\varphi) = \{s \in S \mid \mathcal{TK}, s \models_p \varphi\}$.

---

let $I_0 = I \cup \{0\}$;
$\mathsf{Q}_1 := \mathsf{Sat}(\mathcal{TK}, \varphi_1)$;
$\mathsf{Q}_u := \mathsf{Sat}(\mathcal{TK}, (A\ \varphi_1\ U_{I_0}\ \varphi_2)!)$;
$\mathsf{TR} := \{(s, \tau, s') \in \longrightarrow \mid s, s' \in \mathsf{Q}_u \wedge s \in \mathsf{Q}_1\}$
$\mathsf{Q} := \{s \in \mathsf{Q}_u \mid \nexists s' \in \mathsf{Q}_u, \tau \in \mathcal{T} : (s, \tau, s') \in \mathsf{TR}\}$;
$\mathsf{Q}_{pre} := \{s' \in \mathsf{Q}_1 \setminus \mathsf{Q} \mid \exists s \in \mathsf{Q} : (s', 0, s) \in \longrightarrow\}$;
/* while loop invariant: $s \in \mathsf{Q} \Longrightarrow \mathcal{TK}, s \not\models_p \varphi$ */
while $\mathsf{Q}_{pre} \neq \emptyset$ **do**
  $\mathsf{Q} := \mathsf{Q} \cup \mathsf{Q}_{pre}$;
  $\mathsf{Q}_{pre} := \{s' \in \mathsf{Q}_1 \setminus \mathsf{Q} \mid \exists s \in \mathsf{Q} : (s', 0, s) \in \longrightarrow\}$;
od
return $\mathsf{Q}_u \setminus \mathsf{Q}$;

transition duration in $\mathcal{TK}$,[12] $|\varphi|$ is the number of subformulas in the TCTL formula φ, τ_φ is the largest finite time constant appearing in the interval bounds of φ, and $\bar{\tau} = max(\tau_{max}, \tau_\varphi)$. Our procedures need to add two time values together or check that a time value is less than another time value, etc. Since we have not fixed the time domain, the complexity of our model checking procedure depends on the complexity of these operations on the time values. For simplicity, we assume in the following complexity results that the time values are natural numbers with the standard operations.

Theorem 4.1. *Given a finite Zeno-free timed Kripke structure $\mathcal{TK} = (S, \mathcal{T}, \longrightarrow, L)$ and a TCTL formula φ in normal form, then* Sat$(\mathcal{TK}, \varphi)$ *terminates and has time complexity $O(|\varphi| \cdot \log(\bar{\tau}) \cdot (|S|^2 + |S| \cdot | \longrightarrow | \cdot \bar{k}))$, where $\bar{k} = \lfloor \tau_\varphi / \tau_{min} \rfloor + 1$.*

Proof. We discuss termination and complexity of each basic procedure, assuming that the satisfaction sets for subformulas have been already computed.

- Sat-EU$(\mathcal{TK}, E\ \varphi_1\ U\ \varphi_2)$: the finiteness of S guarantees that eventually Q_{pre} becomes empty and the fixed-point computation of Q terminates; the procedure has complexity $O(|S| + | \longrightarrow |)$ [38].

[12] If $\mathcal{TK}$ has only time-divergent paths then τ_{min} and τ_{max} are well-defined.

- Sat-EG$(\mathcal{TK}, E\ G\ \varphi_1)$: as in the previous case, the finiteness of S guarantees that eventually $\mathsf{Q}_{\mathsf{pre}}$ becomes empty and the fixed-point computation of Q terminates; Tarjan's algorithm has complexity $O(|S| + |\longrightarrow|)$ and the procedure hence has the same complexity [38].
- Sat-EU$_{\mathsf{b}}(\mathcal{TK}, E\ \varphi_1\ U_I\ \varphi_2)$, with $inf(I) = 0 \in I$, $b = sup(I)$: the finiteness of $\longrightarrow$ guarantees that eventually TR becomes empty and the fixed-point computation of Q terminates; in the worst case the while loop is repeated $|S|$ times, each iteration has complexity $O(|S|)$, and each transition in $\mathcal{TK}$ is visited at most once, thus the complexity of the whole procedure is $O(\log(\bar{\tau}) \cdot (|S|^2 + |\longrightarrow|))$, where $\log(\bar{\tau})$ (i.e., the bit size of $\bar{\tau}$) is needed for the operations on time values.[13]
- Sat-EU$_{\mathsf{a}}(\mathcal{TK}, E\ \varphi_1\ U_I\ \varphi_2)$, with $a = inf(I)$ and $sup(I) = \infty$: again, the finiteness of S guarantees that the first while loop terminates, while the finiteness of $\longrightarrow$ guarantees that the topologically ordered (backward) traversal of the directed acyclic graph terminates; the complexity of the latter is $O(|S| + |\longrightarrow|)$ and hence the procedure has the complexity $O(\log(\bar{\tau}) \cdot (|S| + |\longrightarrow|))$, since adding and comparing time values inside the while loop causes the $\log(\bar{\tau})$ factor.
- Sat-EU$_{\mathsf{a,b}}(\mathcal{TK}, E\ \varphi_1\ U_I\ \varphi_2)$, with $a = inf(I)$ and $b = sup(I) < \infty$: termination is guaranteed by Zeno-freeness, so that eventually the set $\mathsf{T}_{\mathsf{pre}}$ becomes empty for each pair left in T; notice that, conversely from all other basic model checking procedures, this procedure is not insensitive to the size of the time bounds appearing in I, and its complexity depends on how big is the ratio between $b = sup(I)$ and the smallest time value appearing on some transition in $\longrightarrow$ (i.e., τ_{min}). This ratio is $\bar{k}$ in the worst case. Furthermore, the complexity of the procedure depends on the number of iterations in the while loop, which is determined by the number of different pairs that are inserted in the set T, and there are at most $|S| \cdot \bar{k}$ such pairs. Each while loop iteration has complexity $O(\log(\bar{\tau}) \cdot |\longrightarrow|)$, and hence the procedure has complexity $O(\log(\bar{\tau}) \cdot |S| \cdot |\longrightarrow| \cdot \bar{k})$.
- Sat-AU$_0(\mathcal{TK}, A\ \varphi_1\ U_I\ \varphi_2)$, with $inf(I) = 0 \notin I$: the finiteness of S guarantees that the backward closure in the while loop terminates, moreover the complexity of the while loop computation is $O(|S| + |\longrightarrow|)$; the computation of Q_{u} reduces to at most two recursive calls of the kind discussed in the previous items; in particular, if $sup(I) = \infty$ then it leads to the procedure call Sat$(\mathcal{TK}, \neg(E\ G\ \neg\varphi_2) \wedge \neg(E\ \neg\varphi_2\ U\ \neg\varphi_1 \wedge \neg\varphi_2))$, while if $sup(I) < \infty$ it leads to the procedure call Sat$(\mathcal{TK}, \neg(E\ \neg\varphi_2\ U_{\sim b}\ true) \wedge \neg(E\ \neg\varphi_2\ U\ \neg\varphi_1 \wedge \neg\varphi_2))$, with $\sim \in \{>, \geq\}$, which both have complexity $O(|S| + |\longrightarrow|)$.

Therefore, each procedure has time complexity $O(\log(\bar{\tau}) \cdot (|S|^2 + |S| \cdot |\longrightarrow| \cdot \bar{k}))$. Since Sat$(\mathcal{TK}, \varphi)$ recursively computes the satisfaction sets of each subformula of φ with a bottom-up approach, the worst case time complexity of the whole procedure is obtained by multiplying the above complexity by the size of the formula φ, i.e., $O(|\varphi| \cdot \log(\bar{\tau}) \cdot (|S|^2 + |S| \cdot |\longrightarrow| \cdot \bar{k}))$. □

The normalization of the formula φ may increase its size. For example, Equivalence 4.16 reduces a formula with one until modality to a normalized formula with 5 until modalities. In the worst case we have that $|\varphi!|$ is $O(5^{d(\varphi)} \cdot |\varphi|)$, where $d(\varphi)$ is the maximal "depth" of nested until modalities in the formula φ, defined recursively as follows:

$$
\begin{aligned}
d(true) &= 0;\\
d(p) &= 0;\\
d(\neg\varphi_1) &= d(\varphi_1);\\
d(\varphi_1 \wedge \varphi_2) &= max(d(\varphi_1), d(\varphi_2));\\
d(E\ \varphi_1\ U_I\ \varphi_2) &= 1 + max(d(\varphi_1), d(\varphi_2));\\
d(A\ \varphi_1\ U_I\ \varphi_2) &= 1 + max(d(\varphi_1), d(\varphi_2)).
\end{aligned}
$$

Corollary 4.1 follows from the fact that MC$(\mathcal{TK}, s, \varphi)$ consists of a call to the procedure Sat$(\mathcal{TK}, \varphi!)$.

Corollary 4.1. *Given a finite Zeno-free timed Kripke structure $\mathcal{TK} = (S, \mathcal{T}, \longrightarrow, L)$ and a TCTL formula φ, then* MC$(\mathcal{TK}, s, \varphi)$ *terminates and has time complexity*

$$O(5^{d(\varphi)} \cdot |\varphi| \cdot \log(\bar{\tau}) \cdot (|S|^2 + |S| \cdot |\longrightarrow| \cdot \bar{k})).$$

Theorem 4.2, whose proof is given in Appendix B, shows that Sat$(\mathcal{TK}, \varphi)$ computes the satisfaction set of the given (normalized) TCTL formula in the pointwise semantics. It follows that the procedure MC$(\mathcal{TK}, s, \varphi)$ returns $\mathcal{TK}, s \models_p \varphi$.

Theorem 4.2. *Given a finite Zeno-free timed Kripke structure $\mathcal{TK} = (S, \mathcal{T}, \longrightarrow, L)$ and a TCTL formula φ in normal form, then*

$$Sat(\varphi) = \{s \in S \mid \mathcal{TK}, s \models_p \varphi\} = \text{Sat}(\mathcal{TK}, \varphi).$$

[13] This complexity can be reduced if the set of distance pairs is memorized in a data structure that provides an efficient operator for finding the pair with the minimal distance, i.e., by using Fibonacci heaps [39], the complexity could be reduced to $O(\log(\bar{\tau}) \cdot (|S| \log|S| + |\longrightarrow|))$.

Corollary 4.2. *Given a finite Zeno-free timed Kripke structure $\mathcal{TK} = (S, \mathcal{T}, \longrightarrow, L)$, a state $s \in S$, and a TCTL formula φ, then, on termination of* $\mathsf{MC}(\mathcal{TK}, s, \varphi)$*:*

$$\mathcal{TK}, s \models_p \varphi \quad \Longleftrightarrow \quad \mathsf{MC}(\mathcal{TK}, s, \varphi).$$

Proof. As noted in Fact 4.1, model checking φ for state s in $\mathcal{TK}$ is equivalent to model checking the unique normal form $\varphi!$, i.e. $\mathcal{TK}, s \models_p \varphi \iff \mathcal{TK}, s \models_p \varphi!$. By definition of Procedure 1, it follows that $\mathsf{MC}(\mathcal{TK}, s, \varphi) \iff s \in \mathsf{Sat}(\mathcal{TK}, \varphi!)$. By Theorem 4.2, it follows that $\mathsf{Sat}(\mathcal{TK}, \varphi!)$ correctly computes the satisfaction set $Sat(\varphi!) = \{s \in S \mid \mathcal{TK}, s \models_p \varphi!\}$, therefore, we have that:

$$\begin{aligned}
\mathcal{TK}, s \models_p \varphi &\overset{\text{by Proposition 4.1}}{\Longleftrightarrow} \mathcal{TK}, s \models_p \varphi! \\
&\overset{\text{by definition}}{\Longleftrightarrow} s \in Sat(\varphi!) \\
&\overset{\text{by Theorem 4.2}}{\Longleftrightarrow} s \in \mathsf{Sat}(\mathcal{TK}, \varphi!) \\
&\overset{\text{by Procedure 1}}{\Longleftrightarrow} \mathsf{MC}(\mathcal{TK}, s, \varphi). \qquad \square
\end{aligned}$$

As explained in Section 3, pointwise model checking can be used to model check the (normalized) formula φ in $\mathcal{TK}$ in the continuous semantics by model checking the modified formula $\beta(\alpha(\varphi))$ in the *gcd*-transformation $\mathcal{TK}_a^{\gamma_h}$ of $\mathcal{TK}$ in the pointwise semantics. We discuss how the modification $\beta(\alpha(\varphi))$ of φ and the *gcd*-transformation $\mathcal{TK}_a^{\gamma_h}$ of $\mathcal{TK}$ affect the time complexity of the model checking procedure.

We first notice that, in practice, the modification $\beta(\alpha(\varphi))$ of φ does not increase the size of the formula. In particular,

$$\alpha(E\ \varphi_1\ U_I\ \varphi_2) = (\neg p_a \wedge E\ \alpha(\varphi_1)\ U_{I_1}\ \alpha(\varphi_2)) \vee (p_a \wedge E\ \alpha(\varphi_1)\ U_{I_2}\ \alpha(\varphi_2))$$

and, for each state, depending on whether the state is a p_a-state or not, only one of the two until formulas $E\ \alpha(\varphi_1)\ U_{I_1}\ \alpha(\varphi_2)$ and $E\ \alpha(\varphi_1)\ U_{I_2}\ \alpha(\varphi_2)$ needs to be computed. Analogously, for formulas of the kind $A\ \varphi_1\ U_I\ \varphi_2$. A similar reasoning can be done for the transformation $\beta(\varphi)$. In particular,

$$\beta(\varphi_2) \wedge (\neg p_a \vee \beta(\varphi_1)) = (\neg p_a \wedge \beta(\varphi_2)) \vee (p_a \wedge \beta(\varphi_1) \wedge \beta(\varphi_2))$$

and depending on whether the state is a p_a-state or not, only one of the two formulas $\beta(\varphi_1)$ and $\beta(\varphi_1) \wedge \beta(\varphi_2)$ needs to be computed. Computing conjunctions or disjunctions introduced in the transformed formula increases the computation time only by a constant value. Checking whether a state is a p_a-state or not also increases the time complexity by a constant value.

Furthermore, the slightly changed time bounds in the modification $\beta(\alpha(\varphi))$ are irrelevant w.r.t. time complexity. In particular, the modified formula $\beta(\varphi)$ has the same time bounds as φ, and for the modified formula $\alpha(\varphi)$ we may have $\tau_{\alpha(\varphi)} = (k_\varphi - 1) \cdot \gamma_h$, for an integer k_φ such that $\tau_\varphi = k_\varphi \gamma_h$.

Therefore, we can conclude that using the pointwise model checker for model checking a TCTL formula φ in the continuous semantics (which corresponds to the procedure call $\mathsf{MC}(\mathcal{TK}_a^\tau, s, \beta(\alpha(\varphi)))$) has the same time complexity as the procedure call $\mathsf{MC}(\mathcal{TK}_a^\tau, s, \varphi)$, which is shown in Theorem 4.3.

Theorem 4.3. *Let AP be a set of atomic propositions and $\mathcal{TK} = (S, \mathcal{T}, \longrightarrow, L)$ a finite Zeno-free timed Kripke structure over AP whose time domain satisfies the theory $TIME^{gcd}$. Let φ be a TCTL formula over AP, $0 < \tau \in \mathcal{T}$ a time value such that $GCD(\mathcal{TK}, \varphi)$ is a multiple of 2τ and $\mathcal{TK}_a^\tau$ the abstract τ-transformation of $\mathcal{TK}$. Then for each state s of $\mathcal{TK}$ it holds that* $\mathsf{MC}(\mathcal{TK}_a^\tau, s, \varphi)$ *has time complexity*

$$O(5^{d(\varphi)} \cdot |\varphi| \cdot \log(\bar{\tau}) \cdot ((|S| \cdot k_{max})^2 + |S| \cdot |\longrightarrow| \cdot k_{max} \cdot (k_\varphi + 1)))$$

where k_φ is an integer value such that the greatest finite time bound in φ is equal to $k_\varphi \cdot \tau$, and k_{max} is an integer value such that the largest non-zero transition duration in $\longrightarrow$ is equal to $k_{max} \cdot \tau$.

Proof. The smallest non-zero transition duration in $\mathcal{TK}_a^\tau$ is $\tau_{min} = \tau$, and, in general, any finite and non-zero time bound in φ and any finite tick transition duration in $\longrightarrow$ can be expressed as a multiple of τ. In particular, we have that $\tau_{max} = k_{max} \cdot \tau$, the greatest finite time bound in φ is $\tau_\varphi = k_\varphi \cdot \tau$, and the size of the ratio between τ_φ and the smallest time value appearing on some transition in $\longrightarrow$ (i.e., τ) is $\bar{k} = \lfloor k_\varphi \tau / \tau \rfloor + 1 = k_\varphi + 1$. Furthermore, in the worst case, each transition in $\mathcal{TK}$ has duration τ_{max} and in $\mathcal{TK}_a^\tau$, where each transitions is split into k_{max} smaller ones, there are approximately $|S| \cdot k_{max}$ states and $|\longrightarrow| \cdot k_{max}$ transitions. The complexity follows from Corollary 4.1. $\square$

5. Preliminaries on real-time rewrite theories and Real-Time Maude

Real-Time Maude [40] is a language and tool extending Maude [9] to support the formal specification and analysis of real-time systems. The specification formalism is based on *real-time rewrite theories* [2]—an extension of *rewriting logic* [41, 42]—and emphasizes ease and generality of specification. In Real-Time Maude the data types of systems are defined by an

algebraic equational specification, the instantaneous transitions of the system are defined by rewrite rules, and time elapse is defined by tick rewrite rules.

Real-Time Maude specifications are *executable* under reasonable assumptions, and Real-Time Maude provides a range of formal analysis techniques [40], including rewriting for system simulation, reachability analysis, and (untimed) linear temporal logic model checking.

In Sections 5.1, 5.2, and 5.3 we briefly introduce rewriting logic, real-time rewrite theories, and Real-Time Maude. Section 5.4 describes how the bridge crossing benchmark can be modeled in Real-Time Maude, and Section 5.5 defines the class of real-time systems satisfying the requirements for our soundness and completeness result.

5.1. Rewriting logic

Since Real-Time Maude specifications extend Maude specifications, we first recall what is a specification in Maude.

A *membership equational logic* (Mel) [43] *signature* is a triple $\Sigma = (K, \Omega, S)$, with K a set of *kinds*, $\Omega = \{\Omega_{w,k}\}_{(w,k)\in K^* \times K}$ a many-kinded algebraic signature, and $S = \{S_k\}_{k\in K}$ a K-kinded family of disjoint sets of *sorts*. The kind of a sort s is denoted $[s]$. A Mel algebra A contains a set A_k for each kind k, a function $A_f : A_{k_1} \times \cdots \times A_{k_n} \to A_k$ for each operator $f \in \Omega_{k_1 \cdots k_n, k}$, and a subset $A_s \subseteq A_k$ for each sort $s \in S_k$. $\mathbb{T}_{\Sigma,k}$ and $\mathbb{T}_\Sigma(X)_k$ denote, respectively, the set of ground Σ-terms with kind k and of Σ-terms with kind k over the set X of kinded variables.

A Mel theory is a pair (Σ, E), where Σ is a Mel signature, and E is a set of conditional equations of the form $(\forall X)\ t = t'$ **if** *cond* and conditional memberships of the form $(\forall X)\ t : s$ **if** *cond*, for $t, t' \in \mathbb{T}_\Sigma(X)_k$, $s \in S_k$, the latter stating that t is a term of sort s, provided the condition holds; the condition *cond* is a conjunction of individual equations $t_i = u_i$ and individual memberships $w_j : s_j$. In Maude, a conjunct in such a condition may also consist of just a single term t'' of kind `[Bool]`, in which case it abbreviates the equation $t'' = \texttt{true}$. Order-sorted notation $s_1 < s_2$ can be used to abbreviate the conditional membership $(\forall x : [s_1])\ x : s_2$ **if** $x : s_1$. Similarly, an operator declaration $f : s_1 \times \cdots \times s_n \to s$ corresponds to declaring f at the kind level and giving the membership axiom $(\forall x_1 : [s_1], \ldots, x_n : [s_n])\ f(x_1, \ldots, x_n) : s$ **if** $\bigwedge_{1\le i\le n} x_i : s_i$.

A Maude module specifies a *rewrite theory* [42,41] of the form $(\Sigma, E \cup A, R)$, where $(\Sigma, E \cup A)$ is a membership equational logic theory with A a set of equational axioms such as associativity, commutativity, and identity, so that equational deduction is performed *modulo* the axioms A. The theory $(\Sigma, E \cup A)$ specifies the system's state space as an algebraic data type. R is a collection of *labeled conditional rewrite rules* specifying the system's local transitions, each of which has the form[14]

$$[l] : t \longrightarrow t' \ \textbf{if} \ \bigwedge_{i=1}^{n} u_i = v_i \wedge \bigwedge_{j=1}^{m} w_j \longrightarrow w'_j,$$

where l is a *label*. Intuitively, such a rule specifies a *one-step transition* from a substitution instance of t to the corresponding substitution instance of t', *provided* the condition holds; that is, the corresponding substitution instances of the equalities $u_i = v_i$ follow from $E \cup A$, and the substitution instances of the w_j can be rewritten (possibly in several steps) to those of the w'_j. The rules are implicitly universally quantified by the variables appearing in the Σ-terms t, t', u_i, v_i, w_j, and w'_j. A rewrite theory is executable if the equational specification (Σ, E) is (ground) *Church–Rosser* and *terminating* modulo the axioms A, and the rules R are *coherent* [44] with the equations E modulo A. The rules are applied *modulo* the equations $E \cup A$. Operationally, a term is reduced to its E-normal form modulo A before any rewrite rule is applied in Maude. Under the coherence assumption this is a complete strategy to achieve the effect of rewriting in $E \cup A$-equivalence classes.

We briefly summarize the syntax of Maude. Operators are introduced with the `op` keyword: `op` $f : s_1 \ldots s_n$ `->` s. They can have user-definable syntax, with underbars '_' marking the argument positions, and are declared with the sorts of their arguments and the sort of their result. Some operators can have equational *attributes*, such as `assoc`, `comm`, and `id`, stating, for example, that the operator is *associative* and *commutative* and has a certain *identity* element. Such attributes are then used by the Maude engine to match terms *modulo* the declared axioms. An operator can also be declared to be a *constructor* (`ctor`) that defines the carrier of a sort. Unconditional and conditional equations, memberships, and rewrite rules are introduced with the keywords `eq`, `ceq`, `mb`, `cmb`, `rl`, and `crl`, respectively. The mathematical variables in such statements are either explicitly declared with the keywords `var` and `vars`, or can be introduced on the fly in a statement without being declared previously, in which case they have the form *var*:*sort*. An equation $f(t_1, \ldots, t_n) = t$ with the `owise` (for "otherwise") attribute can be applied to a subterm $f(\ldots)$ only if no other equation with left-hand side $f(u_1, \ldots, u_n)$ can be applied. Finally, a comment is preceded by '`***`' or '`---`' and lasts until the end of the line.

5.2. Real-time rewrite theories

Real-time rewrite theories [2] are used to specify real-time systems in rewriting logic. Rewrite rules are divided into *tick rules*, that model time elapse in a system, and *instantaneous rules*, that model instantaneous change. Formally a *real-time rewrite theory* $\mathcal{R}$ is a tuple $(\Sigma, E \cup A, R, \phi, \tau)$ such that

[14] In general, the condition of such rules may not only contain equations $u_i = v_i$ and rewrites $w_j \longrightarrow w'_j$, but also memberships $t_k : s_k$.

- $(\Sigma, E \cup A, R)$ is a rewrite theory, with a sort `System` and a sort `GlobalSystem` with no subsorts or supersorts and with only one operator $\{\_\} : \texttt{System} \to \texttt{GlobalSystem}$ which satisfies no non-trivial equations; furthermore, for any $f : s_1 \dots s_n \to s$ in Σ, the sort `GlobalSystem` does not appear in $s_1 \dots s_n$. The set $\mathbb{T}_{\Sigma/(E \cup A),\texttt{GlobalSystem}}$ of *states* in a real-time rewrite theory $\mathcal{R}$ is defined by the E-equivalence classes of ground terms of sort `GlobalSystem`. The equational theory $(\Sigma, E \cup A)$ contains an equational subtheory $(\Sigma_{\mathsf{TIME}}, E_{\mathsf{TIME}}) \subseteq (\Sigma, E \cup A)$, satisfying the TIME axioms in [2]. The theory TIME [2] defines time abstractly as an ordered commutative monoid $(\mathsf{Time}, 0, +, <)$ with additional operators such as $\dot{-}$ and $\leq$. The theory LTIME in [2] extends the theory TIME with additional axioms that define that the time is linear.[15]
- $\phi : \mathsf{TIME} \to (\Sigma, E)$ is an equational theory morphism which defines how TIME is interpreted in $\mathcal{R}$; we write $0, +, \dots$ instead of $\phi(0), \phi(+), \dots$, and write `Time` for $\phi(\mathsf{Time})$.
- τ is an assignment of a term τ_l of sort `Time` to each rewrite rule of the form $[l] : \{t\} \to \{t'\}$ **if** *cond* in R; if $\tau_l \neq 0$ we call the rule a *tick rule* and write

$$[l] : \{t\} \xrightarrow{\tau_l} \{t'\} \textbf{ if } cond$$

  The term τ_l, denoting the *duration* of the tick rule, may contain variables, including variables that do not occur in t, t', or *cond*. Rewrite rules that are not tick rules are called *instantaneous rules* and take zero time. Since the initial state has the form $\{t\}$, the form of the tick rules ensures that time advances uniformly in the whole system.

We write $t \xrightarrow{r} t'$ when t can be rewritten to t' in time r by a *one-step rewrite* [42], and also write $t \xrightarrow{inst} t'$ for one-step rewrites applying an *instantaneous* rule. A one-step rewrite applying a tick rule (respectively, an instantaneous rule) is called *tick step* (respectively, *instantaneous step*).

5.3. Real-Time Maude

A Real-Time Maude module specifies a real-time rewrite theory. Real-Time Maude is parametric in the time domain, which may be discrete or dense, and defines a supersort `TimeInf` of `Time` which adds the infinity element `INF`. The theory `LTIME-INF` axiomatizes a linear time domain with infinity value `INF`. The theory `GCD-TIME` described in Appendix A extends `LTIME-INF` with new operators `gcd`, `_divides_`, and `half`, and it is consistent with the theory $\mathit{TIME}^{gcd}_{\infty}$ presented in Section 2.1.

Real-Time Maude provides the user with some predefined modules specifying useful time domains. For example, the modules `NAT-TIME-DOMAIN-WITH-INF` and `POSRAT-TIME-DOMAIN-WITH-INF` define the time domain to be, respectively, the natural numbers with infinity and the non-negative rational numbers with infinity. `POSRAT-TIME-DOMAIN-WITH-INF` satisfies the theory TIME^{gcd} with the operators `gcd` and `_divides_` interpreting *gcd* and |, and with "division-by-2" being the interpretation of the operator *half*.

To cover all time instances in a dense time domain, tick rules often have one of the forms

```
crl [tick] : {t} => {t'} in time x if x <= u / cond [nonexec] .   (†),
crl [tick] : {t} => {t'} in time x if cond [nonexec] .            (∗), or
rl [tick] : {t} => {t'} in time x [nonexec] .                     (§)
```

where x is a new variable of sort `Time` which does not occur in $\{t\}$ and which is not initialized in the condition. The (possibly empty) condition *cond* should not further constrain x. This ensures that the tick rules can advance time by *any* amount in rules of the form $(*)$ or (§) and by *any* amount less than or equal to u in rules of the form (†). Rules of these forms are called *time-nondeterministic* and are not directly executable in general, since many choices are possible for instantiating the new variable x (which is why they are specified with the `nonexec` attribute).

Real-Time Maude executes time-nondeterministic tick rules by offering a choice of different *time sampling strategies* [40], so that only *some* moments in the time domain are visited. For example, the *maximal* time sampling strategy advances time by the maximum possible time elapse u in each application of tick rules of the form (†) (unless u equals `INF`), and advances time by a user-given time value r in tick rules having other forms. In the *default* mode, each application of a time-nondeterministic tick rule tries to advance time by a given time value r: if the tick rule has the form (†), then the time advance is the minimum of u and r.

The paper [40] explains the semantics of Real-Time Maude. The real-time rewrite theory $\mathcal{R}^{maxDef(r)}$ denotes the real-time rewrite theory $\mathcal{R}$ where the tick rules are applied according to the maximal time sampling strategy, and where tick steps which advance time by 0 are not applied. $\mathcal{R}^{def(r)}$ denotes $\mathcal{R}$ where the tick rules are applied according to the default time sampling strategy with default time increment r.

In the remainder of this paper $\sigma(r)$ denotes the maximal time sampling strategy with default time r or the default time sampling strategy with time r.

[15] The theory LTIME is equivalent to the theory *TIME* described in Appendix A, with the obvious operator correspondence, e.g., max to *max*, and so on.

Formal analysis in Real-Time Maude Real-Time Maude provides a variety of search and model checking commands for analyzing timed modules by exploring *all* possible behaviors—up to a given number of rewrite steps, duration, or satisfaction of other conditions—that can be nondeterministically reached from the initial state. For example, Real-Time Maude extends Maude's *linear temporal logic model checker* to check whether each behavior (possibly "up to a certain time," as explained in [40]) satisfies a temporal logic formula.

A set AP of (possibly parametric) *atomic propositions* on states $\mathbb{T}_{\Sigma/(E \cup A),\texttt{GlobalSystem}}$ can be defined equationally in an extension $(\Sigma \cup AP, (E \cup A) \cup D) \supseteq (\Sigma, E \cup A)$, which defines a *labeling function* $L_{AP} : \mathbb{T}_{\Sigma/(E \cup A),\texttt{GlobalSystem}} \to \mathcal{P}(AP)$ in the obvious way [9]. In particular, *state propositions* are declared as terms of sort `Prop` and the labeling of states with propositions can be specified by (possibly conditional) equations of the form

```
eq {statePattern  } |= prop = b .   and   ceq {statePattern} |= prop = b if cond .
```

for b a term of sort `Bool`, which defines the state proposition *prop* to hold in all states $\{t\}$ where $\{t\} \models prop$ evaluates to `true`. We denote by $L_{AP'} : \mathbb{T}_{\Sigma/(E \cup A),\texttt{GlobalSystem}} \to \mathcal{P}(AP')$ the labeling function "restricted" to a subset $AP' \subseteq AP$.

5.4. Example: the bridge crossing problem in Real-Time Maude

The bridge crossing problem is a well-known benchmark for real-time systems [45]. Four persons must cross a bridge at night using a lamp. Only one lamp is available and at most two persons can cross at the same time. Each person walks with a different speed. When crossing together, they must walk with the speed of the slower one. Once one or two persons have started to cross the bridge, they must reach the other side, i.e., they can not "change their mind" and go back. The interesting question is: how much time is needed for all four people to get to the other side?

The bridge crossing problem is modeled by the following Real-Time Maude module `BRIDGE`.[16]

```
(tmod BRIDGE is
  including POSRAT-TIME-DOMAIN-WITH-INF .

  sort Side .
  ops left right : -> Side [ctor] .

  op otherSide : Side -> Side .
  eq otherSide(left) = right .
  eq otherSide(right) = left .

  --- The lamp can be either on the left or right side of the bridge.
  sort Lamp .
  op lamp : Side -> Lamp [ctor] .

  --- A person is parametric in the crossing speed and side of the bridge
  sort Person .
  op person : Time Side -> Person [ctor] .

  --- A term of sort System is a multiset of persons and lamps:
  subsorts Person Lamp < System .
  op __ : System System -> System [ctor assoc comm] .

  --- Wrapper to denote states in which some person is crossing
  op crossing : System Time -> System .

  --- constants defining the crossing time of each person
  ops P1 P2 P3 P4 : -> Time .
  eq P1 = 5 .   eq P2 = 10 .   eq P3 = 20 .   eq P4 = 25 .

  vars T T' : Time . var S : Side . vars REST SUBSTATE : System .

  --- a person starts to cross with the lamp
  rl [start1] :
      {person(T,S) lamp(S) REST} =>
      {crossing(person(T,S) lamp(S), T) REST} .

  --- two persons starts to cross with the lamp
```

[16] The specification is available at http://folk.uio.no/leprid/TCTL-RTM/.

```
  rl [start2] :
      {person(T,S) person(T',S) lamp(S) REST} =>
      {crossing(person(T,S) person(T',S) lamp(S), max(T,T')) REST} .

  --- (a) crossing person(s) continue to cross with the lamp
  crl [tick] :
      {crossing(SUBSTATE, T') REST} =>
      {crossing(SUBSTATE, (T' - T)) REST} in time T
     if T <= T' [nonexec] .

  --- the crossing person reaches the other side
  rl [end1] :
      crossing(person(T,S) lamp(S), 0) =>
      person(T,otherSide(S)) lamp(otherSide(S)) .

  --- the two crossing persons reach the other side
  rl [end2] :
      crossing(person(T,S) person(T',S) lamp(S), 0) =>
      person(T,otherSide(S)) person(T',otherSide(S)) lamp(otherSide(S)) .

  --- each person is initialized with the respective crossing time and
  --- initially everything is on the right side of the bridge
  op init : -> System .
  eq init = person(P1,right) person(P2,right)
            person(P3,right) person(P4,right) lamp(right) .

endtm)
```

The time domain is `POSRAT-TIME-DOMAIN-WITH-INF`, i.e., $\mathbb{Q}_{\geq 0}$ extended with the infinity element. The lamp is modeled by a term `lamp`(*side*), where *side* is `left` if the lamp is on the left side of the bridge and is `right` otherwise. A person is modeled as a term `person`(r, *side*), where r is the crossing time and *side* is the position of the person. The wrapper `crossing` is used to wrap the person(s) and lamp which are currently crossing the bridge; its second argument is the time remaining until the person(s) have finished crossing.

The system behavior is modeled by four instantaneous rules and one tick rule. In rule `start1`, a person grabs the lamp, which is on the same side, and immediately starts to cross the bridge. The wrapper `crossing` is initialized with the crossing time of the person. Similarly, in rule `start2`, two persons grab the lamp and start to cross the bridge, and the `crossing` wrapper is initialized with the crossing time of the slower person, i.e., `max(T1,T2)`.

The `tick` rule models time advance while somebody is crossing the bridge, so that the time remaining of the crossing is updated accordingly to the elapsed time.[17] This time-nondeterministic tick rule may advance by *any* amount of time less than or equal to the time when the person(s) finish crossing.

The crossing finishes when the remaining crossing time becomes 0. At this point the crossing person(s) and lamp are on the other side. Rule `end1` and `end2` models the end of the crossing when, respectively, one person and two persons were crossing.

The initial state `init` consists of a lamp and four persons on the right side of the bridge. Each person is initialized with a constant speed equal to `5`, `10`, `20` and `25`, respectively.

5.5. Time-robust real-time rewrite theories and tick-invariant atomic propositions

Tick rules in a real-time rewrite theory typically have the form (†) seen in Section 5.3, which ensures that any moment in time (within time u) can be visited. For a dense time domain, achieving a sound and complete model checking analysis by visiting all times is hopeless, since covering all possible behaviors would require to execute all possible tick rewrite sequences. For example, if time advances from r to r' with $r' > r$ there is an *infinite* set of intermediate times r'' with $r \leq r'' \leq r'$ that will *not* be visited if the clock ticks by the amount $r' - r$.

As explained in Section 5.3, Real-Time Maude deals with this problem by using time sampling strategies to instantiate the new variable x in the tick rules, and to analyze the resulting specification instead of the original one. However, since only a *subset* of all possible behaviors are analyzed in this way, these formal analyses are in general not sound and complete.

Nevertheless, if

1. advancing time by an amount smaller than the maximum time elapse cannot lead to behaviors (deadlocks, other tick possibilities, taking instantaneous transitions, etc.) different from those that can be covered by advancing time by the maximum time elapse; and

[17] Time does not advance if nobody is crossing.

2. if state propositions do not change as a result of performing a non-maximal tick step,

then the maximal time sampling strategy yields sound and complete analyses for *untimed* LTL\{◯} (i.e., LTL without the next operator) properties [5].

Systems satisfying the first of the properties above are called *time-robust*, and atomic propositions that satisfy the second property are *tick-invariant*, in the sense formalized below.

The following definition formalizes the kind of tick steps that are possible when the maximal time sampling strategy is applied, i.e., the so-called *maximal*, *non-maximal* and ∞ tick step [5].

Definition 5.1. A one-step rewrite $t \xrightarrow{r} t'$ using a tick rule and having duration r is:

- a *maximal* tick step if there is no time value $r' > r$ such that $t \xrightarrow{r'} t''$ for some t'';
- an ∞ tick step if for each time value $r' > 0$, there is a tick rewrite step $t \xrightarrow{r'} t''$; and
- a *non-maximal* tick step if there is a maximal tick step $t \xrightarrow{r'} t''$ for some $r' > r$ and t''.

Fact 5.1. *The tick steps in the real-time rewrite theory $\mathcal{R}^{maxDef(r)}$ are either maximal tick steps or tick steps of the default duration r (when sampling the ∞ tick steps in $\mathcal{R}$).*

Time-robustness is formally defined as follows (we refer to [5] for a more thorough explanation).

Definition 5.2. A real-time rewrite theory $\mathcal{R}$ is *time-robust* if the following requirements TR1 to TR6 hold for all ground terms t, t', t'', t''' of sort `GlobalSystem`, and all ground terms r, r', r'' of sort `Time`:

TR1. Each one-step rewrite using a tick rule is either a maximal, a non-maximal, or an ∞ tick step;
TR2. A maximal tick step $t \xrightarrow{r+r'} t''$ and a non-maximal tick step $t \xrightarrow{r} t'$ imply that there exists a t''' such that $t' \xrightarrow{r'} t'''$ is maximal tick step;
TR3. A non-maximal tick step $t \xrightarrow{r} t'$ and a maximal tick step $t' \xrightarrow{r'} t''$ imply that there is a maximal tick step $t \xrightarrow{r+r'} t''$;
TR4. If $t \xrightarrow{r} t'$ is a tick step with $r > 0$, and $t' \xrightarrow{inst} t''$ is an instantaneous one-step rewrite, then $t \xrightarrow{r} t'$ is a *maximal* tick step.
TR5. If $t \xrightarrow{r} t'$ is an ∞ tick step, then $t' \xrightarrow{r'} t''$ is also an ∞ tick step.
TR6. $t = t'$ holds in the underlying equational theory for any 0-time tick step $t \xrightarrow{0} t'$.

Fact 5.2. *Time-robustness of $\mathcal{R}$ ensures that for each sequence of non-maximal tick steps followed by a maximal tick step*

$$t \xrightarrow{r_0} t_1 \xrightarrow{r_1} t_2 \xrightarrow{r_2} \cdots \xrightarrow{r_{k-1}} t_k \xrightarrow{r_k} t'$$

there exists a maximal tick step $t \xrightarrow{r} t'$ in $\mathcal{R}$ such that $\sum_{i=0}^{k} r_i = r$.

Example 5.1. The `BRIDGE` module described in Section 5.4 specifies a time-robust system. Intuitively, the system is time-robust since, once a person has started to cross the bridge, nothing else can happen in the system until the person has reached the other side. That is, by stopping time advance in "mid-crossing", nothing can happen except further advancing time until they reach the other side. In particular:

- TR1 holds because the tick rule has the form (†) as described in Section 5, and, thus, all tick steps are either maximal or non-maximal.
- TR2 holds since, when both a (non-zero) maximal tick step $t \xrightarrow{r+r'} t''$ (i.e., a tick step that, during a crossing, advances time all the way up to the current remaining crossing time $r + r' > 0$, and hence the remaining crossing time in t'' is 0) and a non-maximal tick step $t \xrightarrow{r'} t'$ (i.e., a tick step of duration r such that the remaining crossing time in t' is r') are possible, then, from t' it is possible to advance time by its remaining crossing time $r' = (r + r') - r$, and hence a maximal tick step of duration r' is possible from t' to t''.
- TR3 holds since, given a state t happening during a crossing, if $t \xrightarrow{r} t'$ is a non-maximal tick step of duration r such that the remaining crossing time in t' is r', and $t' \xrightarrow{r'} t''$ is a maximal tick step (i.e., the remaining crossing time in t'' is 0), then it must be the case that the remaining crossing time in t is $r + r'$, and hence a maximal tick step $t \xrightarrow{r+r'} t''$ is possible.
- TR4 holds since instantaneous steps are not possible after non-maximal tick steps (i.e., during a crossing), since after a non-maximal tick step the remaining crossing time is still greater than zero, and hence none of the instantaneous rules are enabled.
- TR5 holds trivially since the form of the tick rule does not allow ∞ tick steps: while in a crossing state, time can advance by at most the remaining crossing time.

- TR6 holds since 0-time tick steps do not change the system state. In particular, a 0-time application of the rule `tick` gives the one-step-rewrite $t \xrightarrow{0} t$, since $r - 0$ is equal to r for each time r.

Example 5.2. Consider a modified version of the `BRIDGE` module described in Section 5.4, where we replace rules `end1` and `end2` by the following rules:

```
rl [end1-any-time] :
    crossing(person(T,S) lamp(S), T'') =>
    person(T,otherSide(S)) lamp(otherSide(S)) .

--- the two crossing persons reach the other side
rl [end2-any-time] :
    crossing(person(T,S) person(T',S) lamp(S), T'') =>
    person(T,otherSide(S)) person(T',otherSide(S)) lamp(otherSide(S)) .
```

where `T''` is a variable of sort Time. That is, crossing the bridge can take any time smaller than or equal to the crossing time of the slowest guy. This modified model is *not* time-robust, since the above instantaneous rules can happen after non-maximal tick steps.

A set of atomic propositions is tick-invariant in $\mathcal{R}$ if the value of the atomic propositions can only be changed by instantaneous steps or maximal tick steps in $\mathcal{R}$.

Definition 5.3. Given a time-robust real-time rewrite theory $\mathcal{R} = (\Sigma, E \cup A, R, \phi, \tau)$ and an extension $(\Sigma \cup AP, (E \cup A) \cup D) \supseteq (\Sigma, E \cup A)$ of $\mathcal{R}$ defining a set of atomic propositions AP and giving rise to a labeling function L_{AP}, $AP' \subseteq AP$ is *tick-invariant* if and only if $L_{AP'}(t) = L_{AP'}(t')$ for each non-maximal or ∞ tick step $t \xrightarrow{r} t'$ in $\mathcal{R}$.

In [5] it is shown that if $\mathcal{R}$ is time-robust and the atomic propositions AP are tick-invariant in $\mathcal{R}$ then model checking an LTL$\backslash\{\bigcirc\}$ formula ϕ over AP in $\mathcal{R}^{maxDef(r)}$ is a sound and complete procedure for model checking the same formula in $\mathcal{R}$.

6. Sound and complete TCTL model checking of real-time rewrite theories

In this section, we show how the soundness and completeness results for TCTL model checking in timed Kripke structures can be applied to real-time rewrite theories. We start by showing how $\mathcal{R}$ defines a timed Kripke structure $\mathcal{TK}(\mathcal{R}, AP)$. Then we show that if the following requirements are satisfied:

- $\mathcal{R}$ is *time-robust*; and
- AP is *tick-invariant* in $\mathcal{R}$;

then $\mathcal{R}^{maxDef(r)}$ and $\mathcal{R}$ are TCTL-equivalent in the continuous semantics:

$$\mathcal{TK}(\mathcal{R}^{maxDef(r)}, AP), t \models_c \varphi \iff \mathcal{TK}(\mathcal{R}, AP), t \models_c \varphi.$$

Combining this result with the soundness and completeness result of Section 3, we obtain that, if the time domain of $\mathcal{R}$ satisfies the theory $TIME^{gcd}$, then model checking the modified TCTL_{cb} formula $\beta(\alpha(\varphi))$ in the pointwise semantics in the *gcd*-transformation of the (Zeno-free) timed Kripke structure $\mathcal{TK}_{t_0}(\mathcal{R}^{maxDef(r)}, AP)$ (restricted to states reachable from a given state t_0 in $\mathcal{R}$) defined by $\mathcal{R}^{maxDef(r)}$ is a sound and complete model checking procedure for model checking φ in the continuous semantics in $\mathcal{R}$, i.e.,

$$\mathcal{TK}_{t_0}(\mathcal{R}^{maxDef(r)}, AP)_a^{\gamma_h}, (t, 0) \models_p \beta(\alpha(\varphi)) \iff \mathcal{TK}(\mathcal{R}, AP), t \models_c \varphi,$$

for each state t reachable in $\mathcal{R}^{maxDef(r)}$ from t_0 in a time-divergent path.

In what follows, we assume that AP is a set of atomic propositions defined in an extension of $\mathcal{R}$ and defines the labeling function L_{AP}.

6.1. Applicability

The class of time-robust systems contains many systems encountered in practice. A typical example of a *time-robust* system are "discrete-event" systems, where each discrete event (i.e., each instantaneous transition) is triggered by the expiration of a timer or by the arrival of a message with a deterministic transmission delay. For example, (the Real-Time Maude models of) Ptolemy II discrete-event systems are time-robust.

The second requirement of *tick-invariance* requires that atomic propositions are not changed by (non-maximal) tick steps. This requirement almost always holds in real applications, since the only values changed by applying tick rules are typically clocks and timers, whose values almost never affect atomic propositions in practice. Indeed, the atomic propositions in our case studies are all tick-invariant.

6.2. Timed Kripke structures defined by real-time rewrite theories

Extending the approach of [9], a real-time rewrite theory $\mathcal{R}$, which also defines a set of atomic propositions AP, naturally defines a timed Kripke structure $\mathcal{TK}(\mathcal{R}, AP)$ as follows:

- Each state in $\mathcal{R}$, i.e., each term in $\mathbb{T}_{\Sigma/(E \cup A),\texttt{GlobalSystem}}$, is also a state in $\mathcal{TK}(\mathcal{R}, AP)$.
- Each one-step rewrite in $\mathcal{R}$ corresponds to a transition in $\mathcal{TK}(\mathcal{R}, AP)$. Furthermore, in order to guarantee that the transition relation of $\mathcal{TK}(\mathcal{R}, AP)$ is total, a zero-time loop transition is added to those states that cannot be further rewritten in $\mathcal{R}$.
- Finally, each state t in $\mathcal{TK}(\mathcal{R}, AP)$ is labeled with the atomic propositions $L(t)$.

Formally, each real-time rewrite theory defines a timed Kripke structure as follows:

Definition 6.1. Given a real-time rewrite theory $\mathcal{R} = (\Sigma, E \cup A, R, \phi, \tau)$, a set of atomic propositions AP and an extension $(\Sigma \cup AP, (E \cup A) \cup D) \supseteq (\Sigma, E \cup A)$ defining the atomic propositions, we define the associated timed Kripke structure as

$$\mathcal{TK}(\mathcal{R}, AP) = (\mathbb{T}_{\Sigma/(E \cup A),\texttt{GlobalSystem}}, (\mathbb{T}_{\Sigma/(E \cup A),\texttt{Time}}, +, 0, \ldots), \longrightarrow, L),$$

where $+, 0, \ldots$ are the interpretations of $+, 0, \ldots$ in the algebra $\mathbb{T}_{\Sigma,E}$,

$$\longrightarrow \subseteq \mathbb{T}_{\Sigma/(E \cup A),\texttt{GlobalSystem}} \times \mathbb{T}_{\Sigma/(E \cup A),\phi(\mathsf{Time})} \times \mathbb{T}_{\Sigma/(E \cup A),\texttt{GlobalSystem}}$$

contains all transitions of the kind $t \xrightarrow{r} t'$ which are also *one-step rewrites* in $\mathcal{R}$ and all transitions of the kind $t \xrightarrow{0} t$ for all those states t that cannot be further rewritten in $\mathcal{R}$, and for $L : \mathbb{T}_{\Sigma/(E \cup A),\texttt{GlobalSystem}} \to \mathcal{P}(AP)$ we have that $p \in L(t)$ if and only if $(E \cup A) \cup D \vdash (t \mid = p) = \texttt{true}$.

The satisfaction relation for TCTL formulas and real-time rewrite theories is given by the TCTL satisfaction relation for timed Kripke structures.

Definition 6.2. Given a TCTL formula φ over AP, we define the satisfaction relation of a real-time rewrite theory $\mathcal{R}$ as follows

$$\mathcal{R}, L_{AP}, t \models_p \varphi \iff \mathcal{TK}(\mathcal{R}, AP), t \models_p \varphi \quad \text{and} \quad \mathcal{R}, L_{AP}, t \models_c \varphi \iff \mathcal{TK}(\mathcal{R}, AP), t \models_c \varphi.$$

We denote by $\mathcal{TK}_{t_0}(\mathcal{R}, AP)$ the timed Kripke structure $\mathcal{TK}(\mathcal{R}, AP)$ restricted to states reachable from t_0.

6.3. Sound and complete TCTL model checking of real-time rewrite theories

In dense time, it would be impossible to directly model check $\mathcal{TK}(\mathcal{R}, AP)$ using our technique based on the *gcd*-transformation. In particular,

- $\mathcal{TK}(\mathcal{R}, AP)$ contains *infinitely many* transitions, since a tick rule in $\mathcal{R}$ corresponds to infinitely many transitions in $\mathcal{TK}(\mathcal{R}, AP)$, each having a different duration; and hence
- the greatest common divisor of this infinitely many durations would be 0.

We are interested in analyzing $\mathcal{R}, L_{AP}, t \models_c \varphi$, since the continuous semantics seems to be the most natural semantics. As in model checking untimed LTL formulas, for dense time it is necessary to model check a "discrete" abstraction of $\mathcal{R}$, such as $\mathcal{R}^{maxDef(r)}$. Although, in general, in the continuous semantics model checking $\mathcal{R}^{maxDef(r)}$ is not equivalent to model checking $\mathcal{R}$, we show that if

- $\mathcal{R}$ is *time-robust*; and
- the set of atomic propositions AP is *tick-invariant* in $\mathcal{R}$

then

$$\mathcal{R}^{maxDef(r)}, L_{AP}, t \models_c \varphi \iff \mathcal{R}, L_{AP}, t \models_c \varphi.$$

In dense time, for each maximal tick step $t \xrightarrow{r} t'$, there are possibly infinitely many of the "splittings" described in the equation of Fact 5.2, i.e.:

$$t \xrightarrow{r_0} t_1 \xrightarrow{r_1} t_2 \xrightarrow{r_2} \cdots \xrightarrow{r_{k-1}} t_k \xrightarrow{r_k} t'$$

such that $\sum_{i=0}^{k} r_i = r$, the tick step $t_k \xrightarrow{r_k} t'$ is maximal and all steps before are non-maximal. However, it is also the case that:

- time-robustness of $\mathcal{R}$ ensures that each maximal tick step $t \xrightarrow{r} t'$ in $\mathcal{R}$ is intuitively "atomic", in the sense that it is not possible to take an instantaneous step from some t_i-state in such a sequence; furthermore
- tick-invariance of the set of atomic propositions AP in $\mathcal{R}$ ensures that all the states t_i in such a sequence satisfy the same atomic propositions in AP as t.

Therefore, in the continuous interpretation of $\mathcal{TK}(\mathcal{R}, AP)$, each such sequence above can be represented by the corresponding maximal tick step. Lemma 6.1, whose proof is given in Appendix B, states that if $\mathcal{R}$ is time-robust and $AP' \subseteq AP$ is tick-invariant in $\mathcal{R}$, then $\mathcal{R}$ and $\mathcal{R}^{maxDef(r)}$ have the same continuous semantics.

Lemma 6.1. *Assume a time-robust real-time rewrite theory $\mathcal{R}$. Let AP be a set of atomic propositions with a* labeling function $L_{AP} : \mathbb{T}_{\Sigma/(E \cup A),\texttt{GlobalSystem}} \to \mathcal{P}(AP)$*, such that a subset of atomic propositions $AP' \subseteq AP$ is tick-invariant in $\mathcal{R}$. Let $r \neq 0$ be a term of sort* `Time`*, and φ a TCTL formula over AP'. Then*

$$\mathcal{R}, L_{AP'}, t \models_c \varphi \iff \mathcal{R}^{maxDef(r)}, L_{AP'}, t \models_c \varphi$$

for each state t in $\mathcal{R}$.

Lemma 6.2, whose proof can also be found in Appendix B, states that, under the same conditions stated in Lemma 6.1, $\mathcal{R}^{maxDef(r)}$ and $\mathcal{R}^{def(r)}$ have the same continuous semantics.

Lemma 6.2. *Let $\mathcal{R}$, $L_{AP'}$ and φ be as in Lemma 6.1, and let $r, r' \neq 0$ be two terms of sort* `Time`*, then*

$$\mathcal{R}^{maxDef(r)}, L_{AP'}, t \models_c \varphi \iff \mathcal{R}^{def(r')}, L_{AP'}, t \models_c \varphi$$

for each state t in $\mathcal{R}$.

Recall that by $\mathcal{R}^{\sigma(r)}$ we denote $\mathcal{R}^{maxDef(r)}$ or $\mathcal{R}^{def(r)}$. Theorem 6.1 is a generalization of Lemma 6.1 that takes into account both the maximal and default time sampling strategies.

Theorem 6.1. *Assume a time-robust real-time rewrite theory $\mathcal{R}$. Let AP be a set of atomic propositions defined in an extension $(\Sigma \cup AP, (E \cup A) \cup D) \supseteq (\Sigma, E \cup A)$, with a* labeling function $L_{AP} : \mathbb{T}_{\Sigma/(E \cup A),\texttt{GlobalSystem}} \to \mathcal{P}(AP)$*, such that $\mathcal{R}$ is tick-invariant with respect to a subset of atomic propositions $AP' \subseteq AP$. Let $r \neq 0$ be a term of sort* `Time`*, and φ a TCTL formula over AP'. Then*

$$\mathcal{R}, L_{AP'}, t \models_c \varphi \iff \mathcal{R}^{\sigma(r)}, L_{AP'}, t \models_c \varphi$$

for each state t in $\mathcal{R}^{\sigma(r)}$.

Proof. If $\mathcal{R}^{\sigma(r)} = \mathcal{R}^{maxDef(r)}$ then the theorem follows trivially by Lemma 6.1. If $\mathcal{R}^{\sigma(r)} = \mathcal{R}^{def(r)}$ then

$$\begin{aligned} \mathcal{R}, L_{AP'}, t \models_c \varphi &\overset{\text{by Lemma 6.1}}{\iff} \mathcal{R}^{maxDef(r)}, L_{AP'}, t \models_c \varphi \\ &\overset{\text{by Lemma 6.2}}{\iff} \mathcal{R}^{def(r)}, L_{AP'}, t \models_c \varphi. \end{aligned}$$ □

Theorems 6.1 and 3.4, give our soundness and completeness result: model checking the TCTL_{cb} formula $\beta(\alpha(\varphi))$ in the *gcd*-transformation of the timed Kripke structure defined by $\mathcal{R}^{\sigma(r)}$ in the pointwise semantics is equivalent to model checking φ in a time-robust and tick-invariant real-time rewrite theory $\mathcal{R}$ (whose time domain satisfies the theory $TIME^{gcd}$) in the continuous semantics, if the timed Kripke structure defined by $\mathcal{R}^{\sigma(r)}$ (when restricted to states reachable from a given state t_0 of $\mathcal{R}$) is Zeno-free and the computed greatest common divisor is a defined value.

Theorem 6.2. *Assume a time-robust real-time rewrite theory $\mathcal{R}$ whose time domain satisfies the theory $TIME^{gcd}$. Let AP and AP' be as in Theorem 6.1, and let t_0 be a state of $\mathcal{R}$, $r \neq 0$ be a term of sort* `Time`*, and φ a TCTL formula over AP'. If $\mathcal{TK}_{t_0}(\mathcal{R}^{\sigma(r)}, AP')$ is Zeno-free and $2 \cdot \gamma_h = GCD(\mathcal{TK}_{t_0}(\mathcal{R}^{\sigma(r)}, AP'), \varphi)$ is a defined value, then*

$$\mathcal{R}, L_{AP'}, t \models_c \varphi \iff \mathcal{TK}_{t_0}(\mathcal{R}^{\sigma(r)}, AP')_a^{\gamma_h}, (t, 0) \models_p \beta(\alpha(\varphi))$$

for each state t reachable in $\mathcal{R}^{\sigma(r)}$ from t_0 in a time-divergent path, where $(t, 0)$ is the state in $\mathcal{TK}_{t_0}(\mathcal{R}^{\sigma(r)}, AP')_a^{\gamma_h}$ corresponding to t.

Proof.

$$\begin{aligned} \mathcal{R}, L_{AP'}, t \models_c \varphi &\overset{\text{by Theorem 6.1}}{\iff} \mathcal{R}^{\sigma(r)}, L_{AP'}, t \models_c \varphi \\ &\overset{\text{by definition}}{\iff} \mathcal{TK}_{t_0}(\mathcal{R}^{\sigma(r)}, AP'), t \models_c \varphi \\ &\overset{\text{by Theorem 3.4}}{\iff} \mathcal{TK}_{t_0}(\mathcal{R}^{\sigma(r)}, AP')_a^{\gamma_h}, (t, 0) \models_p \beta(\alpha(\varphi)). \end{aligned}$$ □

6.4. Sound and complete model checking for discrete time

When the time domain is the natural numbers $\mathbb{N}$, a sound and complete model checking for a TCTL formula φ can be achieved by advancing time by 1 in each tick step in $\mathcal{R}$. Theorem 6.3, whose proof is given in Appendix B, states that model checking the theory $\mathcal{R}^{def(1)}$ in the pointwise semantics is a sound and complete model checking procedure for $\mathcal{R}$ in the continuous semantics.

Theorem 6.3. *Assume a real-time rewrite theory $\mathcal{R}$ with time domain* NAT − TIME − DOMAIN − WITH − INF. *Let AP be a set of atomic propositions with labeling function $L_{AP} : \mathbb{T}_{\Sigma/(E \cup A),\texttt{GlobalSystem}} \to \mathcal{P}(AP)$. Let $r \neq 0$ be a term of sort* Time, *and φ a TCTL formula over AP. Then*

$$\mathcal{R}, L_{AP}, t \models_c \varphi \quad \Longleftrightarrow \quad \mathcal{R}^{def(1)}, L_{AP}, t \models_p \varphi$$

for each state t in $\mathcal{R}$.

A similar argument as the one in Section 3.4 for timed Kripke structures with time domain $\mathbb{N}$ can be made for real-time rewrite theories. When $\mathcal{R}$ is time-robust, the atomic propositions AP are tick-invariant, and the computed greatest common divisor $GCD(\mathcal{TK}_{t_0}(\mathcal{R}^{maxDef(r)}, AP), \varphi)$ is greater than 1, we can multiply each finite constant appearing in the timed Kripke structure $\mathcal{TK}_{t_0}(\mathcal{R}^{maxDef(r)}, AP)$ and in the formula φ by 2, so that the computed greatest common divisor becomes even.

Fact 6.1. *Let $\mathcal{R}$ be a time-robust real-time rewrite theory with time domain* NAT − TIME − DOMAIN − WITH − INF, *and let $AP' \subseteq AP$ be a subset of tick-invariant atomic propositions, t_0 a state of $\mathcal{R}$, $r \neq 0$ a term of sort* Time, *and φ a TCTL formula over AP'. If $\mathcal{TK}_{t_0}(\mathcal{R}^{\sigma(r)}, AP')$ is Zeno-free and $2 \cdot \gamma_h = GCD(\mathcal{TK}_{t_0}(\mathcal{R}^{\sigma(r)}, AP'), \varphi) > 1$ is a defined value, then*

$$\mathcal{R}, L_{AP'}, t \models_c \varphi \quad \Longleftrightarrow \quad (2 \cdot \mathcal{TK}_{t_0}(\mathcal{R}^{\sigma(r)}, AP'))^{\gamma_h}_a, (t, 0) \models_p 2 \cdot \beta(\alpha(\varphi)),$$

for all states t reachable in $\mathcal{R}^{\sigma(r)}$ from t_0 in a time-divergent path, where $2 \cdot \mathcal{TK}_{t_0}(\mathcal{R}^{\sigma(r)}, AP')$ is obtained from $\mathcal{TK}_{t_0}(\mathcal{R}^{\sigma(r)}, AP')$ by multiplying each finite transition duration by 2, and $2 \cdot \beta(\alpha(\varphi))$ is obtained from $\beta(\alpha(\varphi))$ by multiplying each finite time value appearing in some interval bound by 2.

7. A TCTL model checker for Real-Time Maude

This section describes our TCTL model checker for Real-Time Maude. The model checker is implemented in Maude and analyzes a formula φ in the *pointwise semantics*.

7.1. Overview

Our TCTL model checker for Real-Time Maude has three main components:

1. the automatic generation of the timed Kripke structure defined by a given Real-Time Maude specification;
2. the implementation of the explicit-state model checking procedure for checking a TCTL formula over timed Kripke structures in the *pointwise semantics* described in Section 4; and
3. the *gcd*-transformation described in Section 3, and the related β and α transformations.

As shown in Section 6, *pointwise* model checking the *gcd*-transformation of the timed Kripke structure defined by the theory $\mathcal{R}^{maxDef(r)}$ provides a sound and complete model checking procedure for the *continuous* semantics of $\mathcal{R}$, for large classes of real-time systems and for the *whole* TCTL logic. In particular, we have

$$\mathcal{R}, L_{AP}, t \models_c \varphi \quad \Longleftrightarrow \quad \mathcal{TK}_{t_0}(\mathcal{R}^{maxDef(r)}, AP)^{\gamma_h}_a, (t, 0) \models_p \beta(\alpha(\varphi))$$

under the following assumptions:

- applying a non-maximal tick rule does not change the valuation of a state proposition;
- instantaneous rewrite rules can only be applied after *maximal* tick steps or after applying an instantaneous rule;
- the state space reachable in $\mathcal{R}^{maxDef(r)}$ from the initial state t_0 is finite;
- $\mathcal{TK}_{t_0}(\mathcal{R}^{maxDef(r)}, AP)$ is Zeno-free, the time domain in $\mathcal{R}$ satisfies the theory $TIME^{gcd}$ and γ_h is a defined value.

At the moment, the model checker returns a yes/no answer and does not generate counterexamples/witnesses. This is in line with current TCTL model checkers, which at most provide counterexamples for very limited subsets of TCTL [46–48]. Whereas counterexamples for linear temporal logics consist of a single path that does not satisfy the formula, generating counterexamples in branching time logics is in general much trickier (see, e.g. [49,50]). For example, a counterexample to the validity of the formula $E\ F\ p$ would be the entire computation tree from the initial state. The situation gets even more complicated in nested formulas containing both path quantifiers E and A. We are currently working on providing more useful results (with both counterexamples and witnesses) than "yes" and "no."

7.2. Timed Kripke structure generation

Model checking $\mathcal{R}$ in the pointwise semantics is a decidable problem when generating the timed Kripke structure defined by a real-time rewrite theory $\mathcal{R}$ is a finite procedure. Fact 7.1 states that model checking the timed Kripke structure $\mathcal{TK}$ defined by a real-time rewrite theory in the pointwise semantics is a decidable problem if $\mathcal{TK}$ is finite and the labeling function is computable; i.e., the atomic propositions can be evaluated in the extended equational specification of $\mathcal{R}$ in finite time.

Fact 7.1. *Given a real-time rewrite theory $\mathcal{R} = (\Sigma, E \cup A, R, \phi, \tau)$, a set of atomic propositions AP and an extension $(\Sigma \cup AP, (E \cup A) \cup D) \supseteq (\Sigma, E \cup A)$ defining the satisfaction of AP, the model checking problem $\mathcal{TK}(\mathcal{R}, AP), t \models_p \varphi$ is decidable if*

1. *the equational specification $(\Sigma \cup AP, E \cup D)$ is ground Church–Rosser and terminating modulo the axioms A,*
2. *the rules R in the real-time rewrite theory $\mathcal{R}$ are* coherent *w.r.t. $E \cup D$ modulo the axioms A,*
3. *the set of states reachable from t in the rewrite theory $\mathcal{R}$ is finite, and*
4. *given a pair of reachable states t_1 and t_2, the number of one-step rewrites of the kind $t_1 \xrightarrow{r} t_2$ in $\mathcal{R}$ is* finite.

The first two conditions guarantee the executability of $\mathcal{R}$ and the computability of the atomic propositions in finite time. Then, if the timed Kripke structure $\mathcal{TK}_{t_0}(\mathcal{R}, AP)$ is finite, it can be constructed in finite time, and this implies that the model checking problem is decidable.

7.3. Implementation of the TCTL model checker

Our model checker assumes that all behaviors starting from the initial state are *time-divergent* w.r.t. the selected time sampling strategy. The user is provided with a command that automatically verifies that such requirement holds by checking for the absence of non-trivial SCCs in the timed Kripke structure without its tick transitions. The current implementation of the model checker assumes that the time domain is either `NAT-TIME-DOMAIN-WITH-INF` or `POSRAT-TIME-DOMAIN-WITH-INF`, and provides the user with two possible model checking strategies:

(i) The *basic strategy*, which performs the model checking on the model obtained by applying the user-defined time sampling strategy on the original model. That is, it performs the model checking for the satisfaction problem $\mathcal{R}^\sigma$, $L_{AP}, t_0 \models_p \varphi$ by computing $\mathcal{TK}_{t_0}(\mathcal{R}^\sigma, AP), t_0 \models_p \varphi!$.
(ii) The *gcd strategy*, which extends the selected *time sampling strategy*[18] $\sigma(r)$ with the *gcd*-transformation to perform the model checking for the satisfaction problem $\mathcal{R}, L_{AP}, t_0 \models_c \varphi$, by computing $\mathcal{TK}_{t_0}(\mathcal{R}^{\sigma(r)}, AP)^{\gamma_h}_a, t_0 \models_p \beta(\alpha(\varphi))!$.

Soundness and completeness provided by the *gcd* strategy comes at the cost of a larger state space due to the application of the *gcd*-transformation. When the *gcd* strategy makes model checking infeasible, the user can still perform model checking with the faster basic strategy, which does not increase the state space and can still be very useful to discover potential bugs, as illustrated in the case studies that will be presented in Section 8. In particular, for the universal fragment TACTL, in the pointwise semantics, if a counter-example exists in the model checking of $\mathcal{R}^\sigma$, then this is also a counter-example in $\mathcal{R}$; that is, for φ_A a TACTL formula, we have

$$\mathcal{R}^\sigma, L_{AP}, s \not\models_p \varphi_A \quad \Longrightarrow \quad \mathcal{R}, L_{AP}, s \not\models_p \varphi_A.$$

However, if a counter-example does not exist in $\mathcal{R}^\sigma$, then this does not exclude the existence of a counter-example in $\mathcal{R}$. Conversely, in the existential fragment, if the given state satisfies the TECTL formula in $\mathcal{R}^\sigma$ in the pointwise semantics, meaning that there exists a path π satisfying the given formula, then this holds also for $\mathcal{R}$, since π is also a path in $\mathcal{R}$; that is, for φ_E a TECTL formula, we have

$$\mathcal{R}^\sigma, L_{AP}, s \models_p \varphi_E \quad \Longrightarrow \quad \mathcal{R}, L_{AP}, s \models_p \varphi_E.$$

Our model checker is implemented in Maude, making extensive use of Maude's meta-programming capabilities. The model checker first constructs the timed Kripke structure, according to the selected model checking strategy, and reduces the given formula φ to its normal form $\varphi!$. The (normalized) formula is further simplified by using the equivalences in Table 3.

Since the meta-representation of the states can be fairly large, performing the model checking procedure described in Section 4 on the generated timed Kripke structure is inefficient. In our current implementation, we therefore assign a unique natural number to each (meta-represented) state in the generated timed Kripke structure, and construct a more compact timed Kripke structure, where all the occurrences of these states are replaced by their respective identifiers. We then perform the recursive computation of the satisfaction set of φ on this compact representation. This optimization led

[18] The *gcd* strategy can be used when either the maximal time sampling strategy or the default one is selected, however selecting the maximal time sampling strategy gives a smaller state space and hence faster model checking.

Table 3
Equivalences for the simplification of a TCTL formula.

7.1.	$\neg true = false$	7.14.	$E\ G\ false = false$
7.2.	$\neg false = true$	7.15.	$E\ G\ true = true$
7.3.	$\neg(\varphi_1 \vee \varphi_2) = \neg\varphi_1 \wedge \neg\varphi_2$	7.16.	$E\ \varphi\ U_I\ false = false$
7.4.	$\neg(\varphi_1 \wedge \varphi_2) = \neg\varphi_1 \vee \neg\varphi_2$	7.17.	$E\ false\ U_I\ \varphi = false$, if $0 \notin I$
7.5.	$true \wedge \varphi = \varphi$	7.18.	$E\ false\ U_I\ \varphi = \varphi$, if $0 \in I$
7.6.	$\varphi \wedge true = \varphi$	7.19.	$E\ \varphi\ U_I\ true = true$, if $0 \in I$
7.7.	$false \wedge \varphi = false$	7.20.	$E\ true\ U_I\ true = true$, if $sup(I) = \infty$
7.8.	$\varphi \wedge false = false$	7.21.	$A\ \varphi\ U_I\ false = false$
7.9.	$true \vee \varphi = true$	7.22.	$A\ false\ U_I\ \varphi = false$, if $0 \notin I$
7.10.	$\varphi \vee true = true$	7.23.	$A\ false\ U_I\ \varphi = \varphi$, if $0 \in I$
7.11.	$false \vee \varphi = \varphi$	7.24.	$A\ \varphi\ U_I\ true = true$, if $0 \in I$
7.12.	$\varphi \vee false = \varphi$	7.25.	$A\ true\ U_I\ true = true$, if $sup(I) = \infty$
7.13.	$\neg\neg\varphi = \varphi$		

Table 4
Syntax of a TCTL formula in Real-Time Maude for p a term of sort `Prop`, φ a term of sort `TCTLFormula`, c a term of sort `Comparator`, r a term of sort `Time`, r' a term of sort `TimeInf`, i a term of sort `Interval`, and e a term of sort `IntervalEnd`.

φ ::=	`tt` | `ff` | p | `not` φ | φ `and` φ | φ `or` φ | φ `implies` φ | φ `iff` φ |
	`E` φ `U` φ | `A` φ `U` φ | `EG` φ | `AG` φ | `EF` φ | `AF` φ |
	`E` φ `U[`c `than` r`]` φ | `A` φ `U[`c `than` r`]` φ | `EG[`c `than` r`]` φ | `AG[`c `than` r`]` φ |
	`EF[`c `than` r`]` φ | `AF[`c `than` r`]` φ |
	`E` φ `U[`i`]` φ | `A` φ `U[`i`]` φ | `EG[`i`]` φ | `AG[`i`]` φ | `EF[`i`]` φ | `AF[`i`]` φ |
c ::=	`<` | `<=` | `>` | `>=`
i ::=	e r, r' e
e ::=	`c` | `o`

to a large performance improvement and made it feasible to apply our model checker to a number of case studies, whereas working directly on meta-represented terms made model checking infeasible even for simple case studies. For example, before this optimization we were not able to model check the Ptolemy II models in Section 8 in reasonable time.

When the *gcd* strategy is selected, the timed Kripke structure is refined by "splitting" the tick transitions into smaller ones of duration equal to half the computed greatest common divisor.[19] As seen in Section 3, the new intermediate states satisfy the same atomic propositions holding at the beginning of the tick transition, and hence such states are labeled with these atomic propositions.

Finally, the satisfaction sets of each subformula are recursively computed as described in Section 4.

The flexibility of the Maude meta-level allowed us to implement the model checker reasonably quickly and easily. However, the convenience of operating at the meta-level comes at a certain cost in terms of computational efficiency, even with our optimizations. Therefore, the current Real-Time Maude model checker should be regarded as a *prototype* of a C++ implementation that should be implemented in the future.

7.4. Using the model checker

The user needs to include the module `TCTL-MODEL-CHECKER` into the Real-Time Maude specification in order to run the TCTL model checking commands. This module also defines TCTL formulas as terms of sort `TCTLFormula`. Table 4 contains the syntax of TCTL formulas in Real-Time Maude. The syntax of TCTL formulas is fairly intuitive, e.g. $E\ \mathtt{p1}\ U_{\leq 5}\ (\neg \mathtt{p2} \wedge A\ G\ (E\ F_{(7,8]}\ \mathtt{p3}))$ is written

```
E p1 U[<= than 5](not p2 and AG (EF[o 7, 8 c] p3))
```

where a time interval is a term of the kind $[e_1\ r,\ r'\ e_2]$, where e_1 (respectively e_2) is `o` if the time interval is left (respectively right) open, and is `c` if it is closed, `p1`, `p2` and `p3` are terms of sort `Prop`, and r is a term of sort `Time` and r' is a term of sort `TimeInf`.

The user can check whether the generated timed Kripke structure (when finite) is Zeno-free by running the command:

[19] The time values on the constructed timed Kripke structure are defined as terms of the Maude built-in sort `RAT`, and hence half of the greatest common divisor can be computed also when the time values in $\mathcal{R}$ are in the discrete time domain `NAT-TIME-DOMAIN-WITH-INF`.

```
(tks-zenofree t .)
```

for t the initial state. The command returns informations about the generated timed Kripke structure (e.g., the number of states and the number of transitions) and whether this timed Kripke structure is Zeno-free.

The model checker provides the user with two TCTL model checking commands, corresponding respectively to the basic and the *gcd* strategy, with syntax

```
(mc-tctl t |= φ .)     and     (mc-tctl-gcd t |= φ .)
```

for t the initial state and φ a TCTL formula.

Our model checker is available at http://folk.uio.no/leprid/TCTL-RTM/ together with the related papers, and the specifications and analysis commands of the case studies and the performance comparison in this paper.

7.5. Example: model checking the bridge crossing model

To model check the `BRIDGE` module modeling the bridge crossing system, we specify an extended module `BRIDGE-MC` containing the state proposition `safe`, which is satisfied by those states where all four persons are on the safe (`left`) side, as follows.

```
(tmod BRIDGE-MC is
  including BRIDGE .
  including TCTL-MODEL-CHECKER .

  op safe : -> Prop .
  eq {person(T:Time, right) REST:System} |= safe = false .
  eq {crossing(SUBSTATE:System, T:Time) REST:System} |= safe = false .
  eq {S:System} |= safe = true [owise] .
endtm)
```

The module `BRIDGE-MC` includes the module `TCTL-MODEL-CHECKER`, which includes the predefined Maude module `MODEL-CHECKER` where the sort `Prop` is defined.

Before performing any analysis, we select the time sampling strategy to be the maximal one:

```
Maude> (set tick max def 5 .)
```

Intuitively, all behaviors in this model are time-divergent since the system does not have a terminal state. That is, the persons will cross the bridge back-and-forth forever. The command

```
Maude> (tks-zenofree {init} .)

Number of States: 254
Number of Transitions: 336
Zeno-free: yes
```

returns that the specified system (restricted to the state space reachable from the initial state) is Zeno-free under the maximal time sampling strategy.

In Section 5, we showed that the `BRIDGE` module described in Section 5.4 specifies a time-robust system. Furthermore, the state proposition `safe` is tick-invariant, since its validity can be changed only by instantaneous rules. Therefore, using the maximal time sampling strategy combined with the *gcd* strategy provides a sound and complete analysis with dense time domain.

From the initial state, the shortest crossing time is 60. In this model of the bridge crossing problem, in both the *pointwise* and *continuous* semantics, using the maximal time sampling, from *any* possible state, it is possible to reach the safe state in time 110, since, in the worst case, once the slowest guy starts to cross alone to the safe side with the lamp, it will take $25 + 25$ time units for this guy to go back to the other side.

We can verify that it takes at most time 110 to reach a safe state from *any* possible state with the following command:

```
Maude> (mc-tctl-gcd {init} |=  AG EF[<= than 110] safe .)

Property satisfied
```

8. Case studies

In this section we present the TCTL analysis of three existing case studies that satisfy the requirements for having a sound and complete analysis when using the *gcd* strategy. In particular, each of them uses `POSRAT-TIME-DOMAIN-WITH-INF`, has time-divergent behaviors, and the specification is both time-robust and tick-invariant with regard to the atomic propositions used in the formulas as explained in what follows.

The case study presented in Section 8.1 analyzes a network of medical devices and it shows how we can apply the basic strategy of our TCTL model checker to discover potential flaws in systems whose *gcd*-transformation would have a state space that is too large to be checked in reasonable time.[20] The case studies presented in Section 8.2 analyze two Ptolemy II discrete-event (DE) models.

The analysis has been performed on a 1.87 GHz Intel® Xeon® CPU with 128 GB of RAM.

8.1. A network of medical devices

We have applied our TCTL model checker on a Real-Time Maude model of an interlock protocol for a small network of medical devices, integrating an X-ray machine, a ventilator machine, and a controller. The example was proposed by Lui Sha, and the Real-Time Maude model is explained in detail in [51].

The ventilator machine helps a sedated patient to breathe during surgery. An X-ray can be taken during the surgery. To allow an X-ray to be taken without blurring the picture, the ventilator must be turned off when taking an X-ray. The X-ray must be taken and then the ventilation machine must be restarted. Furthermore, the ventilation machine should not be stopped too often.

The 4 components (user, controller, X-ray device, and ventilation machine) communicate by message passing. The user pushes a button to take an X-ray. Whenever the user wants to take an X-ray, the controller checks if it is safe to take the X-ray, and if it is safe, then the controller sends a message to the ventilation machine to pause for a short time, and sends a message to the X-ray device to take an X-ray at a given time. The model also addresses *nondeterministic* message delays (within a time interval) and clock *drifts*.

The specification used here is slightly different from the one in [52]. In order to make the reachable state space finite, we now reset the controller's clock each time the controller initiates a pause of the ventilation machine (in the previous specification, this clock would just count the time elapsed, making the reachable state space infinite). One time unit in the specification corresponds to one millisecond in the case study.

The network of medical devices model addresses nondeterministic message delays, which in principle would make the system non-time-robust, since the message delivery could happen nondeterministically at any time between the minimal and maximal message delay. However, in [51] a constant time `INTERVAL` is specified, such that each `INTERVAL` time units a nondeterministic choice is made between delivering a message or further delaying. Therefore, any event (included the message delivery) takes place when some timer expires, and hence, the system is time-robust and can be analyzed by using the *maximal* time sampling strategy to advance time until the next timer expires.

The main combined property that the system must satisfy is that

1. the ventilation machine should not pause for more than two seconds at a time, and
2. the ventilation machine must be breathing for at least 10 minutes between two pauses.

These requirements can be expressed by the TACTL formula

$$A\,G\,(\text{“machine is pausing”} \longrightarrow (A\,F_{\leq 2\ \text{seconds}}\,(A\,G_{\leq 10\ \text{minutes}}\text{“machine is breathing”}))$$

In order to analyze this property, we first define two state propositions, `isPausing` and `isBreathing`, in the expected way. These properties are obviously independent of the values of timers, and are therefore tick-invariant. The property is analyzed using the following command in Real-Time Maude:

```
Maude> (mc-tctl initState |= AG(isPausing implies
           (AF[<= than 2000](AG[<= than 600000] isBreathing))) .)

Property not satisfied
```

In about 36 seconds, Real-Time Maude displays that the property is not satisfied, when `INTERVAL` is set to 25 ms and the maximal message delay is 50 ms. The property is not satisfied because the ventilation machine may indeed pause for 2.22 seconds, due to clock drifts, as explained in [51].

Since the analyzed property is in the universal fragment TACTL, when the model checker reports that the property is violated, then we can conclude that the property is also violated in the original theory $\mathcal{R}$ (in the pointwise semantics).

[20] The clock drifts and nondeterministic message delays are the main cause for the large state space of the *gcd*-transformation of this system.

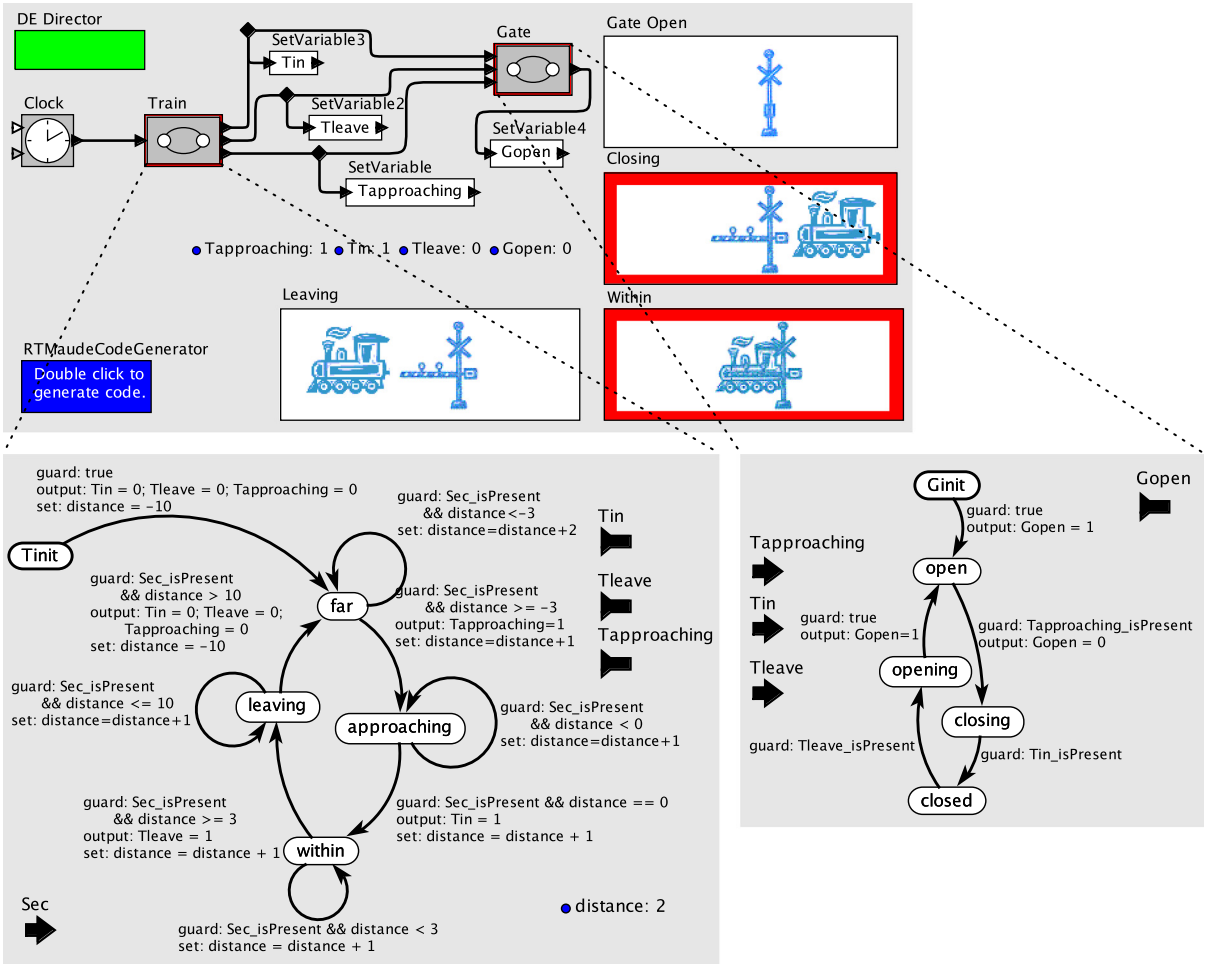


Fig. 6. A Ptolemy II DE model of the railroad crossing benchmark.

The basic strategy is particularly useful for such a system, where the tick step durations under the maximal time sampling strategy can easily vary from an order of 10 to an order of 10^5, and, even if time-robustness and tick-invariance are satisfied, model checking using the *gcd* strategy is infeasible.

8.2. Ptolemy II discrete event models

Real-Time Maude provides a formal analysis tool for a set of modeling languages for embedded systems, including Ptolemy II DE models [18,4]. Ptolemy II [53] is a well-established modeling and simulation tool used in industry that provides a powerful yet intuitive graphical modeling language. Our model checker has been integrated into Ptolemy II by Kyungmin Bae, so that we can now model check TCTL properties of Ptolemy II DE models *from within Ptolemy*.[21] We analyze the existing Ptolemy II DE models of the *railroad crossing* benchmark and the *fault-tolerant traffic light* system. Our model checking has uncovered a previously unknown flaw in the traffic light model.

All Ptolemy II DE models are time-robust, since time advances until the time when the first event(s) in the event queue should fire.

8.2.1. Railroad crossing

Fig. 6 shows the Ptolemy II model of the well-known *railroad crossing benchmark*. In this model, a train approaches a railroad crossing, and a gate should be lowered when a train is in the intersection. The model contains two finite state machines: one for the train and one for the gate. One transition is taken in each time unit in the state machine `Train`. We refer to [4] for a thorough explanation of the model.

[21] Real-Time Maude verification commands can be entered into the dialog box that pops up when the button "Double click to generate code" in Fig. 7 is clicked.

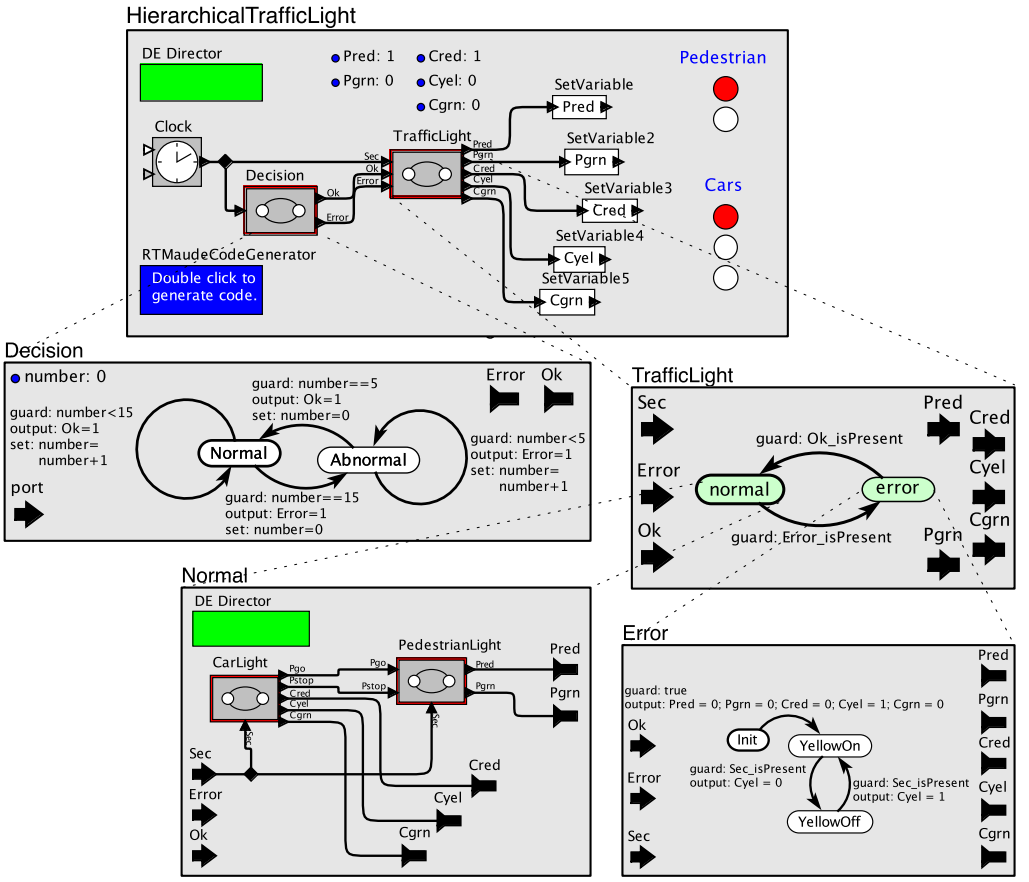


Fig. 7. A fault-tolerant traffic light system in Ptolemy II.

The Real-Time Maude formal analysis back-end for Ptolemy II DE models provides an intuitive syntax for specifying state propositions, so that e.g. the state proposition "the train is in state approaching" is written

```
'RailroadSystem . 'Train @ 'approaching
```

Such state propositions are trivially tick-invariant because their validity can be changed only by instantaneous transitions. Therefore, the following analyses are sound and complete for dense time.

One important property that the system should satisfy is that the gate should open within a reasonable time (here 11 time units) after being lowered, which we can check in the continuous semantics with the *gcd* strategy by running the following Real-Time Maude command:

```
Maude> (mc-tctl-gcd {init} |=
          AG(('RailroadSystem . 'Gate @ 'closed) implies
             AF[<= than 11] ('RailroadSystem . 'Gate @ 'open)) .)

Property satisfied
```

Our model checker confirms (in about 1 second) that the property is satisfied for 11 time units (but not for strictly less than 11 time units).

8.2.2. *A fault-tolerant traffic light system*

Figs. 7 and 8 show a Ptolemy II model of a fault-tolerant traffic light system at a pedestrian crossing, consisting of one car light and one pedestrian light. Each light is represented by a set of actors (`Pred` and `Pgrn` represent the pedestrian light, and `Cred`, `Cyel` and `Cgrn` represent the car light). A light is *on* iff the corresponding variable has the value 1. The model "generates" failures and repairs by alternating between staying in location `Normal` for 15 time units and staying in location `Abnormal` for 5 time units. Whenever the model operates in `error` mode, all lights should be turned off, except for the yellow car light, which should be blinking. We refer to [4] for a thorough explanation of the model.

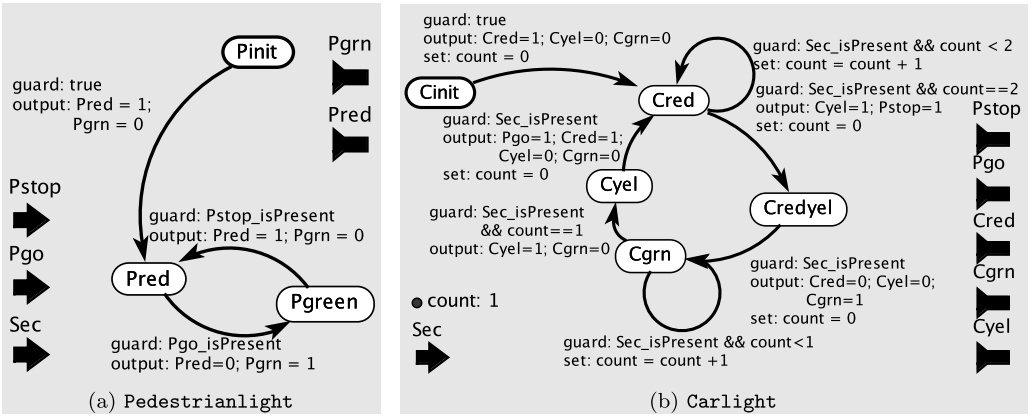


Fig. 8. The FSM actors for pedestrian lights and car lights.

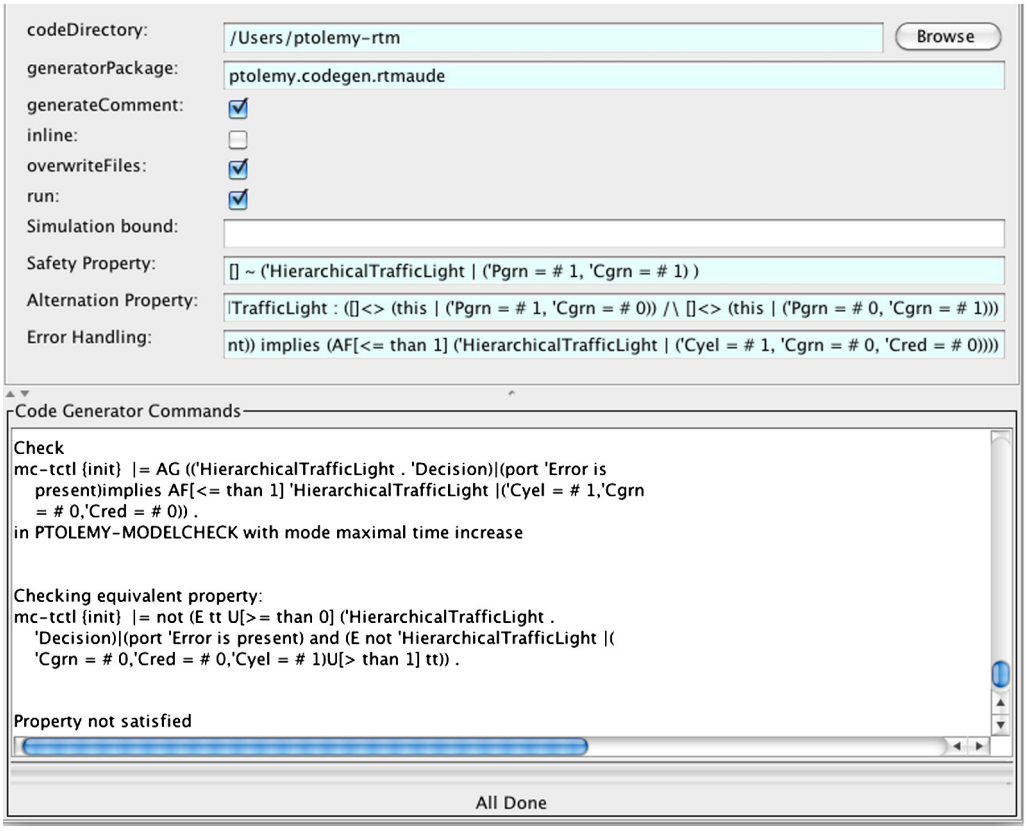


Fig. 9. Dialog window in Ptolemy II that allows us to model check Ptolemy II specifications from within Ptolemy II.

An important fault tolerance property is that the car light will turn yellow, *and only yellow*, within 1 time unit of a failure. We can model check this property with the command:

```
Maude> (mc-tctl-gcd {init} |=
          AG(('HierarchicalTrafficLight . 'Decision | (port 'Error is present)) implies
             AF[<= than 1] ('HierarchicalTrafficLight |
                           ('Cyel = # 1, 'Cgrn = # 0, 'Cred = # 0)))) .)

Property not satisfied
```

Table 5
Execution times for the bridge crossing problem (in seconds).

Initial state	TSMV	Real-Time Maude		RED 7.0
		(pointwise)	(continuous)	
`init(1)`	0.095	0.202	1.266	0.448
`init(10)`	0.161	0.200	0.999	0.408
`init(100)`	1.572	0.195	1.012	0.404
`init(1000)`	70.452	0.196	1.014	0.442
`init+(2)`	0.140	0.834	6.531	1.011
`init+(4)`	0.280	1.936	15.752	1.878
`init+(8)`	0.695	5.607	51.291	15.188
`init+(12)`	0.812	11.717	121.612	72.084
`init+(16)`	1.538	21.952	211.647	244.623

In about 15 seconds, the command returns that the property is not satisfied. We therefore took a very close look at the Ptolemy II visualization of this system, which shows the previously undetected error that, after a failure, the car light may show red or green *in addition* to blinking yellow.

We can also determine a "minimal" time interval such that the above bounded response is satisfied in the system. In particular, we discovered that the interval [5, 12] gives a satisfaction of the bounded response property, by trying different values for a and b in the interval-bounded command

```
Maude> (mc-tctl-gcd {init} |=
           AG(('HierarchicalTrafficLight . 'Decision | (port 'Error is present)) implies
              AF[c a, b c] ('HierarchicalTrafficLight |
                                ('Cyel = # 1, 'Cgrn = # 0, 'Cred = # 0)))) .)
```

Fig. 9 shows the Ptolemy II dialog window resulting from clicking on the (blue) `RTMaudeCodeGenerator` button in the Ptolemy II model in Fig. 7. A number of LTL and CTL formulas have been entered, and the results of the Real-Time Maude model checking of these formulas are shown in the "Code Generator Commands" box.

9. Performance comparison

In this section, we compare the performance of our model checker with the real-time tools TSMV [48] and RED [54] on the bridge crossing benchmark. RED implements a TCTL model checker for linear hybrid automata, and uses dense time. TSMV implements a timed CTL (TCTL with time constraints limited to the form $\sim k$, where $\sim$ can be $<, \leq, =, \geq, >$) model checker for timed Kripke structures with discrete time. Both tools use a symbolic state-space representation, in contrast to our explicit-state model checker. A more thorough discussion on related tools is given in Section 10.

To study the effect of *increasing* the crossing times, but not their relationship, we define the initial state `init(N)` to be parametric in the "scaling factor" `N`:

```
eq init(N) = person(P1 * N,right) person(P2 * N,right)
             person(P3 * N,right) person(P4 * N,right) lamp(right) .
```

Table 5 shows a comparison between our model checker and TSMV and RED.[22] The first 4 rows show a comparison for the formula `AG EF[<= than (110 * n)] safe`, for n the scaling factor. The last 5 rows show a comparison for the same formulas, where `init+(m)` is `init(1)` with m additional `P2` persons, for the formula `AG EF[<= than (110 + m * 15)] safe`. The analysis was performed on a 1.87 GHz Intel® Xeon® with 128 GB of RAM. All specifications are available at http://folk.uio.no/leprid/TCTL-RTM/.

In general, the pointwise semantics of TSMV and Real-Time Maude coincide when the maximal time sampling strategy is selected, while the continuous semantics of RED and Real-Time Maude coincide. Therefore, TSMV should be compared to the "pointwise" results of Real-Time Maude (i.e., running the command `mc-tctl`), while RED should be compared to the "continuous" ones (i.e., running the command `mc-tctl-gcd`).

10. Related work

10.1. Timed Kripke structures

A number of timed extensions of Kripke structures (KSs) have been used as abstract models for real-time systems. For example, [27] and [28] define, respectively, *temporal structures* and labeled *state-transition graphs*; both can be seen as *discrete*

[22] The timed-automata-based tool Uppaal [55] does not support model checking the above TCTL formula.

KSs where each transition has duration one. In these models, a time elapse of duration n is modeled by a sequence of n tick transitions of duration 1, while the same time elapse can be modeled by a single tick transition of duration n in our timed Kripke structures (TKSs). The *timed transition graphs* (TTGs) defined in [28,31] extend state-transition graphs to transitions with duration longer than one time unit; a transition can nondeterministically take any number of unit steps in the interval $[l, u]$, with l, u positive integers. TTGs provide a more succinct representation than our TKSs for discrete time; in particular, a transition with interval $[l, u]$ in a TTG corresponds to the set of transitions of duration n, with $l \leq n \leq u$ in our TKS.

Zero-time transitions are added to state-transition graphs in [29], and [30] defines *discrete* KSs with both unit-time and instantaneous transitions by labeling states where time has elapsed with a special atomic proposition *tick*. Such discrete KSs are extended in [7] by defining *durational transition graphs* (DTGs) where a transition is labeled with an interval $[a, b]$, with $a, b \in \mathbb{N}$, denoting that the transition can have any duration in such interval, and, hence, DTGs provide a more succinct representation than our TKSs for discrete time. *Tight* DTGs, where a transition has exactly one time duration, defined in the same paper, correspond to our TKSs for discrete time.

Timed state graphs (or *tableaux*) are defined in [56] as finite KSs where, in contrast to our TKSs for discrete time, timing information is stored in the states rather than in the transitions; each state is labeled with time-difference propositions $Prev_\delta$, with $\delta \in \mathbb{N}$, indicating that the time-difference from the predecessor state is δ time units. A tick transition from state s to s' with duration δ in our TKS intuitively corresponds to an *untimed* transition from s to $(s', Prev_\delta)$.

While all the models above are restricted to discrete time, our timed Kripke structures are parametric in the time domain, and, in particular, allow dense time.

10.2. Semantics of timed temporal logics

The validity of timed temporal logic formulas is interpreted over the possible behaviors of the given model. We define a pointwise semantics and a continuous semantics for TCTL formulas, which are based on the two corresponding interpretations of possible behaviors of a TKS over its valid configurations: in the pointwise interpretation a tick transition $s_0 \xrightarrow{\tau} s_1$ is seen as an "atomic tick step" that jumps directly from s_0 to s_1; while, in the continuous one, all intermediate configurations can be visited by taking some of the possible (delay) steps $\langle s_0, \delta \rangle \overset{\tau}{\hookrightarrow} \langle s_0, \delta + \tau \rangle$, with $\delta + \tau < \tau'$ for some $\tau' \in \mathcal{T}_{\mathcal{TK}}(s_0)$; the two semantics are furthermore distinguished by the respective notion of pointwise and continuous position of a path.

Likewise, [7] defines both a "jump" and continuous semantics of DTGs, which are intuitively equivalent to, respectively, our pointwise and continuous interpretation of TKSs for discrete time.

For the discrete-time models used in [27–30], where the possible duration of a transition is limited to be either 0 or 1, or where all transitions have duration 1, a distinction between a pointwise and a continuous interpretation is not necessary, since these interpretations would coincide.

While the above work, including our own, consider timed extensions of the *branching time* CTL, the paper [56] considers timed propositional temporal logic (TPTL) a timed extension of the *linear time* PTL. In [56] a pointwise satisfaction relation for TPTL formulas is adopted, while a possible continuous interpretation for timed state graphs is not considered.

The work on the timed transition systems[23] (TTSs) defined in [57] focuses on metric temporal logic (MTL), which extends LTL with bounded modalities of the kind U_I where I is a possibly unbounded time interval whose finite end-points are natural numbers. A continuous interpretation of possible behaviors in the TTS over dense time is used in [57], but, in contrast to our continuous semantics for TCTL that uses a notion of continuous position of a path, the path formulas of the considered temporal logic are evaluated on the pointwise positions of each behavior.

Both linear and branching timed extensions of temporal logic have been considered for timed automata (TA) (see, e.g., [36,32,33]). The first study for TA appears in [6] where a continuous timed CTL semantics is considered; a pointwise and continuous semantics for timed extensions of both LTL and CTL for TA are defined in [36]. These semantics intuitively coincide with our TCTL semantics, which is inspired by the timed CTL semantics for TA in [38].

A continuous timed CTL semantics for dense time, which is similar to the timed CTL semantics for TA and intuitively also coincides with our continuous TCTL semantics, is considered for time Petri nets in [58].

10.3. Timed temporal logic model checking for discrete-time systems

Linear time temporal logic model checking is generally more complex than the branching time one [38]. This is also true for timed extensions of temporal logics. The paper [56] proves that the complexity of model checking TPTL over timed state graphs is EXPSPACE-complete; in particular it is time linear in the size of the timed state graph and doubly exponential in the size of the formula. This is one reason why the following related work all consider various restrictions of TCTL.

The paper [7] presents explicit-state polynomial-time model checking procedures for DTGs and $TCTL_{\leq\geq}$, which is the restricted TCTL with time constraints limited to the form $\sim k$, where $\sim \in \{<, \leq, \geq, >\}$, in both the pointwise and the continuous semantics. Our explicit-state model checking procedure in Section 4 adapts the (pointwise) one in [7], by extending

[23] The dense-time TTSs defined in [57] generalize conventional discrete transition systems by associating a minimal and maximal delay ($l \in \mathbb{N}$ and $u \in \mathbb{N} \cup \{\infty\}$, respectively) to each transition; a transition can be taken only after being continuously enabled for l time units, and it cannot be continuously enabled for more than u time units without being taken.

it to formulas with *interval time bounds* and by simplifying some of the core procedures, which is possible because of our assumption of Zeno-freeness.

In [28,29], the authors give model checking procedures for three classes of properties (the minimum and maximum delay between two states, and the number of occurrences of an event in a given interval) over state-transitions graphs and TTGs. In [31], a model checking algorithm is given for timed CTL formulas, which allow the bounded until operator $U_{[a,b]}$ with $a, b \in \mathbb{N}$, in the continuous semantics. Using a discrete model of time, these procedures all take advantage of symbolic techniques to efficiently represent the transition relation by a binary decision diagram [59] (BDD); this in contrast to our explicit-state model checking procedure.

10.4. Discrete abstraction of dense-time systems for timed temporal logic model checking

A number of discrete abstractions of dense-time systems have been proposed for model checking purpose. For example, the paper [57] shows that a discretization ("digitization") of a TTS is possible, such that model checking some classes of MTL properties—time-bounded invariance and time-bounded response—in this discrete-time model ("digital-clock" model), where each observation time is approximated by an integer, is a sound and complete model checking procedure for the same property in the dense-time model ("analog-clock" model), which records the precise real-numbered time of every observation; e.g., a possible digitization of the dense-time system behavior $(p, 0.2) \to (q, 5.8) \to (p, 5.9) \to (q, 8) \to (p, 8.4)$ is the discrete-time one $(p, 0) \to (q, 6) \to (p, 6) \to (q, 8) \to (p, 8)$. While [57] focuses on a *fragment* of a timed extension of *linear* temporal logic, in this paper, we show how model checking the TCTL_{cb} formula $\beta(\varphi)$ in the *gcd*-transformation of a (dense-time) timed Kripke structure $\mathcal{TK}$ in the pointwise semantics is a sound and complete procedure for model checking the *branching-time* TCTL formula φ in $\mathcal{TK}$ in the continuous semantics. The digitization of a TTS is based on the possibility of identifying *classes of equivalent behaviors*; in particular, a TTS behavior is not "distinguishable" by its digitization (i.e., a behavior and each of its digitization are in the same "equivalence class" in the sense that they satisfy the same fragment of timed linear properties). In comparison, our *gcd*-transformation—that splits each transition in $\mathcal{TK}$ by a sequence of transitions each of duration equal to the half of the greatest common divisor of each non-zero transition duration in $\mathcal{TK}$ and each non-zero finite time bound in φ—is based on the fact that we can identify *classes of valid configurations* in $\mathcal{TK}$, that satisfy the same subformulas of the *full* TCTL formula φ.

As already discussed for discrete time, model checking timed extensions of branching time temporal logics has in general a rather low complexity when compared to the linear time counterparts. The paper [6] shows that model checking a timed CTL formula with time constraints limited to the form $\sim k$, with $k \in \mathbb{N}$ and where $\sim \in \{<, \leq, =, \geq, >\}$, for TA is PSPACE-complete. The idea is to consider a finite quotient of the underlying transition system of a timed automata A, the so-called region transition system, such that model checking an *untimed* CTL formula in this region transition system is equivalent to model checking the original TCTL formula in A. Transforming the *timed* CTL formula φ into an equivalent untimed CTL formula requires the introduction of a new fresh clock z in the timed automata, and the replacement of each finite interval $I \neq [0, \infty)$ in φ with a corresponding atomic clock constraint $z \in I$; CTL model checking can be done in linear time in the size of the transition system, and the size of the region transition system is exponential both in the number of clocks and the length of the timing constraints. In comparison, the size of our *gcd*-transformation for a given timed Kripke structure $\mathcal{TK}$ and a given formula is in the worst case equal the size of $\mathcal{TK}$ multiplied by the ratio of the longest tick transition duration in $\mathcal{TK}$ and the computed greatest common divisor, but our approach does *not* reduce to untimed CTL model checking, yielding a complexity quadratic in the size of the *gcd*-transformation (see Theorem 4.3). Both our *gcd*-transformation and the region construction are based on identifying equivalence classes among the possible system configurations; each state in our *gcd*-transformation represents configurations that satisfy the same subformulas of the given TCTL formula, while each state in the region transition system represents configurations that satisfy the same atomic clock constraints (included the ones over the new clock z).

The region construction for TA suffers from exponential blow-up in the number of clocks and the encoding of the largest integer constant appearing in the clock constraints of the automata, and therefore is in general only used for theoretical complexity results. For practical purposes, several symbolic state-space representation techniques have been proposed: difference-bounded matrices (DBM) for the representation of complex time constraints [60]; and binary decision diagrams (BDD) for the encoding and manipulation of regions [61]. Several BDD-like data structures, improving the results obtained by using BDDs, have been proposed (e.g., numerical decision diagram (NDD) [62], clock-difference diagram (CDD) [63], and clock-restriction diagram (CRD) [54]). Furthermore, to overcome the state space explosion of the region construction, in [64], the authors present time-abstracting bisimulations for TA that, while inducing a coarser untimed state space than the region construction, preserve both linear properties that can be expressed by timed Büchi automata, and branching time properties; model checking the TA against a timed CTL formula[24] is reduced to CTL model checking on the time-abstract finite quotient of the TA. Our model checker does not use any symbolic state-space representation techniques, and all states of our *gcd*-transformation are explicitly encoded in the corresponding TKS.

By using a discretization similar to the region construction for TAs, TCTL model checking for time Petri nets is proven to be PSPACE-complete [58]. The same paper proposes efficient on-the-fly forward model checking algorithms using zone-based

[24] The timed CTL logic considered in [64] allows time interval constraints on the until operator, where the finite interval end-points are in $\mathbb{N}$, conversely to ours that allows end-points in the (dense) time domain.

abstractions for a subset of TPN-TCTL properties; i.e., TPN-TCTL$_S$, consisting of the non-nested until modalities $E\ p\ U_I\ q$, $A\ p\ U_I\ q$, $E\ F_I\ q$, $A\ F_I\ q$, $E\ G_I\ q$, and $A\ G_I\ q$, and bounded response $A\ G\ (p \Longrightarrow A\ F_I\ q)$, where p and q are generalized mutual exclusion constraints [65]. In contrast to this on-the-fly model checking procedure for a subset of timed CTL properties proposed in [58], our approach, which considers full TCTL, requires the computation of the whole reachable state space from the given initial state; while the zone-based abstraction is based on the identification of state classes preserving untimed temporal logic properties, our *gcd*-transformation is based on the identification of classes of valid configurations satisfying the same subformulas of the given TCTL formula.

10.5. TCTL model checkers for real-time systems

The tools Kronos [46], RED [54], and TSMV [48] implement timed CTL model checkers for, respectively, timed automata, linear hybrid automata, and timed Kripke structures that are equivalent to our TKSs. In contrast to our model checker that supports full TCTL model checking in dense time under the continuous semantics, TSMV implements a timed CTL (TCTL with time constraints limited to the form $\sim k$, where $\sim$ can be $<, \leq, =, \geq, >$) model checker for TKSs with discrete time domains under the pointwise semantics. The tool uses symbolic BDD-based techniques.

Kronos implements dense-time timed CTL model checking for timed automata by using an on-the-fly model checking procedure that is based on the time-abstracting bisimulation techniques presented in [64]. Clock constraints are encoded by DBMs.

RED implements dense-time TCTL model checker for linear hybrid automata, and its model checking procedure is based on the symbolic CRD-based techniques presented in [54].

The tool Uppaal [55] provides an efficient symbolic model checking procedure for timed automata for a subset of *non-nested* TCTL properties, which, for example, do not include the TCTL formulas used in the bridge crossing problem in Section 9, the network of medical devices case study in Section 8.1, and the Ptolemy II DE case study in Section 8.2. Clock constraints are encoded as DBMs.

The time-Petri-nets-based tool Roméo [66,58] has an integrated DBM-based on-the-fly timed model checker for *non-nested* TCTL modalities, with the addition of bounded response properties; this fragment of TCTL does not include, e.g., the formulas used in the performance comparison in Section 9, and the network of medical devices case study in Section 8.1.

The tool TINA [67] is also based on time Petri nets, and is provided with an *untimed* linear time temporal logic model checker supporting both state and transition properties. In contrast to our model checker, the tool does not support *timed* temporal logic model checking.

These formalisms are significantly less expressive than real-time rewrite theories [2], which makes their model checking problems decidable. For example, in general, Ptolemy II DE models fall outside the class of systems which can be modeled by timed automata [53], because: certain constructs require the use of data structures (such as lists) of unbounded size; some of the used variables range over infinite domains such as the integers; executing a synchronous step requires a fixed-point computation; and Ptolemy II has a powerful expression language. Furthermore, Real-Time Maude can be used to specify and analyze systems with dynamic object and message creation. In contrast to Kronos, RED, and Uppaal, which have some capabilities for counterexample generation, our model checker does not yet provide features for generating counterexamples/witnesses. While the above tools that support timed temporal logic model checking use some symbolic-state representation technique, our model checker is based on an explicit-state representation.

The first approaches to model checking timed temporal properties for Real-Time Maude are described in [52,68] and analyze important *specific* classes of timed temporal logic formulas (time-bounded response, time-bounded safety, and minimum separation), but only for *flat* object-based specifications. Real-Time Maude comes with model checking features for some of these important subclasses of *metric* linear temporal logic formulas [52]; i.e., bounded response $\Box(p \rightarrow \Diamond_{\leq r} q)$ for p and q two atomic propositions and a time value r, and *minimum separation* formula of the kind $\Box(p \rightarrow (p\ \mathcal{W}\ (\Box_{\leq r} \neg p)))$, which intuitively states that the minimum separation between two non-consecutive p-states is at least r.

Unlike in [52,68], our new model checker is not limited to specific classes of temporal logic properties, but offers the *full* TCTL. The new model checker is also not limited to flat object-oriented systems, but can also analyze any (sensible) Real-Time Maude model, including hierarchical object-oriented specifications, which is crucial, since both AADL and Ptolemy II models are hierarchical.

The work presented in this paper extends the one in [26], where we show our preliminary results for defining a *gcd*-transformation for real-time rewrite theories. That work has been extended here to full TCTL. Furthermore, the definition of pointwise and continuous semantics of [26] are different from the ones used in this paper, and are based on the notion of "time-refinement" of a path. This semantics is slightly "weaker" than the one used in this paper, and therefore it allows the sound and completeness result to be achieved without the need of a transformation of φ into $\beta(\varphi)$.

11. Conclusions and future work

We have defined the pointwise and the continuous semantics for timed Kripke structures (with discrete or dense time domain), and have shown how dense-time TCTL model checking in the continuous semantics can be reduced to TCTL model checking in the pointwise semantics, for which we describe a TCTL model checking procedure, for large classes of

Table 6
Axioms in the theory *TIME*, for $\tau_1, \tau_2, \tau_3 \in Time$.

A1.	$(\tau_1 + \tau_2) + \tau_3 = \tau_1 + (\tau_2 + \tau_3)$	A9.	$0 \leq \tau_1$
A2.	$\tau_1 + 0 = \tau_1$	A10.	$\tau_1 \leq \tau_2 \iff \exists \tau_3 : \tau_1 + \tau_3 = \tau_2 \wedge \tau_3 = \tau_2 \dot{-} \tau_1$
A3.	$\tau_1 + \tau_2 = \tau_2 + \tau_1$	A11.	$max(\tau_1, \tau_2) = max(\tau_2, \tau_1)$
A4.	$\tau_1 < \tau_2 \vee \tau_1 = \tau_2 \vee \tau_2 < \tau_1$	A12.	$max(max(\tau_1, \tau_2), \tau_3) = max(\tau_1, max(\tau_2, \tau_3))$
A5.	$\neg(\tau_1 < \tau_1)$	A13.	$\tau_1 \leq \tau_2 \Longrightarrow max(\tau_1, \tau_2) = \tau_2$
A6.	$\tau_1 < \tau_2 \Longrightarrow \tau_1 + \tau_3 < \tau_2 + \tau_3$	A14.	$min(\tau_1, \tau_2) = min(\tau_2, \tau_1)$
A7.	$\tau_1 + \tau_2 = \tau_1 + \tau_3 \Longrightarrow \tau_2 = \tau_3$	A15.	$min(min(\tau_1, \tau_2), \tau_3) = min(\tau_1, min(\tau_2, \tau_3))$
A8.	$\tau_1 \leq \tau_2 \iff \tau_1 < \tau_2 \vee \tau_1 = \tau_2$		

timed Kripke structures. In particular, these classes of timed Kripke structures include those that are defined by time-robust real-time rewrite theories.

Based on our TCTL pointwise model checking procedure and the *gcd*-transformation, we have developed a TCTL model checker for Real-Time Maude. This means that we can now perform (sound and complete) TCTL model checking on a number of complex dense-time systems that are very hard or impossible to model using other formal real-time tools. Furthermore, since real-time rewrite theories are a suitable semantic framework for other real-time (modeling) languages, our model checker now also provides for free a TCTL model checker for such languages, which include (subsets of) modeling languages used in industry such as Ptolemy II and the avionics modeling standard AADL. Indeed, our TCTL model checker has been integrated into the Ptolemy II tool, and has been used to find a previously undetected flaw in an existing Ptolemy II model.

We should extend our model checker to provide useful counterexamples/witnesses instead of just a yes/no answer. We should also identify other classes of systems that satisfy the requirements for sound and complete *gcd*-based TCTL model checking. Finally, the current version of our tool is implemented in Maude at the Maude meta-level; for efficiency purposes, it should be further optimized and implemented in C++.

Acknowledgements

We would like to thank the anonymous reviewers for their very insightful and useful comments on previous versions of this paper. This work was partially supported by the Research Council of Norway through the Rhytm project (Grant 142808) and by the DAADppp HySmart project.

Appendix A. Time domains

Time domains as first-order theories We define the theory *TIME* as a linearly ordered commutative monoid $(Time, +, 0, <)$, with a monus operator $\dot{-}$ and the binary commutative and associative operators *max* and *min*:

$$\dot{-} : Time \times Time \to Time$$
$$max : Time \times Time \to Time$$
$$min : Time \times Time \to Time$$

satisfying the axioms in Table 6 for each $\tau_1, \tau_2, \tau_3 \in Time$.

We use τ, τ', τ_1, ... to denote time values, and we write $\tau_1 < \tau < \tau_2$ when $\tau_1 < \tau \wedge \tau < \tau_2$, $\tau_1 \leq \tau_2$ when $\tau_1 < \tau_2 \vee \tau_1 = \tau_2$, and $k\tau$ or $k \cdot \tau$ for $\underbrace{\tau + \tau + \cdots + \tau}_{k \text{ times}}$. Furthermore, we denote $Time \setminus \{0\}$ by $Time_{>0}$.

The theory $TIME_\infty$ extends *TIME* with the new infinity element $\infty \notin Time$, such that $Time_\infty = Time \cup \{\infty\}$ and the operators $<$, $+$, $\dot{-}$, *max* and *min* are extended to $Time_\infty$ in the following way:

$$< : Time_\infty \times Time_\infty \to Bool$$
$$+ : Time_\infty \times Time_\infty \to Bool$$
$$\dot{-} : Time_\infty \times Time \to Time_\infty$$
$$max : Time_\infty \times Time_\infty \to Time_\infty$$
$$min : Time_\infty \times Time_\infty \to Time_\infty$$

where *Bool* is the usual boolean values, and the axioms in Table 7 are satisfied for each $\tau_1 \in Time$ and $\tau_2, \tau_3 \in Time_\infty$.

The theory $TIME^{gcd}$ extends *TIME* with the functions

$$| : Time_{>0} \times Time_{>0} \to Bool$$
$$gcd : Time_{>0} \times Time_{>0} \to Time_{>0}$$
$$half : Time_{>0} \to Time_{>0}$$

satisfying the axioms in Table 8 for each $\tau_1, \tau_2, \tau_3 \in Time_{>0}$.

Table 7
Axioms in the theory $TIME_\infty$, where $\tau_1 \in Time$ and $\tau_2, \tau_3 \in Time_\infty$.

A16. $\tau_1 < \infty$	A20. $\infty + \tau_2 = \infty$
A17. $\neg(\infty \leq \tau_1)$	A21. $\infty \dot{-} \tau_1 = \infty$
A18. $\tau_2 \leq \infty$	A22. $max(\tau_2, \infty) = \infty$
A19. $\neg(\infty < \tau_2)$	A23. $max(\tau_2, \tau_3) = \tau_2 \Longrightarrow min(\tau_2, \tau_3) = \tau_3$

Table 8
Axioms for the theory $TIME^{gcd}$, where $\tau_1, \tau_2, \tau_3 \in Time_{>0}$.

A24. $\tau_1 \mid \tau_2 \wedge \tau_2 \mid \tau_3 \Longrightarrow \tau_1 \mid \tau_3$	A30. $gcd(\tau_1, \tau_2) = gcd(\tau_2, \tau_1)$
A25. $\tau_1 \mid \tau_2 \wedge \tau_2 \mid \tau_1 \Longrightarrow \tau_1 = \tau_2$	A31. $gcd(gcd(\tau_1, \tau_2), \tau_3) = gcd(\tau_1, gcd(\tau_2, \tau_3))$
A26. $\tau_1 \mid \tau_1$	A32. $gcd(\tau_1, \tau_2) \mid \tau_1$
A27. $\tau_1 \mid \tau_2 \wedge \tau_1 \mid \tau_3 \Longrightarrow \tau_1 \mid (\tau_2 + \tau_3)$	A33. $(\tau_3 \mid \tau_1 \wedge \tau_3 \mid \tau_2) \Longrightarrow \tau_3 \mid gcd(\tau_1, \tau_2)$
A28. $\tau_2 < \tau_1 \Longrightarrow \neg(\tau_1 \mid \tau_2)$	A34. $half(\tau_1) + half(\tau_1) = \tau_1$
A29. $\tau_1 \mid (\tau_1 + \tau_2) \Longrightarrow \tau_1 \mid \tau_2$	

Time domains as Real-Time Maude functional modules The Real-Time Maude module `LTIME-INF` satisfies the theory LTIME_∞ in [2]. As done above for functional theories, we can extend it with a Real-Time Maude module `GCD-TIME` satisfying the theory $TIME^{gcd}_\infty$:

```
fth GCD-TIME is including LTIME-INF .
 sort NzTime .   subsort NzTime < Time .
 cmb T:Time : NzTime if T:Time =/= 0 .

 op gcd : NzTime NzTime -> NzTime [assoc comm] .
 op _divides_ : NzTime NzTime -> Bool .
 op half : NzTime -> NzTime .

 vars T1 T2 T3 : NzTime . vars T T' : Time .

 eq T1 divides T1 = true .
 ceq T1 divides T2 = false if T2 < T1 .
 eq T1 divides (T1 + T2) =  T1 divides T2 .

 eq gcd(T1, T2) divides T1 = true .
 ceq gcd(T1, T2) >= T3  if T3 divides T1 /\ T3 divides T2 .

 eq half(NZT) + half(NZT) = NZT .
endfth
```

Appendix B. Proofs of lemmas and theorems

Lemma 3.3. *Let AP be a set of atomic propositions and $\mathcal{TK}$ a timed Kripke structure over AP whose time domain satisfies the theory $TIME^{gcd}$. Let φ be a $TCTL_{cb}$ formula over AP, $0 < \tau \in \mathcal{T}$ such that $GCD(\mathcal{TK}, \varphi)$ is a multiple of 2τ and $\mathcal{TK}^\tau = (S, \mathcal{T}, \longrightarrow, L)$ the τ-transformation of $\mathcal{TK}$. Then for each valid configuration σ of $\mathcal{TK}^\tau$ it holds that*

$$\mathcal{TK}^\tau, \sigma \models_c \varphi \quad \Longleftrightarrow \quad \mathcal{TK}^\tau, can(\sigma) \models_c \varphi.$$

Proof. Assume *AP*, $\mathcal{TK}$, φ, τ, $\mathcal{TK}^\tau$ and $\sigma = \langle (s_0, n_0\tau), \delta_0 \rangle$ as described in the lemma. If $can(\sigma) = \sigma$ then the statement holds trivially. Let therefore in the following $can(\sigma) \neq \sigma$.

The proof of the lemma is by structural induction over the structure of φ. Note that if $GCD(\mathcal{TK}, \varphi)$ is a multiple of 2τ then also $GCD(\mathcal{TK}, \varphi')$ is a multiple of 2τ for all subformulas φ' of φ.

The case $\varphi = true$ is trivial. For the base case of atomic propositions the equivalence follows from the definition of $can(\sigma)$ (Definition 3.3), and the fact that $L(s_0, n_0\tau) = L(s_0, n'_0\tau)$ for all states $(s_0, n'_0\tau)$ of $\mathcal{TK}^\tau$ by the definition of $\mathcal{TK}^\tau$ (Definition 3.1).

Assume now that the lemma holds for φ_1 and φ_2.

- $\varphi = \neg\varphi_1$: Note that if $GCD(\mathcal{TK}, \varphi)$ is a multiple of 2τ then also $GCD(\mathcal{TK}, \varphi_1)$ is a multiple of 2τ.

$$\begin{aligned} \mathcal{TK}^\tau, \sigma \models_c \neg\varphi_1 \quad &\Longleftrightarrow \quad not\ (\mathcal{TK}^\tau, \sigma \models_c \varphi_1) \\ &\overset{induction}{\Longleftrightarrow} not\ (\mathcal{TK}^\tau, can(\sigma) \models_c \varphi_1) \\ &\Longleftrightarrow \quad \mathcal{TK}^\tau, can(\sigma) \models_c \neg\varphi_1 \end{aligned}$$

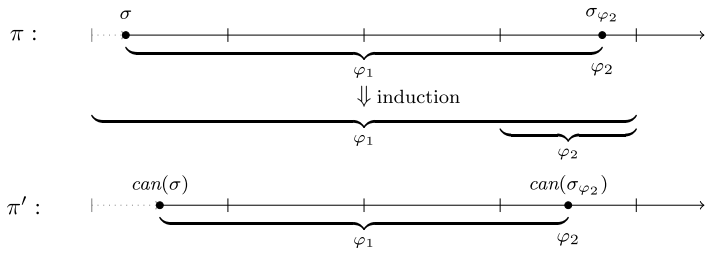


Fig. 10. Existential until, case "$\Longrightarrow$".

- $\varphi = \varphi_1 \wedge \varphi_2$:

$$\begin{aligned} \mathcal{TK}^\tau, \sigma \models_c \varphi_1 \wedge \varphi_2 \quad &\Longleftrightarrow \quad \mathcal{TK}^\tau, \sigma \models_c \varphi_1 \text{ and } \mathcal{TK}^\tau, \sigma \models_c \varphi_2 \\ &\overset{\text{induction}}{\Longleftrightarrow} \quad \mathcal{TK}^\tau, \mathit{can}(\sigma) \models_c \varphi_1 \text{ and } \mathcal{TK}^\tau, \mathit{can}(\sigma) \models_c \varphi_2 \\ &\Longleftrightarrow \quad \mathcal{TK}^\tau, \mathit{can}(\sigma) \models_c \varphi_1 \wedge \varphi_2 \end{aligned}$$

For the remaining cases, note that for each path from $\mathit{tdPaths}_{\mathcal{TK}^\tau}(\sigma)$ there is also a path $\pi \in \mathit{tdPaths}_{\mathcal{TK}^\tau}(\sigma)$ (analogously to the representative elements in the proof of Lemma 3.1) of the form

$$\pi = \underbrace{\langle (s_0, n_0\tau), \delta_0 \rangle}_{\sigma} \xrightarrow{\tau_0 = \tau - \delta_0} \langle (s_1, n_1\tau), 0 \rangle \xrightarrow{\tau_1 \in \{0,\tau\}} \langle (s_2, n_2\tau), 0 \rangle \xrightarrow{\tau_2 \in \{0,\tau\}} \ldots. \tag{B.1}$$

Remember that by assumption $\delta_0 > 0$. Note furthermore that

$$\pi' = \begin{cases} \underbrace{\langle (s_0, n_0\tau), 0 \rangle}_{\mathit{can}(\sigma)} \xrightarrow{\tau} \langle (s_1, n_1\tau), 0 \rangle \xrightarrow{\tau_1 \in \{0,\tau\}} \langle (s_2, n_2\tau), 0 \rangle \xrightarrow{\tau_2 \in \{0,\tau\}} \ldots & \text{if } n_0 \text{ is odd} \\ \underbrace{\langle (s_1, n_1\tau), 0 \rangle}_{\mathit{can}(\sigma)} \xrightarrow{\tau_1 \in \{0,\tau\}} \langle (s_2, n_2\tau), 0 \rangle \xrightarrow{\tau_2 \in \{0,\tau\}} \ldots & \text{else} \end{cases} \tag{B.2}$$

is a time-divergent path starting at $\mathit{can}(\sigma)$ and differing from π only by either removing or extending its first transition.

- "$\Longrightarrow$" case for $\varphi = E\ \varphi_1\ U_I\ \varphi_2$ (see Fig. 10)
  Assume $\mathcal{TK}^\tau, \sigma \models_c E\ \varphi_1\ U_I\ \varphi_2$. By the semantics of TCTL there is a time-divergent path of $\mathcal{TK}^\tau$ starting at σ and satisfying the until formula in the continuous semantics. Then there is also a time-divergent path π of $\mathcal{TK}^\tau$ as in Eq. (B.1) having a continuous position $\sigma_{\varphi_2}@c_{\varphi_2}$, with $c_{\varphi_2} \in I$ and $\mathcal{TK}^\tau, \sigma_{\varphi_2} \models_c \varphi_2$ such that $\mathcal{TK}^\tau, \sigma_{\varphi_1} \models_c \varphi_1$ for all continuous positions $\sigma_{\varphi_1}@c_{\varphi_1}$ in $\mathit{pre}^c_\pi(\sigma_{\varphi_2}@c_{\varphi_2})$.
  We show that $\mathit{can}(\sigma)$ also satisfies the existential until formula, using the path π' from Eq. (B.2) which starts at $\mathit{can}(\sigma)$. We show that π' reaches $\mathit{can}(\sigma_{\varphi_2})$ in a duration in the required interval I such that φ_2 holds at $\mathit{can}(\sigma_{\varphi_2})$ and φ_1 holds for all the positions before.
  From $\mathcal{TK}^\tau, \sigma_{\varphi_2} \models_c \varphi_2$ we get by induction that $\mathcal{TK}^\tau, \mathit{can}(\sigma_{\varphi_2}) \models_c \varphi_2$. Furthermore, using Lemma 3.2 we conclude that the time stamp of the position of $\mathit{can}(\sigma_{\varphi_2})$ in π' is in the interval I.
  If $c_{\varphi_2} < 2\tau$ then $0 \in I$ and we are done. Assume now $c_{\varphi_2} \geq 2\tau$. It remains to show that φ_1 holds along π' all the time before $\mathit{can}(\sigma_{\varphi_2})$.
  We know that $\sigma@0$, appearing prior to σ_{φ_2} in π, satisfies φ_1. We get by induction that all configurations at time stamps $[0, \tau)$ in π' satisfy φ_1.
  All the remaining configurations prior to $\mathit{can}(\sigma_{\varphi_2})$ in π' are also in π. If they all satisfy φ_1 by assumption then we are done. Otherwise, σ_{φ_2} has the form $\langle (s_{\varphi_2}, 2n_{\varphi_2}\tau), \tau_{\varphi_2} \rangle$ with $0 < \tau_{\varphi_2} < \tau$, and we have to show that all $\langle (s_{\varphi_2}, 2n_{\varphi_2}\tau), \bar{\tau} \rangle$ with $0 < \bar{\tau} < \tau$ satisfy φ_1. By assumption any $\langle (s_{\varphi_2}, 2n_{\varphi_2}\tau), \tau^* \rangle$, with $0 < \tau^* < \tau_{\varphi_2}$ also satisfies φ_1. By induction all configurations $\langle (s_{\varphi_2}, 2n_{\varphi_2}\tau), \bar{\tau} \rangle$ with $0 < \bar{\tau} < \tau$ also satisfy φ_1.
  Concluding, $\mathcal{TK}^\tau, \pi' \models_c \varphi_1\ U_I\ \varphi_2$, i.e., $\mathcal{TK}^\tau, \mathit{can}(\sigma) \models_c E\ \varphi_1\ U_I\ \varphi_2$.
- "$\Longleftarrow$" case for $\varphi = E\ \varphi_1\ U_I\ \varphi_2$ (see Fig. 11)
  Assume $\mathcal{TK}^\tau, \mathit{can}(\sigma) \models_c E\ \varphi_1\ U_I\ \varphi_2$. By the semantics of TCTL there is a time-divergent path of $\mathcal{TK}^\tau$ starting at $\mathit{can}(\sigma)$ and satisfying the until formula in the continuous semantics. Then there is also a time-divergent path π' of $\mathcal{TK}^\tau$ of the form given in Eq. (B.2), containing a continuous position $\sigma_{\varphi_2}@c_{\varphi_2}$ with $c_{\varphi_2} \in I$ and $\mathcal{TK}^\tau, \sigma_{\varphi_2} \models_c \varphi_2$ such that $\mathcal{TK}^\tau, \sigma_{\varphi_1} \models_c \varphi_1$ for all continuous positions $\sigma_{\varphi_1}@c_{\varphi_1}$ in $\mathit{pre}^c_{\pi'}(\sigma_{\varphi_2}@c_{\varphi_2})$.
  We show that the same holds for $\mathit{can}(\sigma_{\varphi_2})$ instead of σ_{φ_2}. If $\sigma_{\varphi_2} = \mathit{can}(\sigma_{\varphi_2})$ then the statement is trivial. Otherwise, by induction also $\mathit{can}(\sigma_{\varphi_2})$ satisfies φ_2. Furthermore, Lemma 3.2 implies that the time stamp of $\mathit{can}(\sigma_{\varphi_2})$ in π' is in I. By assumption all positions before $\sigma_{\varphi_2} = \langle (s_{\varphi_2}, \delta_{\varphi_2}), \tau_{\varphi_2} \rangle$ at time c_{φ_2} in π' satisfy φ_1. This holds especially for all the

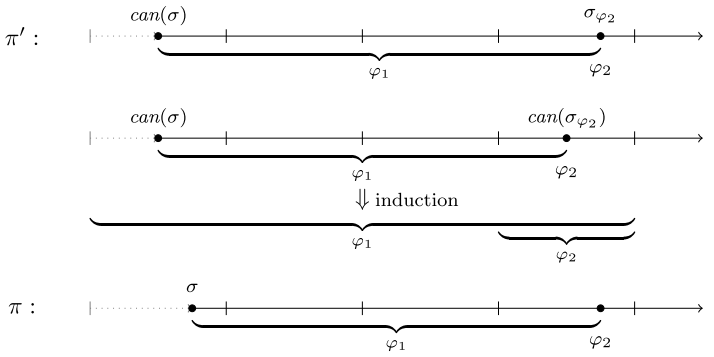


Fig. 11. Existential until, case "$\Longleftarrow$".

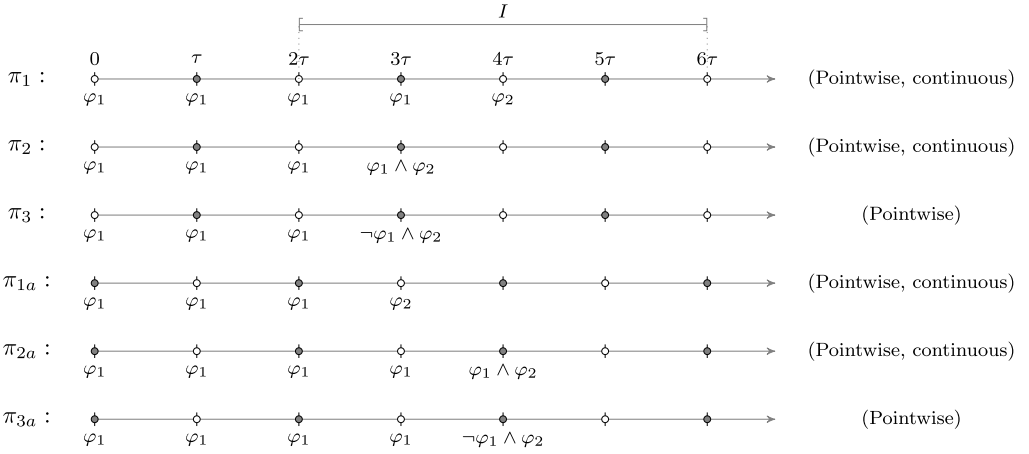


Fig. 12. Structure of paths (shown in the pointwise interpretation) that satisfy the TCTL$_{cb}$ formula $\psi = \varphi_1\, U_I\, \varphi_2$ in the *pointwise* semantics. Gray circles denote abstract configurations and white circles denote concrete ones. Paths that satisfy ψ in the *continuous* semantics have the structure π_1, π_2, π_3, π_{1a}, π_{2a} or π_{3a}.

configurations $\langle(s_{\varphi_2}, \delta_{\varphi_2}), \tau^*\rangle$ such that $0 < \tau^* < \tau_{\varphi_2}$ (note that $\tau_{\varphi_2} > 0$, otherwise $\sigma_{\varphi_2} = can(\sigma_{\varphi_2})$ would hold). By induction, all remaining configurations $\langle(s_{\varphi_2}, \delta_{\varphi_2}), \bar{\tau}\rangle$ before $can(\sigma_{\varphi_2})$ in π' (if any) also satisfy φ_1.
Finally we construct a path π from π' as shown in Eq. (B.1) and illustrated in Fig. 11, witnessing $\mathcal{TK}^\tau, \sigma \models_c E\, \varphi_1\, U_I\, \varphi_2$: If n_0 in $\sigma = \langle(s_0, n_0\tau), \delta_0\rangle$ is odd then we shorten the first transition $\langle(s_0, n_0\tau), 0\rangle \xrightarrow{\tau} \langle(s_1, n_1\tau), 0\rangle$ of π' to $\langle(s_0, n_0\tau), \delta_0\rangle \xrightarrow{\tau-\delta_0} \langle(s_1, n_1\tau), 0\rangle$. Otherwise, if n_0 is even, we insert a new transition $\langle(s_0, n_0\tau), \delta_0\rangle \xrightarrow{\tau-\delta_0} \langle(s_1, n_1\tau), 0\rangle$ at the beginning of π'.
Let c'_{φ_2} be the time stamp of $can(\sigma_{\varphi_2})$ in π'. By assumption and by induction, all continuous positions before $can(\sigma_{\varphi_2})@c'_{\varphi_2}$ in π satisfy φ_1 and $can(\sigma_{\varphi_2})@c'_{\varphi_2}$ satisfies φ_2. Since $c'_{\varphi_2} \in I$, the path π satisfies $\varphi_1\, U_I\, \varphi_2$ and therefore σ satisfies the existential until formula.

- "$\Longrightarrow$" case for $\varphi = A\, \varphi_1\, U_I\, \varphi_2$
  Assume $\mathcal{TK}^\tau, \sigma \models_c A\, \varphi_1\, U_I\, \varphi_2$. We need to show that all time-divergent paths of $\mathcal{TK}^\tau$ starting in $can(\sigma)$ also satisfy $\varphi_1\, U_I\, \varphi_2$. Assume such a path, which we bring to the TCTL-equivalent form π' given in Eq. (B.2). According to Eq. (B.1), we construct a path π starting in σ, which by assumption satisfies $\varphi_1\, U_I\, \varphi_2$. We can use the same argumentation as in the "$\Longrightarrow$" case of the existential until to show that also π' satisfies $\varphi_1\, U_I\, \varphi_2$.
- "$\Longleftarrow$" case for $\varphi = A\, \varphi_1\, U_I\, \varphi_2$
  Assume $\mathcal{TK}^\tau, can(\sigma) \models_c A\, \varphi_1\, U_I\, \varphi_2$. We need to show that all time-divergent paths of $\mathcal{TK}^\tau$ starting in σ also satisfy $\varphi_1\, U_I\, \varphi_2$. Assume such a path, which we bring to the TCTL-equivalent form π given in Eq. (B.1). According to Eq. (B.2), we construct a path π' starting in $can(\sigma)$, which by assumption satisfies $\varphi_1\, U_I\, \varphi_2$. We can use the same argumentation as in the "$\Longleftarrow$" case of the existential until to show that also π satisfies $\varphi_1\, U_I\, \varphi_2$. □

Theorem 3.1. *Let AP be a set of atomic propositions and $\mathcal{TK}$ a timed Kripke structure over AP whose time domain satisfies the theory TIME^{gcd}. Let φ be a TCTL$_{cb}$ formula over AP, $0 < \tau \in \mathcal{T}$ such that GCD$(\mathcal{TK}, \varphi)$ is a multiple of 2τ and $\mathcal{TK}^\tau_a = (S, \mathcal{T}, \longrightarrow, L)$ the abstract τ-transformation of $\mathcal{TK}$, so that $L : S \to \mathcal{P}(AP_a)$ and $AP_a = AP \uplus \{p_a\}$ are as given in the definition of $\mathcal{TK}^\tau_a$. Then for each state (s, δ) of $\mathcal{TK}^\tau_a$ and each subformula φ' of φ, it holds that*

$$\mathcal{TK}^\tau_a, (s, \delta) \models_c \varphi' \quad \Longleftrightarrow \quad \mathcal{TK}^\tau_a, (s, \delta) \models_p \beta(\varphi').$$

Proof. First we observe that paths satisfying $\varphi_1\, U_I\, \varphi_2$ in the *pointwise* semantics can have different forms: either $0 \in I$ and the first configuration on the path satisfies φ_2 or the path has one of the forms shown in Fig. 12. Please note that all π_i start in a concrete state while all π_{ia} start in an abstract one.

The structure of paths that satisfy the until formula in the *continuous* semantics are the same but without π_3 and π_{3a} (as shown in Example 3.11).

Assume $\mathcal{TK}$, φ, τ, $\mathcal{TK}_a^\tau$, AP_a and (s,δ) as described in the theorem, and let $\sigma = \langle (s,\delta), 0\rangle$. The proof is by structural induction over the structure of φ. Note that if $GCD(\mathcal{TK},\varphi)$ is a multiple of 2τ then also $GCD(\mathcal{TK},\varphi')$ is a multiple of 2τ for all subformulas φ' of φ.

The case $\varphi = true$ is trivial. For the base case of atomic propositions we have $\beta(\varphi) = \varphi$, and the equivalence follows from the fact that $L(s,\delta) = L(s,\delta')$ for all states (s,δ), (s,δ') of $\mathcal{TK}_a^\tau$ by the definition of $\mathcal{TK}_a^\tau$.

Assume now that the theorem holds for φ_1 and φ_2.

- $\varphi = \neg\varphi_1$:

$$\begin{aligned}
\mathcal{TK}_a^\tau, \sigma \models_c \neg\varphi_1 \quad &\iff \quad not\ (\mathcal{TK}_a^\tau, \sigma \models_c \varphi_1)\\
&\overset{induction}{\iff} \quad not\ (\mathcal{TK}_a^\tau, \sigma \models_p \beta(\varphi_1))\\
&\iff \quad \mathcal{TK}_a^\tau, \sigma \models_p \neg\beta(\varphi_1)\\
&\iff \quad \mathcal{TK}_a^\tau, \sigma \models_p \beta(\neg\varphi_1)
\end{aligned}$$

- $\varphi = \varphi_1 \wedge \varphi_2$:

$$\begin{aligned}
\mathcal{TK}_a^\tau, \sigma \models_c \varphi_1 \wedge \varphi_2 \quad &\iff \quad \mathcal{TK}_a^\tau, \sigma \models_c \varphi_1\ and\ \mathcal{TK}_a^\tau, \sigma \models_c \varphi_2\\
&\overset{induction}{\iff} \quad \mathcal{TK}_a^\tau, \sigma \models_p \beta(\varphi_1)\ and\ \mathcal{TK}_a^\tau, \sigma \models_p \beta(\varphi_2)\\
&\iff \quad \mathcal{TK}_a^\tau, \sigma \models_p \beta(\varphi_1) \wedge \beta(\varphi_2)\\
&\iff \quad \mathcal{TK}_a^\tau, \sigma \models_p \beta(\varphi_1 \wedge \varphi_2)
\end{aligned}$$

- "$\Longrightarrow$" case for $\varphi = E\ \varphi_1\, U_I\, \varphi_2$
  Assume $\mathcal{TK}_a^\tau, \sigma \models_c E\ \varphi_1\, U_I\, \varphi_2$. By the continuous semantics of TCTL there is a path $\pi \in tdPaths^c_{\mathcal{TK}_a^\tau}(\sigma)$ satisfying the until formula. Notice that $can(\sigma) = \sigma$. As explained in the proof for Lemma 3.3, the time-divergent path π' in $tdPaths^p_{\mathcal{TK}_a^\tau}(\sigma)$ (described in Eq. (B.2)) also satisfies the until formula in the continuous semantics, in particular: π' has a continuous position $\sigma_{\varphi_2}@c_{\varphi_2}$, with $c_{\varphi_2} \in I$ and $\mathcal{TK}_a^\tau, \sigma_{\varphi_2} \models_c \varphi_2$ such that $\mathcal{TK}_a^\tau, \sigma_{\varphi_1} \models_c \varphi_1$ for all continuous positions $\sigma_{\varphi_1}@c_{\varphi_1}$ in $pre^c_{\pi'}(\sigma_{\varphi_2}@c_{\varphi_2})$.
  It is easy to show that π' is a path that guarantees the satisfaction of $\beta(E\ \varphi_1\, U_I\, \varphi_2)$ in σ in the pointwise semantics. If $c_{\varphi_2} = 0$ and $0 \in I$, then $\mathcal{TK}_a^\tau, \sigma \models_c \varphi_2$ and $\mathcal{TK}_a^\tau, \sigma \models_p \beta(\varphi_2)$ follows by induction. Notice that if $0 \notin I$, then it must be the case that $c_{\varphi_2} \geq 2\tau$. We distinguish two cases:
  - the configuration σ_{φ_2} corresponds to a $\neg p_a$-state and $\beta(E\ \varphi_1\, U_I\, \varphi_2) = E\ \beta(\varphi_1)\, U_I\, \beta(\varphi_2)$; then $\mathcal{TK}_a^\tau, \sigma \models_p \beta(E\ \varphi_1\, U_I\, \varphi_2)$ follows trivially by induction, since the pointwise positions of π' satisfy the formula $E\ \beta(\varphi_1)\, U_I\, \beta(\varphi_2)$ in the pointwise semantics.
  - the configuration σ_{φ_2} corresponds to a p_a-state and $\beta(E\ \varphi_1\, U_I\, \varphi_2) = E\ \beta(\varphi_1)\, U_I\, \beta(\varphi_2) \wedge \beta(\varphi_1)$; then we can observe that, by Lemma 3.3 it follows that $\mathcal{TK}_a^\tau, \sigma_{\varphi_2} \models_c \varphi_1$ as well, since there exists a continuous position $\sigma_{\varphi_1}@c_{\varphi_1}$ in π' with $c_{\varphi_2} - \tau < c_{\varphi_1} < c_{\varphi_2}$, i.e., $can(\sigma_{\varphi_1}) = \sigma_{\varphi_2}$. By induction we have that $\mathcal{TK}_a^\tau, \sigma_{\varphi_2} \models_p \beta(\varphi_1)$ and, by definition of $\models_p$ we have that

    $$\mathcal{TK}_a^\tau, \sigma_{\varphi_2} \models_p \beta(\varphi_2) \wedge \beta(\varphi_1).$$

    We can conclude that $\mathcal{TK}_a^\tau, \sigma \models_p \beta(E\ \varphi_1\, U_I\, \varphi_2)$, since the pointwise positions of π' satisfy the formula $E\ \beta(\varphi_1)\, U_I\, \beta(\varphi_2) \wedge \beta(\varphi_1)$.
  Hence, we have that in every case $\mathcal{TK}_a^\tau, \sigma \models_p \beta(E\ \varphi_1\, U_I\, \varphi_2)$.
- "$\Longleftarrow$" case for $\varphi = E\ \varphi_1\, U_I\, \varphi_2$
  Assume $\mathcal{TK}_a^\tau, \sigma \models_p \beta(E\ \varphi_1\, U_I\, \varphi_2)$. We distinguish two cases:
  (i) $0 \in I$ and $\beta(E\ \varphi_1\, U_I\, \varphi_2) = \beta(\varphi_2) \vee E\ \beta(\varphi_1)\, U_I\, (\beta(\varphi_2) \wedge (\neg p_a \vee \beta(\varphi_1)))$. If $\mathcal{TK}_a^\tau, \sigma \models_p \beta(\varphi_2)$ then by induction we have that $\mathcal{TK}_a^\tau, \sigma \models_c \varphi_2$ and hence $\mathcal{TK}_a^\tau, \sigma \models_c \varphi$. Otherwise,

  $$\mathcal{TK}_a^\tau, \sigma \models_p E\ \beta(\varphi_1)\, U_I\, (\beta(\varphi_2) \wedge (\neg p_a \vee \beta(\varphi_1))).$$

  By the pointwise semantics of TCTL there is a path $\pi \in tdPaths^p_{\mathcal{TK}_a^\tau}(\sigma)$ satisfying the until formula. In particular, there exists a pointwise position $\sigma_{\varphi_2}@c_{\varphi_2}$, with $c_{\varphi_2} \in I$ and $\mathcal{TK}_a^\tau, \sigma_{\varphi_2} \models_p \beta(\varphi_2) \wedge (\neg p_a \vee \beta(\varphi_1))$ such that $\mathcal{TK}_a^\tau, \sigma_{\varphi_1} \models_p \beta(\varphi_1)$ for all pointwise positions $\sigma_{\varphi_1}@c_{\varphi_1}$ in $pre^p_\pi(\sigma_{\varphi_2}@c_{\varphi_2})$. By induction we get that $\mathcal{TK}_a^\tau, \sigma_{\varphi_2} \models_c \varphi_2$ and $\mathcal{TK}_a^\tau, \sigma_{\varphi_1} \models_c \varphi_1$ for all pointwise positions $\sigma_{\varphi_1}@c_{\varphi_1}$ in π with $c_{\varphi_1} < c_{\varphi_2}$. Moreover, we can use Lemma 3.3 on

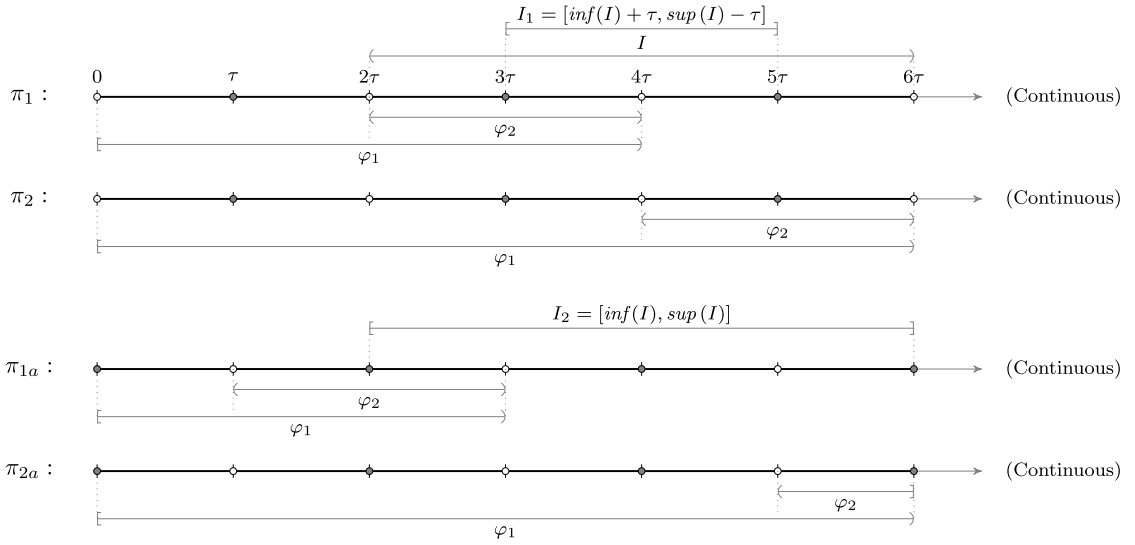


Fig. 13. Some paths in $\mathcal{TK}_a^\tau$ satisfying the TCTL formula $\psi = \varphi_1\ U_I\ \varphi_2$, with $I = (2\tau, 6\tau)$, in the continuous semantics. Gray circles denote abstract configurations and white circles denote concrete ones.

each such pointwise positions $\sigma_{\varphi_1}@c_{\varphi_1}$ where $\sigma_{\varphi_1} = ((s_{\varphi_1}, (2n_{\varphi_1} + 1)\tau), 0)$, for $n_{\varphi_1} \in \mathbb{N}$, is an "abstract" configuration, to show that each continuous position before σ_{φ_2} also satisfy φ_1.
We still have to show that in case σ_{φ_2} is an "abstract" configuration itself, then it must be the case that φ_1 holds in all the continuous positions $\sigma_{\varphi_1}@c_{\varphi_1}$ with $c_{\varphi_2} - \tau < c_{\varphi_1} < c_{\varphi_2}$. We can notice that in this case we have $\mathcal{TK}_a^\tau, \sigma_{\varphi_2} \models_p p_a$ and therefore

$$\mathcal{TK}_a^\tau, \sigma_{\varphi_2} \models_p \beta(\varphi_2) \wedge \beta(\varphi_1) \quad \Longleftrightarrow \quad \mathcal{TK}_a^\tau, \sigma_{\varphi_2} \models_p \beta(\varphi_2 \wedge \varphi_1).$$

By induction, we have that $\mathcal{TK}_a^\tau, \sigma_{\varphi_2} \models_c \varphi_2 \wedge \varphi_1$ and by definition of $\models_c$, it follows that $\mathcal{TK}_a^\tau, \sigma_{\varphi_2} \models_c \varphi_1$. We can again use Lemma 3.3 to prove our claim.
Hence, we can conclude that, in this case, $\mathcal{TK}_a^\tau, \sigma \models_c E\ \varphi_1\ U_I\ \varphi_2$.

(ii) $0 \notin I$ and $\beta(E\ \varphi_1\ U_I\ \varphi_2) = E\ \beta(\varphi_1)\ U_I\ (\beta(\varphi_2) \wedge (\neg p_a \vee \beta(\varphi_1)))$. This case is very similar to the "Otherwise" part of case (i).

- "$\Longrightarrow$" case for $\varphi = A\ \varphi_1\ U_I\ \varphi_2$
Assume $\mathcal{TK}_a^\tau, \sigma \models_c A\ \varphi_1\ U_I\ \varphi_2$. By the continuous semantics of TCTL, we have that each time-divergent path $\pi \in tdPaths^c_{\mathcal{TK}_a^\tau}(\sigma)$ satisfies $\varphi_1\ U_I\ \varphi_2$. Then, for each such path, we can take $\pi' \in tdPaths^p_{\mathcal{TK}_a^\tau}(\sigma)$ as defined in Eq. (B.2), and proceed in an analogous way to what we did for the same direction of the existential until case, to show that $\mathcal{TK}_a^\tau, \sigma \models_p \beta(A\ \varphi_1\ U_I\ \varphi_2)$.
- "$\Longleftarrow$" case for $\varphi = A\ \varphi_1\ U_I\ \varphi_2$
Assume $\mathcal{TK}_a^\tau, \sigma \models_p \beta(A\ \varphi_1\ U_I\ \varphi_2)$. By the pointwise semantics of TCTL, we have that each time-divergent path $\pi \in tdPaths^p_{\mathcal{TK}_a^\tau}(\sigma)$ satisfies the "β-transformation" of the until formula $\varphi_1\ U_I\ \varphi_2$, and we can again proceed case-by-case, in an analogous way to what we did for the same direction of the existential until case, to show that for each such path $\mathcal{TK}_a^\tau, \pi \models_c \varphi_1\ U_I\ \varphi_2$, i.e., $\mathcal{TK}_a^\tau, \sigma \models_c A\ \varphi_1\ U_I\ \varphi_2$. □

Theorem 3.3. *Let AP be a set of atomic propositions and $\mathcal{TK}$ a timed Kripke structure over AP whose time domain satisfies the theory $TIME^{gcd}$. Let φ be a TCTL formula over AP, $0 < \tau \in \mathcal{T}$ a time value such that $GCD(\mathcal{TK}, \varphi)$ is a multiple of 2τ and $\mathcal{TK}_a^\tau = (S, \mathcal{T}, \longrightarrow, L)$ the abstract τ-transformation of $\mathcal{TK}$, so that $L : S \to \mathcal{P}(AP_a)$ and $AP_a = AP \uplus \{p_a\}$ are as given in Definition 3.4. Moreover, let $\alpha(\varphi)$ be defined as above, then the following holds for each state (s, δ) in $\mathcal{TK}_a^\tau$ and each subformula φ' of φ:*

$$\mathcal{TK}_a^\tau, (s, \delta) \models_c \varphi' \quad \Longleftrightarrow \quad \mathcal{TK}_a^\tau, (s, \delta) \models_c \alpha(\varphi').$$

Proof. Fig. 13 shows some general patterns of paths that satisfy $\psi = \varphi_1\ U_I\ \varphi_2$ with $I = (2\tau, 6\tau)$ in $\mathcal{TK}_a^\tau$ in the continuous semantics. Again, the π_i start from a concrete configuration (i.e., the corresponding state is labeled with $\neg p_a$), while π_{ia} start from an abstract configuration (i.e., the corresponding state is labeled with p_a). These patterns are the intuition behind the transformation $\alpha(\varphi)$. In all four paths, φ_2 holds in an abstract configuration.

The key observation (which follows from Facts 3.4 and 3.3) is that a path starting from a concrete configuration will reach the (finite) bounds $inf(I)$ and $sup(I)$ at another concrete configuration. Similarly, a path starting from an abstract configuration will reach the above interval bounds at another abstract configuration.

We observe that π_1 and π_2 satisfy both ψ and the TCTL$_{cb}$ path formula $\varphi_1\, U_{I_1}\, \varphi_2$, with $I_1 = [3\tau, 5\tau]$. Intuitively, since all tick transitions in $\mathcal{TK}_a^\tau$ have duration τ, we can safely "shrink" the open-bound interval I to the close-bound interval $I_1 = [inf(I) + \tau, sup(I) - \tau]$, when the initial configuration is concrete. For π_{1a} and π_{2a} notice that they satisfy both ψ and the TCTL$_{cb}$ path formula $\varphi_1\, U_{I_2}\, \varphi_2$, with $I_2 = [2\tau, 6\tau]$. This follows by Lemma 3.3, since the configuration at the φ_2-position is abstract in these two paths.

Let $\mathcal{TK}$, τ, φ, $\mathcal{TK}_a^\tau$ and (s, δ) be as in the theorem and let $\sigma = \langle (s, \delta), 0\rangle$. The proof is given by structural induction on the structure of φ. Note that if $GCD(\mathcal{TK}, \varphi)$ is a multiple of 2τ then also $GCD(\mathcal{TK}, \varphi')$ is a multiple of 2τ for all subformulas φ' of φ.

The case $\varphi = true$ is trivial. For the base case of atomic propositions the equivalence follows from the fact that $L(s, \delta) = L(s, \delta')$ for all states (s, δ), (s, δ') of $\mathcal{TK}_a^\tau$ by the definition of $\mathcal{TK}_a^\tau$, and the fact that $\alpha(\varphi) = \varphi$ in this case.

For the remaining cases, we assume that the equivalence holds on the subformulas of φ.

- $\varphi = \neg\varphi_1$:

$$\begin{aligned} \mathcal{TK}_a^\tau, \sigma \models_c \neg\varphi_1 &\iff not\ (\mathcal{TK}_a^\tau, \sigma \models_c \varphi_1) \\ &\overset{induction}{\iff} not\ (\mathcal{TK}_a^\tau, \sigma \models_c \alpha(\varphi_1)) \\ &\iff \mathcal{TK}_a^\tau, \sigma \models_c \neg\alpha(\varphi_1) \\ &\iff \mathcal{TK}_a^\tau, \sigma \models_c \alpha(\neg\varphi_1) \end{aligned}$$

- $\varphi = \varphi_1 \wedge \varphi_2$:

$$\begin{aligned} \mathcal{TK}_a^\tau, \sigma \models_c \varphi_1 \wedge \varphi_2 &\iff \mathcal{TK}_a^\tau, \sigma \models_c \varphi_1\ and\ \mathcal{TK}_a^\tau, \sigma \models_c \varphi_2 \\ &\overset{induction}{\iff} \mathcal{TK}_a^\tau, \sigma \models_c \alpha(\varphi_1)\ and\ \mathcal{TK}_a^\tau, \sigma \models_c \alpha(\varphi_2) \\ &\iff \mathcal{TK}_a^\tau, \sigma \models_c \alpha(\varphi_1) \wedge \alpha(\varphi_2) \\ &\iff \mathcal{TK}_a^\tau, \sigma \models_c \alpha(\varphi_1 \wedge \varphi_2) \end{aligned}$$

In the following, we assume that $2n_a\tau = inf(I)$ and $2n_b\tau = sup(I)$ for some $n_a, n_b \in \mathbb{N}$. For the cases $\varphi = E\, \varphi_1\, U_I\, \varphi_2$ and $\varphi = A\, \varphi_1\, U_I\, \varphi_2$, we observe that

$$\alpha(E\, \varphi_1\, U_I\, \varphi_2) = \begin{cases} E\, \alpha(\varphi_1)\, U_{I_1}\, \alpha(\varphi_2) & \text{if } p_a \notin L(s, \delta) \\ E\, \alpha(\varphi_1)\, U_{I_2}\, \alpha(\varphi_2) & \text{otherwise,} \end{cases}$$

$$\alpha(A\, \varphi_1\, U_I\, \varphi_2) = \begin{cases} A\, \alpha(\varphi_1)\, U_{I_1}\, \alpha(\varphi_2) & \text{if } p_a \notin L(s, \delta) \\ A\, \alpha(\varphi_1)\, U_{I_2}\, \alpha(\varphi_2) & \text{otherwise.} \end{cases}$$

We can show that for each $\pi \in tdPaths_{\mathcal{TK}_a^\tau}((s, \delta))$

$$\text{(L1)}\quad \mathcal{TK}_a^\tau, \pi \models_c \varphi_1\, U_I\, \varphi_2 \iff \mathcal{TK}_a^\tau, \pi \models_c \alpha(\varphi_1)\, U_{I_1}\, \alpha(\varphi_2) \quad \text{if } p_a \notin L(s, \delta),$$

$$\text{(L2)}\quad \mathcal{TK}_a^\tau, \pi \models_c \varphi_1\, U_I\, \varphi_2 \iff \mathcal{TK}_a^\tau, \pi \models_c \alpha(\varphi_1)\, U_{I_2}\, \alpha(\varphi_2) \quad \text{if } p_a \in L(s, \delta).$$

Moreover, we know that

$$p_a \notin L(s, \delta) \iff \delta = 2n\tau \quad \text{for some } n \in \mathbb{N},$$

$$p_a \in L(s, \delta) \iff \delta = (2n+1)\tau \quad \text{for some } n \in \mathbb{N}.$$

That is, an abstract state, which is labeled with p_a, occurs at an odd multiple of τ, while other states occur at an even multiple of τ.

"(L1)": Notice that $I \supseteq I_1$, therefore showing the "$\Longleftarrow$" direction is trivial and we here show the other direction. By definition of $\models_c$ we have that:

$$\mathcal{TK}_a^\tau, \pi \models_c \varphi_1\, U_I\, \varphi_2 \quad \text{iff} \quad \begin{array}{l} \text{there exists a continuous position } \sigma_{\varphi_2}@c_{\varphi_2} \text{ in } \pi \text{ s.t. } c_{\varphi_2} \in I, \\ \mathcal{TK}_a^\tau, \sigma_{\varphi_2} \models_c \varphi_2 \text{ and } \mathcal{TK}_a^\tau, \sigma_{\varphi_1} \models_c \varphi_1 \text{ for all continuous positions} \\ \sigma_{\varphi_1}@c_{\varphi_1} \text{ in } pre_\pi^c(\sigma_{\varphi_2}@c_{\varphi_2}). \end{array}$$

By induction we have that $\mathcal{TK}_a^\tau, \sigma_{\varphi_2} \models_c \alpha(\varphi_2)$, and $\mathcal{TK}_a^\tau, \sigma_{\varphi_1} \models_c \alpha(\varphi_1)$ for all continuous positions $\sigma_{\varphi_1}@c_{\varphi_1}$ in $pre_\pi^c(\sigma_{\varphi_2}@c_{\varphi_2})$. We can show that π is a path satisfying $\alpha(\varphi_1)\, U_{I_1}\, \alpha(\varphi_2)$. This holds trivially if $c_{\varphi_2} \in I_1$, which includes the case when I is a single point interval. On the other side, if $c_{\varphi_2} \notin I_1$ (i.e., the "duration" of I is larger than τ and $n_a < n_b$), we can notice that the state (s_a, δ_a) corresponding to the continuous position $\langle (s_a, \delta_a), 0\rangle @ 2n_a\tau$ in π occurs, like (s, δ), at an even multiple of τ (i.e. δ_a is an even multiple of τ) and hence the state corresponding to the

continuous position $\langle (s_a^*, \delta_a^*), 0\rangle @ (2n_a+1)\tau$ in π is a p_a-state. Similarly we can observe that the state corresponding to the continuous position $\langle (s_b^*, \delta_b^*), 0\rangle @ (2n_b-1)\tau$ in π is also a p_a-state. We can distinguish two cases:

i) $inf(I) \notin I$ and $2n_a\tau < c_{\varphi_2} < (2n_a+1)\tau = inf(I_1)$: in this case it follows by Lemma 3.3 that $\mathcal{TK}_a^\tau, (s_a^*, \delta_a^*) \models_c \alpha(\varphi_2)$. Moreover, or all continuous positions $\sigma_{\varphi_1} @ c_{\varphi_1}$ in $pre_\pi^c(\sigma_{\varphi_2} @ c_{\varphi_2})$, we have that $\mathcal{TK}_a^\tau, \sigma_{\varphi_1} \models_c \alpha(\varphi_1)$, it follows again by Lemma 3.3 that $\mathcal{TK}_a^\tau, (s_a^*, \delta_a^*) \models_c \alpha(\varphi_1)$ as well, and in particular for all continuous positions $\sigma_{\varphi_1}^* @ c_{\varphi_1}^*$ in π with $c_{\varphi_1}^* < (2n_a+1)\tau$ it holds that $\mathcal{TK}_a^\tau, \sigma_{\varphi_1}^* \models_c \alpha(\varphi_1)$. Thus $\mathcal{TK}_a^\tau, \pi \models_c \alpha(\varphi_1)\, U_{I_1}\, \alpha(\varphi_2)$.

ii) $sup(I) \notin I$ and $sup(I_1) = (2n_b-1)\tau < c_{\varphi_2} < 2n_b\tau$: similarly to the previous case, we can prove by Lemma 3.3 that the state (s_b^*, δ_b^*) corresponding to the continuous position $\langle (s_b^*, \delta_b^*), 0\rangle @ (2n_b-1)\tau$ satisfies $\alpha(\varphi_2)$, then $\mathcal{TK}_a^\tau, \pi \models_c \alpha(\varphi_1)\, U_{I_1}\, \alpha(\varphi_2)$ follows trivially.

"(L2)": Notice that $I \subseteq I_2$, therefore showing the "$\Longrightarrow$" direction is trivial and we here show the other direction. By definition of $\models_c$ we have that:

$$\mathcal{TK}_a^\tau, \pi \models_c \alpha(\varphi_1)\, U_{I_2}\, \alpha(\varphi_2) \quad \text{iff} \quad \text{there exists a continuous position } \sigma_{\varphi_2} @ c_{\varphi_2} \text{ in } \pi \text{ s.t. } c_{\varphi_2} \in I_2, \ \mathcal{TK}_a^\tau, \sigma_{\varphi_2} \models_c \alpha(\varphi_2) \text{ and } \mathcal{TK}_a^\tau, \sigma_{\varphi_1} \models_c \alpha(\varphi_1) \text{ for all continuous positions } \sigma_{\varphi_1} @ c_{\varphi_1} \text{ in } pre_\pi^c(\sigma_{\varphi_2} @ c_{\varphi_2}).$$

By induction we have that $\mathcal{TK}_a^\tau, \sigma_{\varphi_2} \models_c \varphi_2$, and $\mathcal{TK}_a^\tau, \sigma_{\varphi_1} \models_c \varphi_1$ for all continuous positions $\sigma_{\varphi_1} @ c_{\varphi_1}$ in $pre_\pi^c(\sigma_{\varphi_2} @ c_{\varphi_2})$. We can show that π is a path satisfying $\varphi_1\, U_I\, \varphi_2$. This holds trivially if $c_{\varphi_2} \in I$, which includes the case when I is a single point interval. On the other side, if $c_\varphi \notin I$ (i.e., the "duration" of I is larger than τ and $n_a < n_b$), we can notice that the state (s_a, δ_a) corresponding to the continuous position $\langle (s_a, \delta_a), 0\rangle @ 2n_a\tau$ occurs, like (s, δ), at an odd multiple of τ (i.e. δ_a is an odd multiple of τ) and hence it is a p_a-state. Similarly we can observe that the state corresponding to the continuous position $\langle (s_b, \delta_b), 0\rangle @ 2n_b\tau$ is also a p_a-state. We can again distinguish two cases:

i) $inf(I) \notin I$ and $inf(I) = inf(I_2) = 2n_a\tau = c_{\varphi_2}$: in this case it follows by Lemma 3.3 that for all continuous positions $\sigma_a^* @ c_a^*$ with $(2n_a-1)\tau < c_a^* < (2n_a+1)\tau$ it holds that $\mathcal{TK}_a^\tau, \sigma_a^* \models_c \alpha(\varphi_2)$ and hence, by induction, $\mathcal{TK}_a^\tau, \sigma_a^* \models_c \varphi_2$. In particular, there exists a continuous position $\sigma^* @ c^*$ with $2n_a\tau < c^* < (2n_a+1)\tau$, and hence $c^* \in I$, satisfying φ_2. Moreover, since for all continuous positions $\sigma_{\varphi_1} @ c_{\varphi_1}$ in $pre_\pi^c(\sigma_{\varphi_2} @ c_{\varphi_2})$, we have that $\mathcal{TK}_a^\tau, \sigma_{\varphi_1} @ c_{\varphi_1} \models_c \alpha(\varphi_1)$, it follows again by Lemma 3.3 that $\mathcal{TK}_a^\tau, \sigma_a^* \models_c \varphi_1$ as well, and in particular for all continuous positions $\sigma_1^* @ c_1^*$ in π with $(2n_a-1)\tau < c_1^* < c^*$ it holds that $\mathcal{TK}_a^\tau, \sigma_1^* \models_c \alpha(\varphi_1)$, and hence, by induction, $\mathcal{TK}_a^\tau, \sigma_1^* \models_c \varphi_1$. Thus $\mathcal{TK}_a^\tau, \pi \models_c \varphi_1\, U_I\, \varphi_2$.

ii) $sup(I) \notin I$ and $sup(I) = sup(I_2) = 2n_b\tau = c_{\varphi_2}$: the proof is very similar to the previous case. □

Theorem 4.2. *Given a finite Zeno-free timed Kripke structure $\mathcal{TK} = (S, \mathcal{T}, \longrightarrow, L)$ and a TCTL formula φ in normal form, then*

$$Sat(\varphi) = \{s \in S \mid \mathcal{TK}, s \models_p \varphi\} = \mathsf{Sat}(\mathcal{TK}, \varphi).$$

Proof. Theorem 4.1 implies that finiteness of $\mathcal{TK}$ and the assumption of time-divergent paths are sufficient to guarantee termination of $\mathsf{Sat}(\mathcal{TK}, \varphi)$. Notice that proving the theorem is equivalent to show that:

$$\forall s \in S.\ (\mathcal{TK}, s \models_p \varphi \iff s \in \mathsf{Sat}(\mathcal{TK}, \varphi))$$

Since the valid configurations of $\mathcal{TK}$ in the pointwise interpretation are of the kind $\langle s, 0\rangle$ with $s \in S$, a path π starting in s_0 can be also written as a sequence of transitions:

$$s_0 \xrightarrow{\tau_0} s_1 \xrightarrow{\tau_1} s_2 \xrightarrow{\tau_2} \cdots.$$

Furthermore, we can restate the TCTL pointwise semantics in terms of states and integer positions on a path:

$\mathcal{TK}, s \models_p true$		always hold
$\mathcal{TK}, s \models_p p$	iff	$p \in L(s)$
$\mathcal{TK}, s \models_p \neg\varphi_1$	iff	$\mathcal{TK}, s \not\models_p \varphi_1$
$\mathcal{TK}, s \models_p \varphi_1 \wedge \varphi_2$	iff	$\mathcal{TK}, s \models_p \varphi_1$ and $\mathcal{TK}, s \models_p \varphi_2$
$\mathcal{TK}, s \models_p E\, \varphi_1\, U_I\, \varphi_2$	iff	there exist $\pi \in tdPaths^p_{\mathcal{TK}}(s)$ and an index j s.t. $c_j^\pi \in I$, $\mathcal{TK}, s_j^\pi \models_p \varphi_2$, and $\forall\, 0 \le i < j\ \ \mathcal{TK}, s_i^\pi \models_p \varphi_1$.
$\mathcal{TK}, s \models_p A\, \varphi_1\, U_I\, \varphi_2$	iff	for each $\pi \in tdPaths^p_{\mathcal{TK}}(s)$, there exists an index j s.t. $c_j^\pi \in I$, $\mathcal{TK}, s_j^\pi \models_p \varphi_2$, and $\forall\, 0 \le i < j\ \ \mathcal{TK}, s_i^\pi \models_p \varphi_1$.

The above definition of $\models_p$ and notation for a path are used in the proof, which is by induction on the normal form structure of φ.

- $\varphi = true$: $Sat(true) = S = \mathsf{Sat}(\mathcal{TK}, true)$ by definition of $\models_p$ and $\mathsf{Sat}(\mathcal{TK}, \varphi)$.
- $\varphi = p$: $Sat(p) = \{s \mid p \in L(s)\} = \mathsf{Sat}(\mathcal{TK}, p)$ by definition of $\models_p$ and $\mathsf{Sat}(\mathcal{TK}, \varphi)$.

Assume by induction hypothesis that the theorem holds for the subformulas φ_1 and φ_2.

- $\varphi = \neg\varphi_1$:

$$\begin{aligned} Sat(\neg\varphi_1) &= S \setminus Sat(\varphi_1) \\ &\overset{induction}{=} S \setminus \mathsf{Sat}(\mathcal{TK}, \varphi_1) \\ &= \mathsf{Sat}(\mathcal{TK}, \neg\varphi_1) \end{aligned}$$

- $\varphi = \varphi_1 \wedge \varphi_2$:

$$\begin{aligned} Sat(\varphi_1 \wedge \varphi_2) &= Sat(\varphi_1) \cap Sat(\varphi_2) \\ &\overset{induction}{=} \mathsf{Sat}(\mathcal{TK}, \varphi_1) \cap \mathsf{Sat}(\mathcal{TK}, \varphi_2) \\ &= \mathsf{Sat}(\mathcal{TK}, \varphi_1 \wedge \varphi_2) \end{aligned}$$

- $\varphi = \varphi_1 \vee \varphi_2$:

$$\begin{aligned} Sat(\varphi_1 \vee \varphi_2) &= Sat(\varphi_1) \cup Sat(\varphi_2) \\ &\overset{induction}{=} \mathsf{Sat}(\mathcal{TK}, \varphi_1) \cup \mathsf{Sat}(\mathcal{TK}, \varphi_2) \\ &= \mathsf{Sat}(\mathcal{TK}, \varphi_1 \vee \varphi_2) \end{aligned}$$

- $\varphi = E\ \varphi_1\ U\ \varphi_2$:
  In this case, we have that $\mathsf{Sat}(\mathcal{TK}, \varphi) = \mathsf{Sat\text{-}EU}(\mathcal{TK}, \varphi)$. We show that both $Sat(\varphi) \supseteq \mathsf{Sat\text{-}EU}(\mathcal{TK}, \varphi)$ and $Sat(\varphi) \subseteq \mathsf{Sat\text{-}EU}(\mathcal{TK}, \varphi)$ hold.
  "$\supseteq$": We prove the while loop invariant $\mathsf{Q} \subseteq Sat(\varphi)$.
  - loop entry: $\mathsf{Q} = \mathsf{Sat}(\mathcal{TK}, \varphi_2) \overset{induction}{=} Sat(\varphi_2) \subseteq Sat(\varphi)$, since, by definition of $\models_p$, each φ_2-state is also a $(E\ \varphi_1\ U\ \varphi_2)$-state. In particular, for each $s_2 \in Sat(\varphi_2)$, any $\pi \in tdPaths^p_{\mathcal{TK}}(s_2)$ is a $(\varphi_1\ U\ \varphi_2)$-path.
  - while iteration: Before Q is assigned, the invariant $\mathsf{Q} \subseteq Sat(\varphi)$ holds and $\mathsf{Q}_{\mathsf{pre}} \neq \emptyset$. We must prove that

$$\mathsf{Q} \cup (\mathsf{Q}_{\mathsf{pre}} = \{s' \in \mathsf{Q}_1 \setminus \mathsf{Q} \mid \exists s \in \mathsf{Q}, \tau \in \mathcal{T} : s' \xrightarrow{\tau} s \in \longrightarrow\}) \subseteq Sat(\varphi).$$

    Let $s' \in \mathsf{Q}_{\mathsf{pre}}$, and let $s \in \mathsf{Q}$ and $\tau \in \mathcal{T}$ be those state and time such that $s' \in \mathsf{Q}_{\mathsf{pre}}$. By induction on the structure of φ we have that

$$s' \in \mathsf{Q}_1 = \mathsf{Sat}(\mathcal{TK}, \varphi_1) \iff s' \in Sat(\varphi_1).$$

    By definition of $\models_p$ and the invariant $\mathsf{Q} \subseteq Sat(\varphi)$ (holding before the assignment), we have that

$$\begin{aligned} s \in \mathsf{Q} \subseteq Sat(\varphi) \iff\ & \text{there exist } \pi \in tdPaths^p_{\mathcal{TK}}(s) \text{ and an index } j \text{ s.t. } \mathcal{TK}, s^\pi_j \models_p \varphi_2, \\ & \text{and } \forall\, 0 \leq i < j\ \ \mathcal{TK}, s^\pi_i \models_p \varphi_1. \end{aligned}$$

    Let $\pi = s \xrightarrow{\tau_0} s_1 \xrightarrow{\tau_1} \cdots \xrightarrow{\tau_{j-1}} s_j \xrightarrow{\tau_j} \cdots$ be such a path and j such an index. By taking

$$\pi' = s' \xrightarrow{\tau} s \xrightarrow{\tau_0} s_1 \xrightarrow{\tau_1} \cdots \xrightarrow{\tau_{j-1}} s_j \xrightarrow{\tau_j} \cdots$$

    it follows that $\mathcal{TK}, s' \models_p \varphi$, and hence $s' \in \mathsf{Q}_{\mathsf{pre}} \Longrightarrow s \in Sat(\varphi)$ (or equivalently $\mathsf{Q}_{\mathsf{pre}} \subseteq Sat(\varphi)$) and hence $\mathsf{Q} \cup \mathsf{Q}_{\mathsf{pre}} \subseteq Sat(\varphi)$.
  - loop exit: The set $\mathsf{Q}_{\mathsf{pre}}$ is empty and Q is returned.

  Thus, we proved that $\mathsf{Sat}(\mathcal{TK}, \varphi) \subseteq Sat(\varphi)$. For the "$\subseteq$" direction, let $\mathsf{Q}^0, \mathsf{Q}^1, \ldots, \mathsf{Q}^k$ be, respectively, the set Q initially, after one "while loop" iteration, and so on, such that $\mathsf{Q}^k = \mathsf{Sat}(\mathcal{TK}, \varphi)$ is the least fixed-point of such iterations.[25]
  "$\subseteq$": Let $s \in Sat(\varphi)$. By the definition of $\models_p$ it holds that

$$\begin{aligned} \mathcal{TK}, s \models_p \varphi \iff\ & \text{there exist } \pi \in tdPaths^p_{\mathcal{TK}}(s) \text{ and an index } j \text{ s.t. } \mathcal{TK}, s^\pi_j \models_p \varphi_2, \\ & \text{and } \forall\, 0 \leq i < j\ \ \mathcal{TK}, s^\pi_i \models_p \varphi_1. \end{aligned}$$

  Let $\pi = s_0 \xrightarrow{\tau_0} s_1 \xrightarrow{\tau_1} \cdots \xrightarrow{\tau_{j-1}} s_j \xrightarrow{\tau_j} \cdots$ be such a path and j such an index. We show that for all $0 \leq i \leq j$, s_{j-i} belongs to Q^i.
  - "$i = 0$": We have $s_j \models_p \varphi_2$ which implies by structural induction $s_j \in \mathsf{Sat}(\mathcal{TK}, \varphi_2)$, and hence $s_j \in \mathsf{Q}^0$. If $j = 0$, we are done.
  - "$i = n + 1$" and $j > 0$: By definition of $\models_p$ it holds that $\mathcal{TK}, s_{j-(n+1)} \models_p \varphi_1$, and hence it follows by structural induction that $s_{j-(n+1)} \in \mathsf{Sat}(\mathcal{TK}, \varphi_1) = \mathsf{Q}_1$. We can assume that the property holds for $i = n$, that is $s_{j-n} \in \mathsf{Q}^n$. We know that there is a transition $s_{j-(n+1)} \xrightarrow{\tau} s_{j-n}$, for some $\tau \in \mathcal{T}$. Then at the $(n+1)$th iteration of the while loop we have that $s_{j-(n+1)}$ is a predecessor of $s_{j-n} \in \mathsf{Q}^n$, through the transition $s_{j-(n+1)} \xrightarrow{\tau} s_{j-n}$. We can distinguish two cases: either $s_{j-(n+1)} \in \mathsf{Q}^n$ or $s_{j-(n+1)} \notin \mathsf{Q}^n$, in this second case we have that $s_{j-(n+1)} \in \mathsf{Q}_{\mathsf{pre}}$ and hence $s_{j-(n+1)} \in \mathsf{Q}^{n+1}$. In particular, we proved that $s = s_0 \in \mathsf{Sat}(\mathcal{TK}, \varphi)$ and hence $Sat(\varphi) \subseteq \mathsf{Sat}(\mathcal{TK}, \varphi)$.

[25] Kleene's fixed-point theorem guarantees that a least fixed-point exists and can be computed by the $\mathsf{Q}^0, \mathsf{Q}^1, \ldots, \mathsf{Q}^k$ iterates.

- $\varphi = E\ G\ \varphi_1$:
In this case, we have that $\mathsf{Sat}(\mathcal{TK}, \varphi) = \mathsf{Sat\text{-}EG}(\mathcal{TK}, \varphi)$. We can show that both $Sat(\varphi) \supseteq \mathsf{Sat\text{-}EG}(\mathcal{TK}, \varphi)$ and $Sat(\varphi) \subseteq \mathsf{Sat\text{-}EG}(\mathcal{TK}, \varphi)$ hold. Let $\mathsf{Q}^0, \mathsf{Q}^1, \ldots, \mathsf{Q}^k$ be as for the proof case $\varphi = E\ \varphi_1\ U\ \varphi_2$, so that $\mathsf{Q}^0 = \mathsf{Tarjan}(\mathsf{Q}_1, \longrightarrow)$.
"$\supseteq$": It is easy to check the while loop invariant $\mathsf{Q} \subseteq Sat(\varphi)$. Initially we have that $\mathsf{Q} = \mathsf{Tarjan}(\mathsf{Q}_1, \longrightarrow)$, that is the set of non-trivial strongly connected components of φ_1-states, for which there surely exists an infinite path of φ_1-states. A proof similar to the one carried out in the proof case $\varphi = E\ \varphi_1\ U\ \varphi_2$ can easily show that the invariant holds also after each while loop iteration and when the exit condition is true.
"$\subseteq$": The proof is similar to the one for the respective direction of the case $\varphi = E\ \varphi_1\ U\ \varphi_2$. Let $s \in Sat(\varphi)$, by definition of $\models_p$ it holds that

$$\mathcal{TK}, s \models_p \varphi \quad \Longleftrightarrow \quad \text{there exists } \pi \in tdPaths^p_{\mathcal{TK}}(s) \text{ s.t. for each index } k\text{: } \mathcal{TK}, s^\pi_k \models_p \varphi_1.$$

Let $\pi = s_0 \xrightarrow{\tau_0} s_1 \xrightarrow{\tau_1} \cdots$ be such a path. Since we assumed that S is finite, it must be the case that there exists an index j s.t. $s^\pi_j \in SCC(\varphi_1)$. Let j be the smallest of such indexes, we can show that $0 \leq i \leq j$, s^π_{j-i} belongs to Q^i.
  - "$i = 0$": By construction of π, we have that $s^\pi_j \in \mathsf{Q}^0 = \mathsf{Tarjan}(\mathsf{Q}_1, \longrightarrow)$. If $j = 0$ we are done.
  - "$i = n + 1$" and $j > 0$: By definition of $\models_p$ it holds that $\mathcal{TK}, s^\pi_{j-(n+1)} \models_p \varphi_1$, and hence it follows by structural induction that $s^\pi_{j-(n+1)} \in \mathsf{Sat}(\mathcal{TK}, \varphi_1) = \mathsf{Q}_1$. We can assume that the property holds for $i = n$, that is $s^\pi_{j-n} \in \mathsf{Q}^n$. We know that there is a transition $s^\pi_{j-(n+1)} \xrightarrow{\tau} s^\pi_{j-n}$, for some $\tau \in \mathcal{T}$. Then at the $(n+1)$th iteration of the while loop we have that $s^\pi_{j-(n+1)}$ is a predecessor of $s^\pi_{j-n} \in \mathsf{Q}^n$, through the transition $s^\pi_{j-(n+1)} \xrightarrow{\tau} s^\pi_{j-n}$. We can distinguish two cases: either $s^\pi_{j-(n+1)} \in \mathsf{Q}^n$ or $s^\pi_{j-(n+1)} \notin \mathsf{Q}^n$, in this second case we have that $s^\pi_{j-(n+1)} \in \mathsf{Q}_{\mathsf{pre}}$ and hence $s^\pi_{j-(n+1)} \in \mathsf{Q}^{n+1}$. In particular, we proved that $s = s_0 \in \mathsf{Sat}(\mathcal{TK}, \varphi)$ and hence $Sat(\varphi) \subseteq \mathsf{Sat}(\mathcal{TK}, \varphi)$.
- $\varphi = E\ \varphi_1\ U_I\ \varphi_2$, with $inf(I) = 0 \in I$, $b = sup(I) \neq \infty$ and $\sim = (\text{if } b \in I \text{ then } \leq \text{ else } <)$:
In this case, we have that $\mathsf{Sat}(\mathcal{TK}, \varphi) = \mathsf{Sat\text{-}EU_b}(\mathcal{TK}, \varphi)$. We show that both $Sat(\varphi) \supseteq \mathsf{Sat\text{-}EU_b}(\mathcal{TK}, \varphi)$ and $Sat(\varphi) \subseteq \mathsf{Sat\text{-}EU_b}(\mathcal{TK}, \varphi)$ hold. By structural induction we have that $\mathsf{Q}_1 = \mathsf{Sat}(\mathcal{TK}, \varphi_1) = Sat(\varphi_1)$ and $\mathsf{Q}_2 = \mathsf{Sat}(\mathcal{TK}, \varphi_2) = Sat(\varphi_2)$.
Furthermore, we have already proven that $\mathsf{Q}_u = \mathsf{Sat}(\mathcal{TK}, E\ \varphi_1\ U\ \varphi_2) = Sat(E\ \varphi_1\ U\ \varphi_2)$.
Since $Sat(\varphi) \subseteq Sat(E\ \varphi_1\ U\ \varphi_2)$, we can safely restrict to the subgraph $G = (\mathsf{Q}_u, \mathsf{TR})$ for discovering the states satisfying φ. Notice that by setting T to be initially the set of φ_2-states with distance zero, and TR the set of outgoing transitions from $(\varphi_1 \wedge \neg\varphi_2)$-states satisfying $E\ \varphi_1\ U\ \varphi_2$, it is guaranteed that the shortest path computation following the Dijkstra's approach will eventually discover all the possible shortest distances (up to the bound b) from each state in Q_u to some φ_2-state in Q_u. Therefore, all the state with such distance will eventually be added to Q.
"$\supseteq$": We show that $\mathsf{Q} \subseteq Sat(\varphi)$ is an invariant of the while loop. This trivially holds before the while loop. At each iteration the state with the minimal distance in T is added to Q. Dijkstra algorithm guarantees that this local minimum is indeed a global minimum and hence its distance was the shortest possible one to some φ_2-state. Moreover the check $(\tau + \tau' \sim b)$, guarantees that this shortest distance is within the bound b. Therefore we can conclude that there exists a path from s to some φ_2 state with total duration within the time bound b and hence $\mathsf{Q} \cup \{s\} \subseteq Sat(\varphi)$.
"$\subseteq$": Let $s \in Sat(\varphi)$. By definition of $\models_p$ it holds that

$$\mathcal{TK}, s \models_p \varphi \quad \Longleftrightarrow \quad \begin{array}{l}\text{there exist } \pi \in tdPaths^p_{\mathcal{TK}}(s) \text{ and an index } j \text{ s.t. } c^\pi_j \sim b,\ \mathcal{TK}, s^\pi_j \models_p \varphi_2,\\ \text{and } \forall\, 0 \leq i < j\ \ \mathcal{TK}, s^\pi_i \models_p \varphi_1.\end{array}$$

Let $\pi = s \xrightarrow{\tau_0} s_1 \xrightarrow{\tau_1} \cdots \xrightarrow{\tau_{j-1}} s_j \xrightarrow{\tau_j} \cdots$ be the shortest of such paths, with j such an index. Dijkstra's algorithm ensures that s is eventually added to Q and hence $Sat(\varphi) \subseteq \mathsf{Sat}(\mathcal{TK}, \varphi)$.
- $\varphi = E\ \varphi_1\ U_I\ \varphi_2$, with $a = inf(I)$, $sup(I) = \infty$ and $0 \notin I$:
In this case, we have that $\mathsf{Sat}(\mathcal{TK}, \varphi) = \mathsf{Sat\text{-}EU_a}(\mathcal{TK}, \varphi)$. Since, as explained in Section 4.2, this is a two-step procedure, we also discuss the proof in two parts. Again, by structural induction we have that $\mathsf{Q}_1 = \mathsf{Sat}(\mathcal{TK}, \varphi_1) = Sat(\varphi_1)$ and we have already proven that $\mathsf{Q}_u = \mathsf{Sat}(\mathcal{TK}, E\ \varphi_1\ U\ \varphi_2) = Sat(E\ \varphi_1\ U\ \varphi_2)$.
The first part of the procedure, which ends with the first while loop, performs operations that are equivalent to computing $Sat(E\ \varphi_1\ U$ ("$SCC(\varphi_1)$-state" $\wedge\ E\ \varphi_1\ U\ \varphi_2)$), since we have already proven that the satisfaction set for this formula can be correctly computed by the respective Sat call, we can safely assume that all states satisfying φ as described in point (a) of the description of Procedure 6 (Section 4.2, page 30) are correctly computed.
For the second part, we can notice that the computation of the time distances relies on the well-known topologically ordered visit of a directed acyclic graph, which guarantees that the computed distances are the longest possible. With a reasoning similar to the one for the respective direction of the previous proof case, we can conclude that all states satisfying φ as described in point b in the description of Procedure 6 (Section 4.2, page 30) are correctly computed.
- $\varphi = E\ \varphi_1\ U_I\ \varphi_2$, with $a = inf(I)$, $b = sup(I) < \infty$ and $0 \notin I$:
In this case, we have that $\mathsf{Sat}(\mathcal{TK}, \varphi) = \mathsf{Sat\text{-}EU_{a,b}}(\mathcal{TK}, \varphi)$. As before, we can observe that by structural induction we have that $\mathsf{Q}_1 = \mathsf{Sat}(\mathcal{TK}, \varphi_1) = Sat(\varphi_1)$, $\mathsf{Q}_2 = \mathsf{Sat}(\mathcal{TK}, \varphi_1) = Sat(\varphi_1)$ and we have already proven that $\mathsf{Q}_u = \mathsf{Sat}(\mathcal{TK}, E\ \varphi_1\ U\ \varphi_2) = Sat(E\ \varphi_1\ U\ \varphi_2)$. The procedure $\mathsf{Sat\text{-}EU_{a,b}}$ is similar to $\mathsf{Sat\text{-}EU_b}$, with these differences:

- TR is here initialized with the set of all outgoing transitions from some φ_1-state in $\mathsf{Q_u}$; this allows us to compute all the possible distances (within the time bound b) between φ_1-states and φ_2-states in the restricted graph $G = (\mathsf{Q_u}, \mathsf{TR})$;
- at each while iteration, we can pick *any* of the distance pair (s, τ) in T to perform the backward visit;
- at each while iteration, no transition is removed from TR;
- the chosen state s is added to Q only if its distance τ is higher than the time bound a.

These key points guarantees that at termination Q will contain the states s for which $\mathcal{TK}, s \models_p \varphi$ and no more, since by construction it is a least fixed-point.

- $\varphi = A\ \varphi_1\ U_I\ \varphi_2$, with $inf(I) = 0 \notin I$:
In this case, we have that $\mathsf{Sat}(\mathcal{TK}, \varphi) = \mathsf{Sat\text{-}AU_0}(\mathcal{TK}, \varphi)$. As before, by structural induction we have that $\mathsf{Q_1} = \mathsf{Sat}(\mathcal{TK}, \varphi_1) = Sat(\varphi_1)$. Recall that I_0 is the interval I extended with the time 0, i.e., $I_0 = I \cup \{0\}$. We can observe that $\mathsf{Sat}(\mathcal{TK}, (A\ \varphi_1\ U_{I_0}\ \varphi_2)!)$ correctly computes $Sat(A\ \varphi_1\ U_{I_0}\ \varphi_2)$, since

$$(A\ \varphi_1\ U_{I_0}\ \varphi_2)! = \begin{cases} \neg(E\ G\ \neg\varphi_2) \wedge \neg(E\ \neg\varphi_2\ U\ \neg\varphi_1 \wedge \neg\varphi_2) & \text{if } I_0 = [0, \infty), \\ \neg(E\ \neg\varphi_2\ U_{>b}\ true) \wedge \neg(E\ \neg\varphi_2\ U\ \neg\varphi_1 \wedge \neg\varphi_2) & \text{if } I_0 = [0, b], \\ \neg(E\ \neg\varphi_2\ U_{\geq b}\ true) \wedge \neg(E\ \neg\varphi_2\ U\ \neg\varphi_1 \wedge \neg\varphi_2) & \text{if } I_0 = [0, b). \end{cases}$$

Each of the possible temporal modalities in the normal form have already been proven correct.
We can easily show that $Sat(\varphi) \subseteq Sat(A\ \varphi_1\ U_{I_0}\ \varphi_2) = \mathsf{Q_u}$. By definition of $\models_p$, we have that

$$\mathcal{TK}, s \models_p \varphi \iff \text{for each } \pi \in tdPaths^p_{\mathcal{TK}}(s) \text{ there is an index } j \text{ s.t. } c^\pi_j \in I,\ \mathcal{TK}, s^\pi_j \models_p \varphi_2, \text{ and } \mathcal{TK}, s^\pi_i \models_p \varphi_1 \text{ for all } 0 \leq i < j$$

and it must be the case that $c^\pi_j > 0$. Formula $A\ \varphi_1\ U_{I_0}\ \varphi_2$ relaxes this condition to $c^\pi_j \geq 0$, and therefore we have that $Sat(\varphi) \subseteq Sat(A\ \varphi_1\ U_{I_0}\ \varphi_2) = \mathsf{Q_u}$. If we compute the set Q of $\neg\varphi$-states in $\mathsf{Q_u}$, then we have that $Sat(\varphi) = \mathsf{Q_u} \setminus Q$.
The satisfaction set of $A\ \varphi_1\ U_{I_0}\ \varphi_2$ consists of $Sat(\varphi)$ together with those $\neg\varphi$-states for which there exists a $(\varphi_1\ U\ \varphi_2)$-path of "duration strictly equal to zero." In order to clarify the "nature" of the $\neg\varphi$-states in $A\ \varphi_1\ U_{I_0}\ \varphi_2$, let us consider $\mathcal{TK}$ as a graph, and let us take the subgraph induced by the subset of states $Sat(A\ \varphi_1\ U_{I_0}\ \varphi_2)$ and all the transitions between these states, excluding the outgoing transitions from $\neg(\varphi_1)$-states, that is $G = (\mathsf{Q_u}, \mathsf{TR})$. For the sake of simplicity, let us call a finite path a "path fragment." Then we have that $\neg\varphi$-states appear in G on some path fragment of the following form:

$$\underbrace{\underbrace{s_0 \xrightarrow{0} s_1 \xrightarrow{0} \cdots \xrightarrow{0}}_{\varphi_1}\ \underbrace{s_j}_{\varphi_2}}_{\neg\varphi}$$

where s_j is a states that has no outgoing transitions in G. We have that Q is initialized exactly with these s_j states, and at each while iteration the set $\mathsf{Q_{pre}}$ of states (not already in Q) that can reach a Q-state in time zero is added to Q. It is easy to show that the after the "while loop" iterations, Q contains the least fixed-point set of $\neg\varphi$-states in $\mathsf{Q_u}$. □

Lemma 6.1. *Assume a time-robust real-time rewrite theory $\mathcal{R}$ whose time domain satisfies the theory $TIME^{gcd}_\infty$. Let AP be a set of atomic propositions defined in an extension $(\Sigma \cup AP, E \cup D) \supseteq (\Sigma, E)$, with a* labeling function $L_{AP} : \mathbb{T}_{\Sigma/(E \cup A), \texttt{GlobalSystem}} \to \mathcal{P}(AP)$, *such that a subset of atomic propositions $AP' \subseteq AP$ is tick-invariant in $\mathcal{R}$. Let $r \neq 0$ be a term of sort* `Time`, *and φ a TCTL formula over AP'. Then*

$$\mathcal{R}, L_{AP'}, t \models_c \varphi \iff \mathcal{R}^{maxDef(r)}, L_{AP'}, t \models_c \varphi$$

for each state t in $\mathcal{R}$.

Proof. By Definition 6.2, it follows that proving the theorem is equivalent to showing

$$\mathcal{TK}(\mathcal{R}, AP'), t \models_c \varphi \iff \mathcal{TK}(\mathcal{R}^{maxDef(r)}, AP'), t \models_c \varphi.$$

Let $\mathcal{TK}$ denote $\mathcal{TK}(\mathcal{R}, AP')$, and let $\mathcal{TK}^{max}$ denote $\mathcal{TK}(\mathcal{R}^{maxDef(r)}, AP')$.

By Definition 6.2, each one-step rewrite $t \xrightarrow{r'} t'$ in $\mathcal{R}$ has a corresponding transition $t \xrightarrow{r'} t'$ in $\mathcal{TK}$, and similarly each one-step rewrite $t \xrightarrow{r'} t'$ in $\mathcal{R}^{maxDef(r)}$ has a corresponding transition $t \xrightarrow{r'} t'$ in $\mathcal{TK}^{max}$.

By Fact 5.2 (which holds because $\mathcal{R}$ is time-robust), for each sequence of non-maximal tick steps followed by a maximal tick step $t \xrightarrow{r_0} t_1 \xrightarrow{r_1} t_2 \xrightarrow{r_2} \cdots \xrightarrow{r_{k-1}} t_k \xrightarrow{r_k} t'$ in $\mathcal{R}$ there exists a maximal tick step $t \xrightarrow{r'} t'$ in $\mathcal{R}$ such that $r' = \sum_{i=0}^{k} r_i$. By definition of $\mathcal{R}^{maxDef(r)}$, such maximal tick step $t \xrightarrow{r'} t'$ is also a tick step in $\mathcal{R}^{maxDef(r)}$.

By Definition 5.1 and by axiom TR5 of time-robustness, the existence of an ∞ tick step $t \xrightarrow{r'} t'$ in $\mathcal{R}$ implies that time can advance beyond any limit in t and no other behaviors are possible from t. Furthermore, by definition of $\mathcal{R}^{maxDef(r)}$, for

each ∞ tick step $t \xrightarrow{r'} t'$, with $r' = kr + r''$ for some $k \in \mathbb{N}$ and $r'' < r$, in $\mathcal{R}$ there exists a sequence of $k+1$ tick steps of default duration r in $\mathcal{R}^{maxDef(r)}$ (which are also ∞ tick step in $\mathcal{R}$).

We define the relation $\mathcal{R}_{\mathcal{C}} \subseteq \mathcal{C}^c_{\mathcal{TK}} \times \mathcal{C}^c_{\mathcal{TK}^{max}}$ between valid configurations in $\mathcal{TK}$ and valid configurations in $\mathcal{TK}^{max}$ in the following way:

- for each instantaneous step $t \xrightarrow{inst} t'$ in $\mathcal{R}$ we define
  - $\langle t, 0\rangle \mathcal{R}_{\mathcal{C}} \langle t, 0\rangle$, and
  - $\langle t', 0\rangle \mathcal{R}_{\mathcal{C}} \langle t', 0\rangle$; and
- for each sequence non-maximal tick steps followed by a maximal tick step $t \xrightarrow{r_0} t_1 \xrightarrow{r_1} t_2 \xrightarrow{r_2} \cdots \xrightarrow{r_{k-1}} t_k \xrightarrow{r_k} t'$ in $\mathcal{R}$ corresponding to the tick step $t \xrightarrow{r'} t'$ in $\mathcal{R}^{maxDef(r)}$ we define
  - $\langle t, \delta\rangle \mathcal{R}_{\mathcal{C}} \langle t, \delta\rangle$, for $0 \le \delta < r_0$;
  - $\langle t_i, \delta\rangle \mathcal{R}_{\mathcal{C}} \langle t, \bar{r} + \delta\rangle$, with $\bar{r} = \sum_{j=0}^{i-1} r_i$, for $0 \le \delta < r_i$; and
  - $\langle t', 0\rangle \mathcal{R}_{\mathcal{C}} \langle t', 0\rangle$.
- Furthermore, for each ∞ tick step $t_0 \xrightarrow{r'} t'$, with $r' = kr + r''$ for some $k \in \mathbb{N}$ and $r'' < r$, in $\mathcal{R}$ corresponding to the sequence of tick steps $t_0 \xrightarrow{r} t_1 \xrightarrow{r} t_2 \xrightarrow{r} \cdots \xrightarrow{r} t_k \xrightarrow{r} t_{k+1}$ in $\mathcal{R}^{maxDef(r)}$, we define
  - $\langle t_0, i \cdot r + \delta\rangle \mathcal{R}_{\mathcal{C}} \langle t_i, \delta\rangle$, for $0 \le i < k$ and $0 \le \delta < r$; and
  - $\langle t_0, k \cdot r + \delta\rangle \mathcal{R}_{\mathcal{C}} \langle t_k, \delta\rangle$, for $0 \le \delta < r''$.

By Definition 5.1, for each non-maximal tick step $t \xrightarrow{r'} t'$ in $\mathcal{R}$ there exists a maximal tick step $t \xrightarrow{r''} t''$, such that $r'' > r'$, and hence there exists a time $r''' > 0$ such that $r' + r''' = r''$. Thus the maximal tick step $t \xrightarrow{r''} t''$ can be analogously written as $t \xrightarrow{r'+r'''} t''$, and axiom TR2 implies that there exists a t''' such that $t' \xrightarrow{r'''} t'''$ is a maximal tick step. Therefore, each non-maximal tick step in $\mathcal{R}$ appears on some sequence of non-maximal tick steps followed by a maximal tick step in $\mathcal{R}$. Each maximal tick step in $\mathcal{R}$ also appears in some such sequence (consisting of this maximal tick step only). These conditions, together with the fact that, by time-robustness of $\mathcal{R}$, each one-step rewrite using a tick rule is either a maximal, a non-maximal, or an ∞ tick step, guarantees that the relation $\mathcal{R}_{\mathcal{C}}$ is total, since it is defined for all the possible valid configurations $\mathcal{C}^c_{\mathcal{TK}}$.

Furthermore, because of tick-invariance it holds that $\sigma \mathcal{R}_{\mathcal{C}} \sigma'$ implies that σ and σ' satisfy the same atomic propositions in AP'.

We prove the following stronger equivalence that holds for any configuration $\sigma \in \mathcal{C}^c_{\mathcal{TK}}$:

$$\mathcal{TK}(\mathcal{R}, AP'), \sigma \models_c \varphi \quad \Longleftrightarrow \quad \mathcal{TK}(\mathcal{R}^{maxDef(r)}, AP'), \sigma^{max} \models_c \varphi,$$

where $\sigma \mathcal{R}_{\mathcal{C}} \sigma^{max}$. Notice that if $\sigma = \langle t, 0\rangle$, with t a state in $\mathcal{R}$ then $\sigma^{max} = \sigma$.

The proof of the above stronger equivalence is very similar to the proof for Lemma 3.1. We define an equivalence relation $R \subseteq tdPaths_{\mathcal{TK}}(\sigma) \times tdPaths_{\mathcal{TK}}(\sigma)$ on the time-divergent paths of $\mathcal{TK}$ starting in the configuration σ in the continuous semantics such that two paths are equivalent iff they have the same set of continuous positions. Trivially, equivalent paths satisfy the same path formulas. For each equivalence class we define as representative element a path π that has only maximal steps, i.e., a path of the form

$$\pi = \sigma \overset{r_1}{\hookrightarrow} \sigma_1 \overset{r_2}{\hookrightarrow} \sigma_2 \overset{r_3}{\hookrightarrow} \cdots$$

such that each $\sigma_i = \langle t, 0\rangle$ for some state t in $\mathcal{TK}$. Such a representative element exists for each equivalence class and is unique. $[\pi]_R$ denotes the representative element of the equivalence class of π and $\Pi_R = \{[\pi]_R \mid \pi \in tdPaths_{\mathcal{TK}}(\sigma)\}$ denotes the set of all representative elements.

We define a mapping between representative elements of R and the time divergent paths starting in σ^{max} in $\mathcal{TK}^{max}$, i.e., $tdPaths_{\mathcal{TK}^{max}}(\sigma^{max})$. relations $f : \Pi_R \to \Pi_{R^{max}}$, mapping to each

$$\pi = \sigma \overset{r_1}{\hookrightarrow} \sigma_1 \overset{r_2}{\hookrightarrow} \sigma_2 \overset{r_3}{\hookrightarrow} \cdots$$

the unique path

$$f(\pi) = \sigma^{max} \overset{r_1}{\hookrightarrow} \sigma'_1 \overset{r_2}{\hookrightarrow} \sigma'_2 \overset{r_3}{\hookrightarrow} \cdots$$

such that, for each i, $\sigma_i \mathcal{R}_{\mathcal{C}} \sigma'_i$. Note that f is bijective and that f maps TCTL-equivalent paths, i.e., for all TCTL path formulas ψ and all $\pi \in \Pi_R$ we have that $\mathcal{TK}, \pi \models_c \psi$ iff $\mathcal{TK}^{max}, f(\pi) \models_c \psi$. In particular, the image of Π_R under f is the entire set of time divergent paths starting in σ^{max} in $\mathcal{TK}^{max}$, i.e., $f[\Pi_R] = tdPaths_{\mathcal{TK}^{max}}(\sigma^{max})$.

We prove that $\mathcal{TK}, \sigma \models_c \varphi$ iff $\mathcal{TK}^{max}, \sigma^{max} \models_c \varphi$, by induction on the structure of φ. We already noticed that tick-invariance guarantees that the equivalence holds for atomic propositions. The cases for the boolean connectives are straightforward.

- For $\varphi = A\,\psi$ we have that

$$
\begin{array}{lll}
\mathcal{TK}, \sigma \models_c A\ \psi & \overset{\text{TCTL semantics}}{\iff} & \forall \pi \in tdPaths_{\mathcal{TK}}(\sigma).\ \mathcal{TK}, \pi \models_c \psi \\
& \overset{R \text{ is TCTL-preserving}}{\iff} & \forall \pi \in \Pi_R.\ \mathcal{TK}, \pi \models_c \psi \\
& \overset{f \text{ is bijective and TCTL-preserving}}{\iff} & \forall \pi^{max} \in tdPaths_{\mathcal{TK}^{max}}(\sigma^{max}).\ \mathcal{TK}^{max}, \pi^{max} \models_c \psi \\
& \overset{\text{TCTL semantics}}{\iff} & \mathcal{TK}^{max}, \sigma^{max} \models_c A\ \psi
\end{array}
$$

- For $\varphi = E\ \psi$ we have that

$$
\begin{array}{lll}
\mathcal{TK}, \sigma \models_c E\ \psi & \overset{\text{TCTL semantics}}{\iff} & \exists \pi \in tdPaths_{\mathcal{TK}}(\sigma).\ \mathcal{TK}, \pi \models_c \psi \\
& \overset{R \text{ is TCTL-preserving}}{\iff} & \exists \pi \in \Pi_R.\ \mathcal{TK}, \pi \models_c \psi \\
& \overset{f \text{ is bijective and TCTL-preserving}}{\iff} & \exists \pi^{max} \in tdPaths_{\mathcal{TK}^{max}}(\sigma^{max}).\ \mathcal{TK}^{max}, \pi^{max} \models_c \psi \\
& \overset{\text{TCTL semantics}}{\iff} & \mathcal{TK}^{max}, \sigma^{max} \models_c E\ \psi \qquad \square
\end{array}
$$

Lemma 6.2. *Assume a time-robust real-time rewrite theory $\mathcal{R}$ whose time domain satisfies the theory $TIME_\infty^{gcd}$. Let AP be a set of atomic propositions defined in an extension $(\Sigma \cup AP, E \cup D) \supseteq (\Sigma, E)$, with a* labeling function $L_{AP} : \mathbb{T}_{\Sigma/(E \cup A),\texttt{GlobalSystem}} \to \mathcal{P}(AP)$, *such that a subset of atomic propositions $AP' \subseteq AP$ is tick-invariant in $\mathcal{R}$. Let $r, r' \neq 0$ be two terms of sort* `Time`, *and φ a TCTL formula over AP'. Then*

$$\mathcal{R}^{maxDef(r)}, L_{AP'}, t \models_c \varphi \iff \mathcal{R}^{def(r')}, L_{AP'}, t \models_c \varphi$$

for each state t in $\mathcal{R}$.

Proof. By Definition 6.2, it follows that proving the theorem is equivalent to showing

$$\mathcal{TK}(\mathcal{R}^{maxDef(r)}, AP'), t \models_c \varphi \iff \mathcal{TK}(\mathcal{R}^{def(r')}, AP'), t \models_c \varphi.$$

Let $\mathcal{TK}^{maxDef(r)}$ denote $\mathcal{TK}(\mathcal{R}^{maxDef(r)}, AP')$, and let $\mathcal{TK}^{def(r')}$ denote $\mathcal{TK}(\mathcal{R}^{def(r')}, AP')$. We show that $\mathcal{TK}^{def(r')}$ is equivalent to the r'-transformation of $\mathcal{TK}^{maxDef(r)}$ up to a renaming of its states. In particular, by Definition 6.1 $\mathcal{TK}^{maxDef(r)}$ and $\mathcal{TK}^{def(r')}$ have the same instantaneous transitions, and by Definition 3.1 each instantaneous transition $t \xrightarrow{0} t'$ in $\mathcal{TK}^{maxDef(r)}$ corresponds to the instantaneous transition $(t, 0) \xrightarrow{0} (t', 0)$ in $(\mathcal{TK}^{maxDef(r)})^{r'}$.

Furthermore, for each tick transition $t \xrightarrow{r_1} t'$ in $\mathcal{TK}^{maxDef(r)}$, with $r_1 > r'$, there is a corresponding sequence $t \xrightarrow{r'} t_1 \xrightarrow{r'} t_2 \xrightarrow{r'} \cdots \xrightarrow{r'} t_k \xrightarrow{r_1 \dot{-} kr'} t'$ of tick transitions in $\mathcal{TK}^{def(r')}$, where the duration of the last transition is the remainder of the division of r_1 by r'. For the same maximal tick transition there is a corresponding sequence $(t, 0) \xrightarrow{r'} (t, r') \xrightarrow{r'} (t, 2r') \xrightarrow{r'} \cdots \xrightarrow{r'} (t, kr') \xrightarrow{r_1 \dot{-} k \cdot r'} (t', 0)$ of tick transitions in $(\mathcal{TK}^{maxDef(r)})^{r'}$. Finally, for each tick transition $t \xrightarrow{r_1} t'$ in $\mathcal{TK}^{maxDef(r)}$, with $r_1 \leq r'$, there is a corresponding tick transition $t \xrightarrow{r_1} t'$ in $\mathcal{TK}^{def(r')}$, and a corresponding tick transition $(t, 0) \xrightarrow{r_1} (t', 0)$ in $(\mathcal{TK}^{maxDef(r)})^{r'}$.

We can easily define a bijective mapping f from the states in $\mathcal{TK}^{def(r')}$ to the states in $(\mathcal{TK}^{maxDef(r)})^{r'}$ such that the transitions in $\mathcal{TK}^{def(r')}$ are preserved by f as follows:

- for each transition $t \xrightarrow{0} t'$ in $\mathcal{TK}^{def(r')}$ we define
  - $f(t) = (t, 0)$, and
  - $f(t') = (t', 0)$; and
- for each tick transition $t \xrightarrow{r_1} t'$, with $r_1 > r'$, in $\mathcal{TK}^{maxDef(r)}$ and the corresponding sequence $t \xrightarrow{r'} t_1 \xrightarrow{r'} t_2 \xrightarrow{r'} \cdots \xrightarrow{r'} t_k \xrightarrow{r_1 \dot{-} kr'} t'$ of tick transitions in $\mathcal{TK}^{def(r')}$, where the duration of the last transition is the remainder of the division of r_1 by r', we define
  - $f(t) = (t, 0)$,
  - $f(t_i) = (t, i \cdot r')$, and
  - $f(t') = (t', 0)$.

  Furthermore, for each tick transition $t \xrightarrow{r_1} t'$, with $r_1 \leq r'$, we define
- $f(t) = (t, 0)$, and
- $f(t') = (t', 0)$.

It is easy to see that f is a bijective function preserving the transitions in $\mathcal{TK}^{def(r')}$, i.e., for each transition $t \xrightarrow{r_1} t'$ in $\mathcal{TK}^{def(r')}$, $f(t) \xrightarrow{r_1} (t')$ is the corresponding transition in $(\mathcal{TK}^{maxDef(r)})^{r'}$. By Lemma 3.1 we have that $(\mathcal{TK}^{maxDef(r)})^{r'}, (t, 0) \models_c \varphi \iff \mathcal{TK}^{maxDef(r)}, t \models_c \varphi$. $\square$

Theorem 6.3. *Assume a real-time rewrite theory $\mathcal{R}$ with time domain* `NAT-TIME-DOMAIN-WITH-INF`. *Let AP be a set of atomic propositions defined in an extension $(\Sigma \cup AP, (E \cup A) \cup D) \supseteq (\Sigma, E \cup A)$, with a labeling function $L_{AP} : \mathbb{T}_{\Sigma/(E \cup A),\texttt{GlobalSystem}} \to \mathcal{P}(AP)$. Let $r \neq 0$ be a term of sort* `Time`, *and φ a TCTL formula over AP. Then*

$$\mathcal{R}, L_{AP}, t \models_c \varphi \quad \Longleftrightarrow \quad \mathcal{R}^{def(1)}, L_{AP}, t \models_p \varphi$$

for each state t in $\mathcal{R}$.

Proof. By Definition 6.2, it follows that proving the theorem is equivalent to showing

$$\mathcal{TK}(\mathcal{R}, AP), t \models_c \varphi \quad \Longleftrightarrow \quad \mathcal{TK}(\mathcal{R}^{def(1)}, AP), t \models_p \varphi.$$

By Theorem 3.5, it holds that

$$\mathcal{TK}(\mathcal{R}, AP), t \models_c \varphi \quad \Longleftrightarrow \quad \mathcal{TK}(\mathcal{R}, AP)^1, (t, 0) \models_p \varphi,$$

where $\mathcal{TK}(\mathcal{R}, AP)^1$ is the 1-transformation of $\mathcal{TK}(\mathcal{R}, AP)$.

Let $\mathcal{TK}^{def(1)}$ denote $\mathcal{TK}(\mathcal{R}^{def(1)}, AP)$ and $\mathcal{TK}$ denote $\mathcal{TK}(\mathcal{R}, AP)$. The proof is similar to the one for Lemma 6.2. In particular, we show that $\mathcal{TK}^{def(1)}$ is equivalent to the 1-transformation of $\mathcal{TK}$ up to a renaming of its states. By Definition 6.1 $\mathcal{TK}^{def(1)}$ and $\mathcal{TK}$ have the same instantaneous transitions, and by Definition 3.1 each instantaneous transition $t \xrightarrow{0} t'$ in $\mathcal{TK}$ corresponds to the instantaneous transition $(t, 0) \xrightarrow{0} (t', 0)$ in $\mathcal{TK}^1$. Furthermore, for each tick step $t \xrightarrow{n} t'$ in $\mathcal{R}$, with $n > 0$, there is a corresponding tick transition $t \xrightarrow{n} t'$ in $\mathcal{TK}$, and a corresponding sequence $t \xrightarrow{1} t_2 \xrightarrow{1} t_3 \xrightarrow{1} \cdots \xrightarrow{1} t_n \xrightarrow{1} t'$ of tick transitions in $\mathcal{TK}^{def(1)}$. For the same tick step there is a corresponding sequence $(t, 0) \xrightarrow{1} (t, 1) \xrightarrow{1} (t, 2) \xrightarrow{1} \cdots \xrightarrow{1} (t, n-1) \xrightarrow{1} (t', 0)$ of tick transitions in $\mathcal{TK}^1$.

We can easily define a bijective mapping f from the states in $\mathcal{TK}^{def(1)}$ to the states in $\mathcal{TK}^1$ such that the transitions in $\mathcal{TK}^{def(1)}$ are preserved by f as follows:

- for each transition $t \xrightarrow{0} t'$ in $\mathcal{TK}^{def(1)}$ we define
  - $f(t) = (t, 0)$, and
  - $f(t') = (t', 0)$; and
- for each tick transition $t \xrightarrow{n} t'$, with $n > 0$ in $\mathcal{TK}$ and the corresponding sequence $t \xrightarrow{1} t_2 \xrightarrow{1} t_3 \xrightarrow{1} \cdots \xrightarrow{1} t_n \xrightarrow{1} t'$ of tick transitions in $\mathcal{TK}^{def(1)}$ we define
  - $f(t) = (t, 0)$,
  - $f(t_{i+1}) = (t, i)$, for $0 < i < n$, and
  - $f(t') = (t', 0)$.

It is easy to see that f is a bijective function preserving the transitions in $\mathcal{TK}^{def(1)}$, i.e., for each transition $t \xrightarrow{n} t'$ in $\mathcal{TK}^{def(1)}$, $f(t) \xrightarrow{n} (t')$ is the corresponding transition in $\mathcal{TK}^1$. □

References

[1] R. Alur, D.L. Dill, A theory of timed automata, Theor. Comput. Sci. 126 (1994) 183–235.
[2] P.C. Ölveczky, J. Meseguer, Specification of real-time and hybrid systems in rewriting logic, Theor. Comput. Sci. 285 (2002) 359–405.
[3] B. Dutertre, M. Sorea, Modeling and verification of a fault-tolerant real-time startup protocol using calendar automata, in: FORMATS/FTRTFT, in: Lecture Notes in Computer Science, vol. 3253, Springer, 2004, pp. 199–214.
[4] K. Bae, P.C. Ölveczky, T.H. Feng, E.A. Lee, S. Tripakis, Verifying hierarchical Ptolemy II discrete-event models using Real-Time Maude, Sci. Comput. Program. 77 (2012) 1235–1271.
[5] P.C. Ölveczky, J. Meseguer, Abstraction and completeness for Real-Time Maude, Electron. Notes Theor. Comput. Sci. 176 (2007) 5–27.
[6] R. Alur, C. Courcoubetis, D. Dill, Model-checking in dense real-time, Inf. Comput. 104 (1993) 2–34.
[7] F. Laroussinie, N. Markey, P. Schnoebelen, Efficient timed model checking for discrete-time systems, Theor. Comput. Sci. 353 (2006) 249–271.
[8] P.C. Ölveczky, J. Meseguer, The Real-Time Maude tool, in: TACAS, in: Lecture Notes in Computer Science, vol. 4963, Springer-Verlag, 2008, pp. 332–336.
[9] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, C. Talcott, All about Maude, Lecture Notes in Computer Science, vol. 4350, Springer-Verlag, 2007.
[10] M. Katelman, J. Meseguer, J. Hou, Redesign of the LMST wireless sensor protocol through formal modeling and statistical model checking, in: FMOODS, in: Lecture Notes in Computer Science, vol. 5051, Springer-Verlag, 2008, pp. 150–169.
[11] P.C. Ölveczky, S. Thorvaldsen, Formal modeling, performance estimation, and model checking of wireless sensor network algorithms in Real-Time Maude, Theor. Comput. Sci. 410 (2009) 254–280.
[12] P.C. Ölveczky, J. Meseguer, C.L. Talcott, Specification and analysis of the AER/NCA active network protocol suite in Real-Time Maude, Form. Methods Syst. Des. 29 (2006) 253–293.
[13] E. Lien, P.C. Ölveczky, Formal modeling and analysis of an IETF multicast protocol, in: SEFM, IEEE Computer Society, 2009, pp. 273–282.
[14] P.C. Ölveczky, M. Caccamo, Formal simulation and analysis of the CASH scheduling algorithm in Real-Time Maude, in: FASE, in: Lecture Notes in Computer Science, vol. 3922, Springer-Verlag, 2006, pp. 357–372.
[15] A. Riesco, A. Verdejo, Implementing and analyzing in Maude the enhanced interior gateway routing protocol, Electron. Notes Theor. Comput. Sci. 238 (2009) 249–266.
[16] J. Grov, P.C. Ölveczky, Scalable and fully consistent transactions in the cloud through hierarchical validation, in: Globe, in: Lecture Notes in Computer Science, vol. 8059, Springer, 2013, pp. 26–38.
[17] J. Grov, P.C. Ölveczky, Formal modeling and analysis of Google's Megastore in Real-Time Maude, in: Specification, Algebra, and Software, in: Lecture Notes in Computer Science, vol. 8373, Springer, 2014, pp. 494–519.

[18] P.C. Ölveczky, Semantics, simulation, and formal analysis of modeling languages for embedded systems in Real-Time Maude, in: Formal Modeling: Actors, Open Systems, Biological Systems, in: Lecture Notes in Computer Science, vol. 7000, Springer-Verlag, 2011, pp. 368–402.
[19] P.C. Ölveczky, A. Boronat, J. Meseguer, Formal semantics and analysis of behavioral AADL models in Real-Time Maude, in: FMOODS/FORTE, in: Lecture Notes in Computer Science, vol. 6117, Springer-Verlag, 2010, pp. 47–62.
[20] K. Bae, P.C. Ölveczky, A. Al-Nayeem, J. Meseguer, Synchronous AADL and its formal analysis in Real-Time Maude, in: ICFEM, in: Lecture Notes in Computer Science, vol. 6991, Springer-Verlag, 2011, pp. 651–667.
[21] K. Bae, P.C. Ölveczky, J. Meseguer, A. Al-Nayeem, The SynchAADL2Maude tool, in: FASE, in: Lecture Notes in Computer Science, vol. 7212, Springer, 2012, pp. 59–62.
[22] M. AlTurki, J. Meseguer, Real-time rewriting semantics of Orc, in: PPDP, ACM, 2007, pp. 131–142.
[23] J.E. Rivera, F. Durán, A. Vallecillo, On the behavioral semantics of real-time domain specific visual languages, in: WRLA, in: Lecture Notes in Computer Science, vol. 6381, Springer-Verlag, 2010, pp. 174–190.
[24] A. Boronat, P.C. Ölveczky, Formal real-time model transformations in MOMENT2, in: FASE, in: Lecture Notes in Computer Science, vol. 6013, Springer-Verlag, 2010, pp. 29–43.
[25] Z. Sabahi-Kaviani, R. Khosravi, M. Sirjani, P.C. Ölveczky, E. Khamespanah, Formal semantics and analysis of Timed Rebeca in Real-Time Maude, in: FTSCS, in: Communications in Computer and Information Science, vol. 419, Springer, 2014, pp. 178–194.
[26] D. Lepri, E. Ábrahám, P.C. Ölveczky, Timed CTL model checking in Real-Time Maude, in: WRLA, in: Lecture Notes in Computer Science, vol. 7571, Springer, 2012, pp. 182–200.
[27] E.A. Emerson, A.K. Mok, A.P. Sistla, J. Srinivasan, Quantitative temporal reasoning, Real-Time Syst. 4 (1992) 331–352.
[28] S.V.A. Campos, E.M. Clarke, W.R. Marrero, M. Minea, H. Hiraishi, Computing quantitative characteristics of finite-state real-time systems, in: RTSS, IEEE Computer Society, 1994, pp. 266–270.
[29] S.V.A. Campos, M. Teixeira, M. Minea, A. Kuehlmann, E.M. Clarke, Model checking semi-continuous time models using BDDs, Electron. Notes Theor. Comput. Sci. 23 (1999) 75–87.
[30] F. Laroussinie, P. Schnoebelen, M. Turuani, On the expressivity and complexity of quantitative branching-time temporal logics, Theor. Comput. Sci. 297 (2003) 297–315.
[31] S.V. Campos, E.M. Clarke, Real-time symbolic model checking for discrete time models, AMAST Ser. Comput. 2 (1995) 129–145.
[32] R. Alur, T. Henzinger, Logics and models of real time: a survey, in: Real Time: Theory in Practice, in: Lecture Notes in Computer Science, vol. 600, Springer-Verlag, 1992, pp. 74–106.
[33] F. Wang, Formal verification of timed systems: a survey and perspective, Proc. IEEE 92 (2004) 1283–1307.
[34] E.M. Clarke, E.A. Emerson, Design and synthesis of synchronization skeletons using branching-time temporal logic, in: Logic of Programs, in: Lecture Notes in Computer Science, vol. 131, Springer, 1981, pp. 52–71.
[35] E. Clarke, O. Grumberg, D.A. Peled, Model Checking, MIT Press, 1999.
[36] P. Bouyer, Model-checking timed temporal logics, Electron. Notes Theor. Comput. Sci. 231 (2009) 323–341.
[37] R. Alur, C. Courcoubetis, D. Dill, Model-checking in dense real-time, Inf. Comput. 104 (1993) 2–34.
[38] C. Baier, J.-P. Katoen, Principles of Model Checking, MIT Press, 2008.
[39] M.L. Fredman, R.E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, J. ACM 34 (1987) 596–615.
[40] P.C. Ölveczky, J. Meseguer, Semantics and pragmatics of Real-Time Maude, High.-Order Symb. Comput. 20 (2007) 161–196.
[41] R. Bruni, J. Meseguer, Semantic foundations for generalized rewrite theories, Theor. Comput. Sci. 360 (2006) 386–414.
[42] J. Meseguer, Conditional rewriting logic as a unified model of concurrency, Theor. Comput. Sci. 96 (1992) 73–155.
[43] J. Meseguer, Membership algebra as a logical framework for equational specification, in: WADT, in: Lecture Notes in Computer Science, vol. 1376, Springer-Verlag, 1998, pp. 18–61.
[44] P. Viry, Equational rules for rewriting logic, Theor. Comput. Sci. 285 (2002) 487–517.
[45] G. Rote, Crossing the bridge at night, EATCS Bull. 78 (2002) 241–246.
[46] S. Yovine, Kronos: a verification tool for real-time systems, Int. J. Softw. Tools Technol. Transf. 1 (1997) 123–133.
[47] F. Wang, REDLIB for the formal verification of embedded systems, in: ISoLA IEEE, 2006, pp. 341–346.
[48] N. Markey, P. Schnoebelen, TSMV: a symbolic model checker for quantitative analysis of systems, in: QEST, IEEE Computer Society, 2004, pp. 330–331.
[49] E.M. Clarke, O. Grumberg, K.L. McMillan, X. Zhao, Efficient generation of counterexamples and witnesses in symbolic model checking, in: DAC, 1995, pp. 427–432.
[50] E.M. Clarke, S. Jha, Y. Lu, H. Veith, Tree-like counterexamples in model checking, in: LICS, IEEE Computer Society, 2002, pp. 19–29.
[51] P.C. Ölveczky, Towards formal modeling and analysis of networks of embedded medical devices in Real-Time Maude, in: SNPD, IEEE, 2008, pp. 241–248.
[52] D. Lepri, P.C. Ölveczky, E. Ábrahám, Model checking classes of metric LTL properties of object-oriented Real-Time Maude specifications, in: RTRTS, in: Electronic Proceedings in Theoretical Computer Science, vol. 36, 2010, pp. 117–136.
[53] J. Eker, J.W. Janneck, E.A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, Y. Xiong, Taming heterogeneity—the Ptolemy approach, Proc. IEEE 91 (2003) 127–144.
[54] F. Wang, Efficient verification of timed automata with BDD-like data structures, Int. J. Softw. Tools Technol. Transf. 6 (2004) 77–97.
[55] G. Behrmann, A. David, K.G. Larsen, A tutorial on Uppaal, in: SFM-RT, in: Lecture Notes in Computer Science, vol. 3185, Springer-Verlag, 2004, pp. 200–236.
[56] R. Alur, T.A. Henzinger, A really temporal logic, J. ACM 41 (1994) 181–203.
[57] T.A. Henzinger, Z. Manna, A. Pnueli, What good are digital clocks?, in: ICALP, in: Lecture Notes in Computer Science, vol. 623, Springer, 1992, pp. 545–558.
[58] H. Boucheneb, G. Gardey, O.H. Roux, TCTL model checking of time Petri nets, J. Log. Comput. 19 (2009) 1509–1540.
[59] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, L.J. Hwang, Symbolic model checking: 10^{20} states and beyond, Inf. Comput. 98 (1992) 142–170.
[60] D. Dill, Timing assumptions and verification of finite-state concurrent systems, in: Automatic Verification Methods for Finite State Systems, in: Lecture Notes in Computer Science, vol. 407, Springer, 1990, pp. 197–212.
[61] F. Wang, A.K. Mok, E.A. Emerson, Symbolic model checking for distributed real-time systems, in: FME, in: Lecture Notes in Computer Science, vol. 670, Springer, 1993, pp. 632–651.
[62] E. Asarin, M. Bozga, A. Kerbrat, O. Maler, A. Pnueli, A. Rasse, Data-structures for the verification of timed automata, in: HART, in: Lecture Notes in Computer Science, vol. 1201, Springer, 1997, pp. 346–360.
[63] G. Behrmann, K.G. Larsen, J. Pearson, C. Weise, W. Yi, Efficient timed reachability analysis using clock difference diagrams, in: CAV, in: Lecture Notes in Computer Science, vol. 1633, Springer, 1999, pp. 341–353.
[64] S. Tripakis, S. Yovine, Analysis of timed systems using time-abstracting bisimulations, Form. Methods Syst. Des. 18 (2001) 25–68.
[65] A. Giua, F. DiCesare, M. Silva, Generalized mutual exclusion constraints on nets with uncontrollable transitions, IEEE Trans. Syst. Man Cybern. (1992) 974–979.
[66] G. Gardey, D. Lime, M. Magnin, O.H. Roux, Roméo: a tool for analyzing time Petri nets, in: CAV, in: Lecture Notes in Computer Science, vol. 3576, Springer, 2005, pp. 418–423.

[67] B. Berthomieu, F. Vernadat, Time Petri nets analysis with TINA, in: QEST, IEEE Computer Society, 2006, pp. 123–124.
[68] M. Wirsing, S.S. Bauer, A. Schroeder, Modeling and analyzing adaptive user-centric systems in Real-Time Maude, in: RTRTS, in: Electronic Proceedings in Theoretical Computer Science, vol. 36, 2010, pp. 1–25.