Model Checking Timed Recursive CTL

Florian Bruse ✉
School of Electrical Engineering and Computer Science, University of Kassel, Germany

Martin Lange ✉
School of Electrical Engineering and Computer Science, University of Kassel, Germany

Abstract

We introduce Timed Recursive CTL, a merger of two extensions of the well-known branching-time logic CTL: Timed CTL is interpreted over real-time systems like timed automata; Recursive CTL introduces a powerful recursion operator which takes the expressiveness of this logic CTL well beyond that of regular properties. The result is an expressive logic for real-time properties. We show that its model checking problem is decidable over timed automata, namely 2-EXPTIME-complete.

2012 ACM Subject Classification Theory of computation → Modal and temporal logics; Theory of computation → Program specifications

Keywords and phrases formal specification, temporal logic, real-time systems

Digital Object Identifier 10.4230/LIPIcs.TIME.2021.12

Acknowledgements The open access publication of this article was supported by the Alpen-Adria-Universität Klagenfurt, Austria.

1 Introduction

Temporal logics are widely used as formal languages for the specification of properties of reactive systems. The most widely known such logics are LTL [19] and CTL [12], having achieved this status partially due to their simplicity as extensions of propositional logic by a small set of intuitive temporal operators. This simplicity in syntax is also reflected by relatively low expressive power; both do not even reach up to full regularity in the sense that they are not equi-expressive to finite-state word, resp. tree automata.

Regular expressive power is a cornerstone in the study of the theory of temporal specification languages, as logics not exceeding this expressivity limit typically possess appealing properties like decidability of their model and satisfiability checking problems.

On the other hand, there are also interesting program properties which are not regular and can therefore not be expressed in such logics, like the absence of buffer over-/underflows, assume-guarantee properties, etc. The literature contains several non-regular extensions of temporal logics or related modal fixpoint logics, e.g. PDL[CFL] [13], FLC [18] and HFL [21]. These have certain features in common: a syntax that makes it difficult to understand the meaning of formulas, and – despite undecidability of their satisfiability problems – a decidable model-checking problem over finite structures [15, 16, 5]. The upshot to take from this is that model checking need not become undecidable when going beyond regular expressiveness.

In order to overcome issues with unintuitive syntaxes in expressive temporal logics, we recently proposed Recursive CTL (RecCTL) [10], an extension of the basic branching-time temporal logic CTL with a single recursion operator which takes formulas as arguments that can be manipulated using other operators and passed into a recursive call. This achieves expressive power, capturing all regular branching-time properties and many non-regular ones. At the same time, model checking is decidable albeit exponentially worse: it is EXPTIME-complete compared to P-completeness for CTL.

Another way of extending the expressive power of temporal logics, which has been followed in the literature for quite some time, is more semantic in nature: the labelled transition systems that logics like CTL are interpreted over model the evolution of time very abstractly by discrete steps taken when passing from one state to another. Hence, the only real timing properties expressible in such logics are unitless and non-quantitative like "*at some point in the future*" etc. This is not sufficient for the modelling of embedded or real-time systems where concrete timing constraints play a role in correctness properties, for instance as in "*within 5 milliseconds of receiving a signal, a control command is issued.*"

In order to capture such effects, transition systems have been extended to model the flow of time more realistically with non-negative, real-numbered delays between time point. Timed Automata [3] are a popular model for the finite representation of such systems. Their great expressiveness compared to ordinary discrete systems shows by the fact that the basis for temporal logics, the simple reachability problem, is already PSPACE-complete.

One of the most popular temporal logics for expressing more complex reachability properties of Timed Automata is Timed CTL [2], an extension of CTL that is capable of making simple assertions about the amount of time that passes before certain events occur on some, resp. all paths. Its model checking problem over Timed Automata is not more difficult than simple reachability: it is also PSPACE-complete, cf. [17].

Here we introduce and study Timed Recursive CTL (TRCTL), a logic that arises from combining the extensions to real-time on one hand, and to non-regular properties on the other. We show that TRCTL retains decidability of model checking over timed automata, but the combination increases the complexity to 2-EXPTIME-completeness.

The paper is organised as follows. In Sect. 2 we recall necessary preliminaries about Timed Automata and TCTL, about RecCTL, and characterise doubly exponential time complexity. In Sect. 3 we introduce TRCTL formally. In Sect. 4 we establish 2-EXPTIME-completeness of its model checking problem. The upper bound is obtained by an exponential reduction to the RecCTL model checking problem, making use of the known region graph abstraction. The lower bound requires a fair amount of encoding large numbers as propositions interpreted over timed automata. TRCTL is then capable of mimicking the aforementioned characterisation of doubly exponential time. We conclude in Sect. 5 with remarks on further work.

2 Preliminaries

2.1 Doubly Exponential Time Complexity

The main result of this paper is 2-EXPTIME-completeness of an expressive extension of Timed CTL interpreted over Timed Automata. For the lower bound we reduce from a problem that is essentially a reformulation of the problem to decide whether a deterministic Turing Machine (DTM) accepts the empty word in 2^{2^n} steps, given some $n \geq 0$.

Suppose Q is the set of states and Γ the tape alphabet of a DTM $\mathcal{M}$, containing a special $\square$ symbol and not containing $\#$. Let $\hat{\Gamma} := \Gamma \cup (Q \times \Gamma) \cup \{\#\}$. Let $f : \mathbb{N} \to \mathbb{N}$. The unique $f(n)$-time-bounded computation of $\mathcal{M}$ on the empty input can be represented by a square, containing $f(n)$ rows, representing time, each of which contains $f(n)$ symbols from $\hat{\Gamma}$, representing a configuration, or space. Each row is of the form $\#w\#$ for some $w \in (\hat{\Gamma} \setminus \{\#\})^{f(n)-2}$, and, if q_0, q_{acc} are $\mathcal{M}$'s starting and accepting states, the bottom row is $\#(q_0, \square)\square^{f(n)-3}\#$, and the top row is of the form $\#(q_{\mathsf{acc}}, \square)w\#$ for some $w \in \hat{\Gamma}^{f(n)-3}$.

Now suppose that δ is $\mathcal{M}$'s transition function. This gives rise to a relation $\hat{\delta} \subseteq \hat{\Gamma}$ such that $(y_1, y_2, y_3, x) \in \hat{\delta}$ iff whenever y_1, y_2, y_3 are consecutive symbols in row t at positions $s-1, s, s+1$, then x is the symbol at position s in row $t+1$.

An $f(n)$-*certificate* (for $\mathcal{M}$ and given n) is a set of mutually recursive predicates $Cert_a : [f(n)] \times [f(n)] \to \{\top, \bot\}$, one for each $a \in \hat{\Gamma}$ with the following properties. Intuitively, $Cert_a(t,s) = \top$ iff the s-th symbol in the t-the configuration of the unique computation of $\mathcal{M}$ on the empty input is a. Clearly, $t, s \leq f(n)$. Formally,

- $Cert_{(q_{\mathrm{acc}},\Box)}(f(n)-1, 0) = \top$,
- for all $t \in \{1, \dots, f(n)-1\}$, $s \in \{1, \dots, f(n)-2\}$ and $a \in \hat{\Gamma} \setminus \{\#\}$ with $Cert_a(t,s)$ there are $b_1, b_2, b_3 \in \hat{\Gamma}$ with $(b_1, b_2, b_3, a) \in \hat{\delta}$ and
$$Cert_{y_1}(t-1, s-1) \wedge Cert_{y_2}(t-1, s) \wedge Cert_{y_3}(t-1, s+1) \ ,$$
- for all $t \in \{0, \dots, f(n)-1\}$, $s \in \{0, f(n)-1\}$ we have $Cert_a(t,s)$ iff $a = \#$,
- $Cert_a(0,1)$ iff $a = (q_0, \Box)$, and for all $s = 2, \dots, f(n)-2$: $Cert_a(0,s)$ iff $a = \Box$.

Note that the last two clauses determine the values of a in $Cert_a(t,s)$ uniquely for the left, lower und right edge of the square defined by the coordinates t, s, and determinism of the TM $\mathcal{A}$ then determines the values at the inner coordinates uniquely as well.

This characterisation of acceptance in deterministic time-bounded Turing Machines is taken from [11] and can also be used to establish a generic 2-EXPTIME-hardness result.

▶ **Proposition 1.** *It is* 2-EXPTIME*-hard to decide, given a DTM $\mathcal{M}$ and an $n \in \mathbb{N}$ encoded unarily, whether or not there is a 2^{2^n}-certificate for $\mathcal{M}$ and n in the sense above.*

2.2 Models of Real-Time Systems

Timed Transition Systems. A *timed labelled transition system* (TLTS) over a finite set *Prop* of atomic propositions is a $\mathcal{T} = (\mathcal{S}, \to, s_0, \lambda)$ such that

- $\mathcal{S}$ is a set of *states* containing a designated starting state s_0,
- $\to \subseteq \mathcal{S} \times \mathcal{S} \cup \mathcal{S} \times \mathbb{R}^{\geq 0} \times \mathcal{S}$ is the transition relation, consisting of two kinds:
  - *discrete transitions* of the form $s \to t$ for $s, t \in \mathcal{S}$, and
  - *delay transitions* of the form $s \xrightarrow{d} t$ for $s, t \in \mathcal{S}$ and $d \in \mathbb{R}^{\geq 0}$, satisfying $s \xrightarrow{0} t$ iff $s = t$ for any $s, t \in \mathcal{S}$, and
  $$\forall d, d_1, d_2 \in \mathbb{R}^{\geq 0}, \forall s, t \in \mathcal{S} : d = d_1 + d_2 \text{ and } s \xrightarrow{d} t \Leftrightarrow \exists u \in \mathcal{S} \text{ s.t. } s \xrightarrow{d_1} u \text{ and } u \xrightarrow{d_2} t \ ,$$
- $\lambda : \mathcal{S} \to 2^{Prop}$ labels the states with the set of atomic propositions that hold true in it.

Here we consider TLTS over a singleton set of discrete actions. This is purely done since the temporal logics based on CTL here do not consider different actions. It would be possible to extend the entire theory to TLTS over several discrete transition relations $\xrightarrow{a}, \xrightarrow{b}, \dots$, and make the logics aware of these. The *extended transition relations* $\overset{d}{\Longrightarrow}$, $d \in \mathbb{R}^{\geq 0}$, are obtained by padding discrete transitions with delays:

$$s \overset{d}{\Longrightarrow} t \quad \text{iff} \quad \exists d_1, d_2 \in \mathbb{R}^{\geq 0}, s', t' \in \mathcal{S} \text{ s.t. } s \xrightarrow{d_1} s', s' \to t', t' \xrightarrow{d_2} t \text{ and } d = d_1 + d_2$$

A *trace* is a sequence $\pi = s_0 \overset{d_0}{\Longrightarrow} s_1 \overset{d_1}{\Longrightarrow} \dots$

An (untimed) labeled transition system (LTS) is a TLTS over an empty delay transition relation. It is *finite* if the set of its states is finite.

Clock Constraints. Let $\mathcal{X} = \{\mathtt{x}, \mathtt{y}, \ldots\}$ be a set of $\mathbb{R}^{\geq 0}$-valued variables called *clocks*. By $CC(\mathcal{X})$ we denote the set of *clock constraints* over $\mathcal{X}$ which are conjunctive formulas of the form $\top$ or $\mathtt{x} \oplus c$ for $\mathtt{x} \in \mathcal{X}$, $c \in \mathbb{N}$ and $\oplus \in \{\leq, <, \geq, >, =\}$.

A *clock evaluation* is an $\eta : \mathcal{X} \to \mathbb{R}^{\geq 0}$. A clock constraint φ is interpreted in a clock evaluation η in the obvious way:

- $\eta \models \top$ holds for any η,
- $\eta \models \varphi_1 \wedge \varphi_2$ if $\eta \models \varphi_1$ and $\eta \models \varphi_2$,
- $\eta \models \mathtt{x} \oplus c$ if $\eta(\mathtt{x}) \oplus q$ for $\oplus \in \{\leq, <, \geq, >, =\}$.

Given a clock evaluation η, $d \in \mathbb{R}^{\geq 0}$ and a set $R \subseteq \mathcal{X}$, we write $\eta{+}d$ for the clock evaluation that is defined by $(\eta{+}d)(\mathtt{x}) = \eta(\mathtt{x}) + d$ for any $\mathtt{x} \in \mathcal{X}$, and $\eta|_R$ for the clock evaluation that is defined by $\eta|_R(\mathtt{x}) = 0$ for $\mathtt{x} \in R$ and $\eta|_R(\mathtt{x}) = \eta(\mathtt{x})$ otherwise.

Timed Automata. As with TLTS, here we consider timed automata whose transitions are always taken with a single action which is consequently not named. As above, the reason for considering this simplified model is purely the fact that CTL-based logics as defined – the main object of study in this paper – are oblivious of differences in actions anyway.

A *timed automaton* (TA) over *Prop* is a $\mathcal{A} = (L, \mathcal{X}, \ell_0, \iota, \delta, \lambda)$ where

- L is a finite set of so-called *locations* containing a designated *initial* location $\ell_0 \in L$,
- $\mathcal{X}$ is a finite set of clocks,
- $\iota : L \to CC(\mathcal{X})$ assigns a clock constraint, called *invariant*, to each location,
- $\delta \subseteq L \times CC(\mathcal{X}) \times 2^{\mathcal{X}} \times L$ is a finite set of transitions. We write $\ell \xrightarrow{g,R} \ell'$ instead of $(\ell, g, R, \ell') \in \delta$. In such a transition, g is called the *guard*, and $R \subseteq \mathcal{X}$ are the *reset* clocks of this transition.

The *index* of the TA $\mathcal{A}$ is the largest constant occurring in its invariants or guards, denoted $m(\mathcal{A})$. The *size* of $\mathcal{A}$ is

$$|\mathcal{A}| = |\delta| \cdot (2 \cdot (\log L) + |\mathcal{X}| + \log m(\mathcal{A})) + |L| \cdot 2 \cdot (\log |\mathcal{X}| + \log m(\mathcal{A})) + |L| \cdot |Prop|.$$

Note that the size is only logarithmic in the value of constants used in clock constraints as they can be represented in binary notation for instance.

TA are models of state-based real-time systems. The semantics, resp. behaviour of a TA $\mathcal{A} = (L, \mathcal{X}, \ell_0, \iota, \delta, \lambda)$ is given by an TLTS $\mathcal{T}_{\mathcal{A}}$ over the time domain $\mathbb{R}^{\geq 0}$ as follows.

- The state set is $\mathcal{S} = \{(\ell, \eta) \mid \ell \in L, \eta \in (\mathcal{X} \to \mathbb{R}^{\geq 0})$ such that $\eta \models \iota(\ell)\}$ consisting of pairs of locations and clock evaluations that satisfy the locations's invariant.
- The initial state is $s_0 = (\ell_0, \eta_0)$ where $\eta_0(\mathtt{x}) = 0$ for all $\mathtt{x} \in \mathcal{X}$.
- Delay transitions retain the underlying location and (possibly) advance the value of clocks in a state: for any $(\ell, \eta) \in \mathcal{S}$ and $d \in \mathbb{R}^{\geq 0}$ we have $(\ell, \eta) \xrightarrow{d} (\ell, \eta{+}d)$ if $\eta{+}d \models \iota(\ell)$.
- Discrete transitions possibly change the location and reset clocks: for any $(\ell, \eta) \in \mathcal{S}$, $\ell' \in L$ and $R \subseteq \mathcal{X}$ we have $(\ell, \eta) \to (\ell', \eta|_R)$ if there is $g \in CC(\mathcal{X})$ such that $(\ell, g, R, \ell') \in \delta$ and $\eta|_R \models \iota(\ell')$.
- The propositional label of a state is inherited from the propositional label of the underlying location: $\lambda(\ell, \eta) = \lambda(\ell)$.

In other words, a TA finitely represents a TLTS. Clearly, not every TLTS is finitely representable, so only a subset is captured by TA.

For a detailed introduction into timed automata we refer to the literature [3, 6]. Here we give an example which will be used later on in the lower bound proof in Sect. 4. The TA here act as *gadgets* which means that they have defined locations by which they can be connected to form larger TA. This may entail putting guards or resets onto transitions which do not connect locations in this gadget, as they will be connected later on. The example TA in Fig. 1 are used to encode a counter of some width.

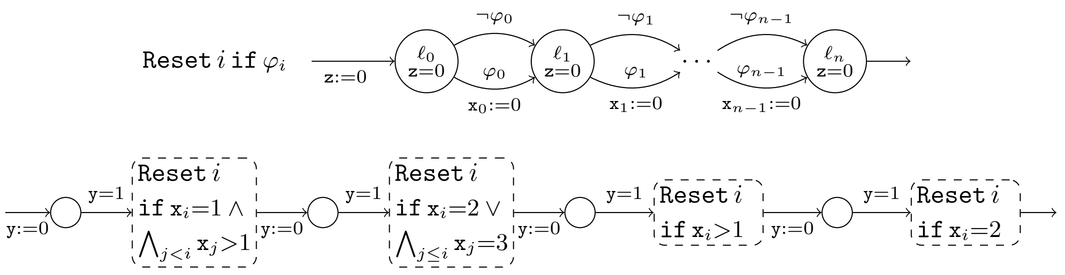


Figure 1 Examples of timed automata: `Reset` i `if` φ_i for arbitrary clock constraints $\varphi_0, \dots, \varphi_{n-1}$ (upper TA), and $\texttt{Incr}_{\bar{\texttt{x}}}$ (lower TA).

▶ **Definition 2.** *Let $\bar{\mathtt{x}} = (\mathtt{x_0}, \dots, \mathtt{x_{n-1}})$. An environment η is called a* small *$\bar{\mathtt{x}}$-counter if $\eta(\mathtt{x_i}) \in \{0,1\}$ for all $i = 0, \dots, n-1$. Its* value *is $\langle \eta_{\bar{\mathtt{x}}} \rangle = \sum_{i=0}^{n-1} \eta(\mathtt{x_i}) \cdot 2^i$.*

We drop the annotation by $\bar{\mathtt{x}}$ if it is clear from context. Now consider the TA `Reset` i `if` φ (read "for $i = 0, \dots, n-1$ reset the i-th clock if φ_i holds") and $\texttt{Incr}_{\bar{\texttt{x}}}$ in Fig. 1.

▶ **Observation 3.**

- *Suppose $\bar{\varphi} = (\varphi_0, \dots, \varphi_{n-1})$ is a tuple of clock constraints over the clocks in $\bar{\mathtt{x}}$. Then, in the TLTS associated to the TA* `Reset` i `if` φ*, there is a path from (ℓ_0, η) to (ℓ_n, η') iff $\eta' = \eta|_{\{\mathtt{x}_i \mid \varphi_i\}}$.*
- *Suppose η is small counter encoding the value $m \in [2^n]$ over n clocks in $\bar{\mathtt{x}}$. Then, if entering the sub-TLTS generated by the gadget $\texttt{Incr}_{\bar{\texttt{x}}}$ from some state (ℓ, η), this sub-TLTS is left towards some (ℓ', η') such that η' encodes $m + 1$ modulo 2^n.*

Clearly, there is a gadget similar to $\texttt{Incr}_{\bar{\texttt{x}}}$ that decreases the value encoded in these clocks. We denote it by $\texttt{Decr}_{\bar{\texttt{x}}}$. Note that technically, $\texttt{Incr}_{\bar{\texttt{x}}}$ is not a TA since the conditions on the resets, resp. their negations, which are used as guards in the reset gadget, contain disjunctions. However, in this case, the guards can be brought into disjunctive normal form at a minimal blowup, whence a disjunction $\psi_1 \vee \ldots \vee \psi_k$ in a supposed guard can be replaced by k separate transitions.

The Region Abstraction. There is a well-known abstraction of an TLTS arising from a TA $\mathcal{A}$ into a finite LTS known as the *region graph* $\mathcal{R}_\mathcal{A}$, used in decidability proofs for decision problems on TA.

In the following we only consider TLTS $\mathcal{T}_\mathcal{A}$ that arise from some TA $\mathcal{A} = (L, \mathcal{X}, \ell_0, \iota, \delta, \lambda)$. The region abstraction is a mapping of such $\mathbb{R}^{\geq 0}$-TLTS into finite LTS. It is based on an equivalence relation $\simeq_m$, for $m \in \mathbb{N}$, on clock evaluations defined as follows.

$$\begin{aligned} \eta \simeq_m \eta' \quad \text{iff} \quad & \text{for all } x \in \mathcal{X} : \eta(x) > m \text{ and } \eta'(x) > m \\ & \quad \text{or } \lfloor \eta(x) \rfloor = \lfloor \eta'(x) \rfloor \text{ and } \mathit{frac}(\eta(x)) = 0 \Leftrightarrow \mathit{frac}(\eta'(x)) = 0 \\ & \qquad \text{and for all } y \in \mathcal{X} \text{ with } \eta(y) \leq m \text{ and } \eta'(y) \leq m : \\ & \qquad\quad \mathit{frac}(\eta(x)) \leq \mathit{frac}(\eta(y)) \Leftrightarrow \mathit{frac}(\eta'(x)) \leq \mathit{frac}(\eta'(y)) \end{aligned}$$

Here, $\mathit{frac}(r)$ denotes the fractional part of a real number. It is easy to see that $\simeq_m$ is indeed an equivalence relation for any m. It is lifted to states of the TLTS $\mathcal{T}_\mathcal{A}$ in the most straight-forward way:

$$(\ell, \eta) \simeq_m (\ell', \eta') \quad \text{iff} \quad \ell = \ell' \text{ and } \eta \simeq_m \eta' \ .$$

We write $[\eta]_m$ for the equivalence class of η under $\simeq_m$ and likewise for $[(\ell, \eta)]_m$. When m is clear from the context we may also drop it and simply write $[\eta]$, resp. $[(\ell, \eta)]$.

This is not only an equivalence relation on the state space of $\mathcal{T}_{\mathcal{A}}$ but in fact even a congruence w.r.t. the labelling and discrete and delay transitions when $m \geq m(\mathcal{A})$. This is what makes it usable in order to abstract the uncountable state space of $\mathcal{T}_{\mathcal{A}}$ into a finite discrete state space as follows.

The *region graph* $\mathcal{R}_{\mathcal{A}}$ of the TA $\mathcal{A}$ is the LTS obtained as the quotient of $\mathcal{T}_{\mathcal{A}}$ under the congruence relation $\simeq_m$ (with $m = m(\mathcal{A})$), together with an additional collapse of delay transitions for different delays into a single "*some-delay*" value τ. Its components are as follows.

- The state space is $\{(\ell, [\eta]) \mid \ell \in L, \eta \in (\mathcal{X} \to \mathbb{R}^{\geq 0}), \eta \models \iota(\ell)\}$. The initial state is $(\ell_0, [\eta_0])$.
- Discrete transitions from one state to another are obtained by possibly delaying, then performing a discrete transition in the timed space and then possibly delaying again afterwards. We have

  $$(\ell,[\eta]) \to (\ell',[\eta']) \quad \text{if there are } d,d' \in \mathbb{R}^{\geq 0}, \hat{\eta}, \hat{\eta}' \text{ s.t. } (\ell,\eta) \xrightarrow{d_1} (\ell,\hat{\eta}) \to (\ell',\hat{\eta}') \xrightarrow{d_2} (\ell',\eta')$$

  for any $\ell, \ell' \in L$, $\eta, \eta' \in \mathcal{X} \to \mathbb{R}^{\geq 0}$.
- The propositional labelling is given as $\lambda(\ell,[\eta]) = \lambda(\ell,\eta) = \lambda(\ell)$.

We obtain the following proposition:

▶ **Proposition 4** ([3]). *Let $\mathcal{A}$ be a TA over n clocks with ℓ locations and of index m. Then $\mathcal{R}_{\mathcal{A}}$ is an (untimed) LTS of size $\ell \cdot 2^{\mathcal{O}(n(\log n + \log m))}$, i.e. exponential in $|\mathcal{A}|$, and there is a trace $s_0 \overset{d_0}{\Longrightarrow} s_1 \overset{d_1}{\Longrightarrow} \ldots$ in $\mathcal{T}_{\mathcal{A}}$ iff there is a path $[s_0] \to [s_1] \to \ldots$ in $\mathcal{R}_{\mathcal{A}}$.*

2.3 Temporal Logics

We recall the two most relevant temporal logics which form the basis for the definition of Timed Recursive CTL in Sect. 3: Timed CTL, the extension of pure CTL by operators to quantitatively speak about the passage of time, and Recursive CTL, the extension of CTL by a recursion operator which gives it much greater expressive power.

Timed Computation Tree Logic. As before, let *Prop* be a set of atomic propositions. Formulas of Timed CTL (TCTL) are given by the following grammar.

$$\varphi ::= q \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathtt{E}(\varphi\, \mathtt{U}^J\, \varphi) \mid \mathtt{A}(\varphi\, \mathtt{U}^J\, \varphi)$$

where $q \in \mathit{Prop}$ and J denotes a natural-number bounded interval in $\mathbb{R}^{\geq 0}$, i.e. it takes one of the forms $[n,m], (n,m], [n,m), (n,m), [n,\infty), (n,\infty)$ with $n,m \in \mathbb{N}$, $n \leq m$.

Other Boolean connectives are defined as abbreviations in the usual way: $\mathtt{tt} := q \vee \neg q$ for some $q \in \mathit{Prop}$, $\varphi \vee \psi := \neg(\neg\varphi \wedge \neg\psi)$, $\varphi \to \psi := \neg\varphi \vee \psi$, etc. Likewise, other familiar temporal operators can be obtained as abbreviations as well: $Q\mathtt{F}^J\varphi := Q(\mathtt{tt}\, \mathtt{U}^J\, \varphi)$ for $Q \in \{\mathtt{E},\mathtt{A}\}$, $Q\mathtt{G}^J\varphi := \neg\overline{Q}\mathtt{F}^J\neg\varphi$ where $\overline{\mathtt{E}} = \mathtt{A}$ and $\overline{\mathtt{A}} = \mathtt{E}$. We also use an intuitive way of the form $\oplus n$ with $\oplus \in \{\leq,<,\geq,>,=\}$ for denoting intervals when possible, for instance $\mathtt{EF}^{>2}q$ stands for $\mathtt{EF}^{(2,\infty)}q$, and $\mathtt{AG}^{\leq 5}q$ stands for $\mathtt{AG}^{[0,5]}q$.

Formulas of TCTL are interpreted over $\mathbb{R}^{\geq 0}$-timed transition systems $\mathcal{T} = (\mathcal{S}, \to, s_0, \lambda)$: $[\![\varphi]\!]^{\mathcal{T}}$ denotes the set of states in $\mathcal{T}$ in which φ holds, defined inductively as follows.

$$\begin{aligned}
[\![q]\!]^{\mathcal{T}} &:= \{s \mid q \in \lambda\} \\
[\![\varphi \wedge \psi]\!]^{\mathcal{T}} &:= [\![\varphi]\!]^{\mathcal{T}} \cap [\![\psi]\!]^{\mathcal{T}} \\
[\![\neg\varphi]\!]^{\mathcal{T}} &:= \mathcal{S} \setminus [\![\varphi]\!]^{\mathcal{T}} \\
[\![\mathtt{E}(\varphi\, \mathtt{U}^J\, \psi)]\!]^{\mathcal{T}} &:= \{s \in \mathcal{S} \mid \text{there is a trace } \pi = s, \ldots \text{ s.t. } \pi \models \varphi\, \mathtt{U}^J\, \psi\} \\
[\![\mathtt{A}(\varphi\, \mathtt{U}^J\, \psi)]\!]^{\mathcal{T}} &:= \{s \in \mathcal{S} \mid \text{for all traces } \pi = s, \ldots \text{ we have } \pi \models \varphi\, \mathtt{U}^J\, \psi\}
\end{aligned}$$

and for a non-zeno trace $\pi = s_0 \xRightarrow{d_0} s_1 \xRightarrow{d_1} s_2 \xRightarrow{d_2} \dots$ we have $\pi \models \varphi \, \mathtt{U}^J \, \psi$ iff

$$\exists i \geq 0, \exists d \in [0, d_i], \exists s' \text{ s.t. } s_i \xRightarrow{d} s' \text{ and } (\sum_{h=0}^{i} d_i) + d \in J \text{ and } \mathcal{T}, s' \models \psi \text{ and}$$
$$\forall j < i, \forall d' \in [0, d_j], \forall s' \text{ s.t. } s_j \xRightarrow{d'} s' \text{ we have } \mathcal{T}, s' \models \varphi \vee \psi \text{ and}$$
$$\forall d' \in [0, d), \forall s' \text{ s.t. } s_i \xRightarrow{d'} s' \text{ we have } \mathcal{T}, s' \models \varphi \vee \psi.$$

We write $\mathcal{T}, s \models \varphi$ if $s \in [\![\varphi]\!]^{\mathcal{T}}$ for arbitrary $s \in \mathcal{S}$, and also $\mathcal{T} \models \varphi$ if $\mathcal{T}, s_0 \models \varphi$.

The *model checking problem* for TCTL is the following: given a TA $\mathcal{A}$ and a TCTL formula φ, decide whether or not $\mathcal{R}_{\mathcal{A}} \models \varphi$.

▶ **Proposition 5** ([1, 17]). *The model checking problem for* TCTL *is* PSPACE-*complete, even for TA over a single clock.*

Temporal Logic with Recursion. We briefly present Recursive CTL (RecCTL), the other building block besides TCTL that make up Timed Recursive CTL, to be defined in the following section.

Let *Prop* be a set of atomic propositions. Formulas of RecCTL are obtained by addition of the recursion operator to the (purely modal part of) CTL. Let $\mathcal{V}_1 = \{x, y, \dots\}$ be a set of propositional variables and $\mathcal{V}_2 = \{\mathcal{F}, \dots\}$ be a set of so-called *recursion* variables. Formulas of RecCTL are given by the following grammar.

$$\varphi ::= q \mid x \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathtt{EX}\varphi \mid \Phi(\varphi, \dots, \varphi) \qquad \Phi ::= \mathcal{F} \mid \mathtt{rec}\, \mathcal{F}(x_1, \dots, x_k; y_1, \dots, y_h).\, \varphi$$

where $x, x_i, y_i \in \mathcal{V}_1$, $\mathcal{F} \in \mathcal{V}_2$.

A formula derived from φ in this grammar is called *propositional*, those derived from Φ are called *first-order*. Formulas are interpreted over (untimed) LTS $\mathcal{T}$ over some state set $\mathcal{S}$. A propositional formula φ denotes a *predicate* $[\![\varphi]\!]^{\mathcal{T}} \in 2^{\mathcal{S}}$, i.e. a set of states just like any CTL formula does; a first-order formula however denotes a *predicate transformer* $[\![\Phi]\!]^{\mathcal{T}} : 2^{\mathcal{S}} \times \ldots \times 2^{\mathcal{S}} \to 2^{\mathcal{S}}$.

We do not give the details of the formal semantics here. It suffices to note that the recursion operator is interpreted as the least fixpoint in the corresponding complete lattice of first-order functions called predicate transformers. For this to work seamlessly, i.e. these fixpoints to exist, we need to guarantee that any variable $\mathcal{F}$ is used monotonically in φ inside of $\mathtt{rec}\, \mathcal{F}(\vec{x}; \vec{y}).\, \varphi$ only. The fact that the logic features negation ($\neg$) and application ($\Phi(\varphi_1, \dots, \varphi_k)$) requires a slightly more involved syntactic criterion for monotonicity. In particular, in order to know whether some variable is used monotonically, it may be required to know this for others as well. This is why the formal parameters $x_1, \dots, x_k; y_1, \dots, y_h$ ($k, h \geq 0$) to a recursion operator are separated into two parts: those left of the divider ";" are used monotonically, those to the right are used antitonically. RecCTL employs a small type system to ensure these properties. For details we refer to the literature [10] or the next section where the machinery is carried out for full Timed Recursive CTL anyway.

An important result on RecCTL to notice here, as it will be used later on in Sect. 4, is decidability of its model checking problem.

▶ **Proposition 6** ([10]). *The model checking problem for* RecCTL *over finite LTS is* EXPTIME-*complete.*

3 Timed Recursive Computation Tree Logic

The formal syntax. Let *Prop* be a set of atomic propositions. The syntax of Timed Recursive CTL (TRCTL) is similar to that of RecCTL in that we distinguish between propositional and first-order formulas. We also need two kinds of variables again: first-order variables $\mathcal{V}_2 = \{\mathcal{F}, \mathcal{G}, \dots\}$ to form recursion anchors and propositional variables $\mathcal{V}_1 = \{x, y, \dots\}$ for formal parameters of recursive formulas. Formulas are then given by

$$\varphi ::= p \mid x \mid \varphi \wedge \psi \mid \neg\varphi \mid \mathtt{E}(\varphi\, \mathtt{U}^J\, \varphi) \mid \Phi(\varphi, \dots, \varphi) \qquad \Phi ::= \mathcal{F} \mid \mathtt{rec}\, \mathcal{F}(x_1, \dots, x_k).\, \varphi$$

where $p \in \mathit{Prop}$, $k \geq 0$, $x, x_1, \dots, x_k \in \mathcal{V}_1$, $\mathcal{F} \in \mathcal{V}_2$ and J denotes an interval in $\mathbb{R}^{\geq 0}$ with integer bounds as in the syntax for TCTL. We write $m(\varphi)$ to denote the largest constant that occurs in interval annotations of the Until operators in φ.

Note that CTL features the *Next* operators $Q\mathtt{X}$ as well as the *Until* operators $Q\mathtt{U}$. The former is missing in TCTL since there is no "next" moment in dense real time. RecCTL, however, seems to feature the *Next* but not the *Until*. This is simply because $Q(\varphi U \psi)$ is expressible via $Q\mathtt{X}$ using the recursion operator which is stronger than propositional fixpoints, i.e. $Q(\varphi \mathtt{U} \psi) \equiv (\mathtt{rec}\, \mathcal{F}().\, \psi \vee (\varphi \wedge Q\mathtt{X}\mathcal{F}()))()$, written more conveniently as $\mathtt{rec}\, \mathcal{F}.\, \psi \vee (\varphi \wedge Q\mathtt{X}\mathcal{F})$, along the lines of the embedding of CTL into the modal μ-calculus. This does not work for the time-bounded *Until* operator anymore. Hence, TRCTL features such the *Until* but not the *Next* operator just like TCTL.

Other Boolean and temporal operators are defined in the usual way, for instance $\mathtt{EF}^J\varphi := \mathtt{E}(\mathtt{tt}\, \mathtt{U}^J\, \varphi)$, $\mathtt{AG}^J\varphi := \neg\mathtt{EF}^J\neg\varphi$, etc. and will be used freely henceforth.

Vectorial form. The semantics of the recursion operator will be explained using least fixpoints in complete function lattices. This makes the Bekič Lemma [7] available which allows formulas with mutual dependencies between recursion variables to be written down in a more readable form. A formula in *vectorial form*, cf. [4] for its use in $\mathcal{L}_\mu$, is a

$$\mathtt{rec}_{\,i} \begin{pmatrix} \mathcal{F}_1(x_1, \dots, x_k) & . & \varphi_1 \\ & \vdots & \\ \mathcal{F}_n(x_1, \dots, x_k) & . & \varphi_n \end{pmatrix} (\psi_1, \dots, \psi_k).$$

Informally, this defines not just one but several functions $\mathcal{F}_1, \dots, \mathcal{F}_n$ which may all depend on each other in a mutually recursive way formalised in the φ_j's. In the end, the function named by $\mathcal{F}_i$ is applied to the initial arguments $\psi_1, \dots, \psi_k$.

Well-formed formulas. Not every formula generated by the formal syntax as introduced above is well-formed. For instance, when a recursion formula has k formal parameters as in $\Phi = \mathtt{rec}\, \mathcal{F}(x_1, \dots, x_k).\, \varphi$, it should only be applied to a tuple of k arguments as in $\Phi(\varphi_1, \dots, \varphi_k)$. The same goes for any subformula of the form $\mathcal{F}(\psi_1, \dots, \psi_k)$.

More importantly, a well-defined semantics can only be given to recursive formulas when the recursion variable occurs monotonically in the defining fixpoint formula only. Here we refrain from giving further formalities in terms of a type system that ensures well-formedness. For what follows, it suffices to work with the intuitive notion of "occurring only monotonically". For formal details we refer to [10] where the notion of well-formedness is made precise for RecCTL. The same principles can be applied here to this real-time extension of this logic.

The formal semantics. As with TCTL, (propositional) formulas of TRCTL are interpreted in states of an TLTS $\mathcal{T} = (\mathcal{S}, \rightarrow, s_0, \lambda)$. In fact, it suffices to extend the semantics of TCTL to those operators (propositional variables and first-order formulas) which do not already occur in the syntax of TCTL. Due to the presence of variables, we need variable interpretations ϑ in order to explain the meaning of a formula inductively. Such a ϑ maps propositional variables to sets of states, $\vartheta(x) \in 2^{\mathcal{S}}$ for $x \in \mathcal{V}_1$, and first-order variables to functions of corresponding arity over these: $\vartheta(\mathcal{F}) : 2^{\mathcal{S}} \times \ldots \times 2^{\mathcal{S}} \rightarrow 2^{\mathcal{S}}$.

These functions form a complete Boolean lattice ordered pointwise, hence least fixpoints of monotone functionals mapping one such function to another exist due to the Knaster-Tarski Theorem [20]. These are used to explain the meaning of the recursion operator. For details, we refer to the exposition on RecCTL [10] or on HFL [21] that this idea goes back to – the only difference is that there, $\mathcal{S}$ is the state space of an untimed LTS rather than a TLTS.

A propositional formula φ gives rise to a set $[\![\varphi]\!]^{\mathcal{T}}_{\vartheta}$ of states that satisfy it under the variable interpretation ϑ, and similarly for first-order formulas and corresponding first-order functions. The semantics is defined as follows. The clauses presented for $\varphi \in$ TCTL apply here as well under the provision that each $[\![\cdot]\!]^{\mathcal{T}}$ is replaced by $[\![\cdot]\!]^{\mathcal{T}}_{\vartheta}$. Additionally,

$$[\![x]\!]^{\mathcal{T}}_{\vartheta} := \vartheta(x) \quad \text{for } x \in \mathcal{V}_1 \;, \qquad [\![\Phi(\varphi_1, \ldots, \varphi_k)]\!]^{\mathcal{T}}_{\vartheta} := [\![\Phi]\!]^{\mathcal{T}}_{\vartheta}([\![\varphi_1]\!]^{\mathcal{T}}_{\vartheta}, \ldots, [\![\varphi_k]\!]^{\mathcal{T}}_{\vartheta})$$

for propositional formulas, while for first-order formulas we set $[\![\mathcal{F}]\!]^{\mathcal{T}}_{\vartheta} := \vartheta(\mathcal{F})$ if $\mathcal{F} \in \mathcal{V}_2$ and

$$[\![\texttt{rec}\,\mathcal{F}(x_1, \ldots, x_k).\,\varphi]\!]^{\mathcal{T}}_{\vartheta} :=$$
$$\bigsqcap\{f : (2^{\mathcal{S}})^k \rightarrow 2^{\mathcal{S}} \mid \forall S_1, \ldots, S_k : [\![\varphi]\!]^{\mathcal{T}}_{\vartheta[\mathcal{F}\mapsto f, x_1\mapsto S_1, \ldots, x_k \mapsto S_k]} \subseteq f(S_1, \ldots, S_k)\}$$

where $\sqcap$ denotes the point-wise intersection for functions: $(f \sqcap g)(S) := f(S) \cap g(S)$.

Examples. We illustrate the use of the recursion operator in TRCTL to form structurally complex properties which cannot be expressed in TCTL. We refer to [10] for more exposition regarding RecCTL. It is helpful, though, to imagine the recursive formulas to be unrolled so that new arguments are being built and these to be plugged in for the formal parameters.

▶ **Example 7.** Consider $\varphi_{\mathsf{ag}} := \big(\texttt{rec}\,\mathcal{F}(x, y).\,(x \wedge \neg y) \vee \mathcal{F}(\texttt{AF}^{\leq 3}x, \texttt{AF}^{\leq 2y})\big)(p, p)$. Unrolling of the recursion shows that it is equivalent to

$$\bigvee_{i\geq 0} \underbrace{\texttt{AF}^{\leq 3}\texttt{AF}^{\leq 3}\ldots\texttt{AF}^{\leq 3}}_{i \text{ times}} p \wedge \neg \underbrace{\texttt{AF}^{\leq 2}\texttt{AF}^{\leq 2}\ldots\texttt{AF}^{\leq 2}}_{i \text{ times}} p$$

stating "there is an i such that on all paths we see i occurrences of p in distances of at most 3 seconds, but not in distances of at most 2 seconds." Negating this to $\neg\varphi_{\mathsf{ag}}$ then formalises "whenever it is possible to see p in distances of 3 seconds i times on a path, then it is also possible to do so in distances of 2 seconds on some path." This is inspired by the formalisation of assume-guarantee properties in HFL [21].

▶ **Example 8.** Note that the context-free grammar G with productions

$$F_1 \rightarrow F_2F_3 \;, \quad F_2 \rightarrow \mathsf{out} \mid \mathsf{in}F_2F_2 \;, \quad F_3 \rightarrow \varepsilon \mid \mathsf{in}F_3 \mid \mathsf{out}F_3$$

generates the set of all $\mathsf{in}, \mathsf{out}$-sequences such that some prefix contains more out's than in's. It can be seen as the set of all finite computations in which a buffer underflow occurs. Now consider the TRCTL formula

$$\varphi_{\mathsf{buf}} := \texttt{rec}\,_1 \begin{pmatrix} \mathcal{F}_1(x) & . & \mathcal{F}_2(\mathcal{F}_3(x)) \\ \mathcal{F}_2(x) & . & \texttt{E}(p_{\mathsf{out}}\,\texttt{U}^{\geq 1}\,x) \vee \texttt{E}(p_{\mathsf{in}}\,\texttt{U}^{\geq 1}\,\mathcal{F}_2(\mathcal{F}_2(x))) \\ \mathcal{F}_3(x) & . & x \vee \texttt{E}(p_{\mathsf{in}}\,\texttt{U}^{\geq 1}\,\mathcal{F}_3(x)) \vee \texttt{E}(p_{\mathsf{out}}\,\texttt{U}^{\geq 1}\,\mathcal{F}_3(x)) \end{pmatrix} (\texttt{tt}) \;.$$

It states that there is a path forming a buffer underflow, provided that consecutive traversal of states satisfying p_{in}, resp. p_{out} for at least 1sec are taken as input/output actions for the buffer. Then $\neg\varphi_{\mathsf{buf}}$ formalises absence of such underflows under this interpretation.

4 The Complexity of Model Checking

In this section we show that the model checking problem for TRCTL is 2-EXPTIME-complete. We begin with the upper bound.

Upper Bound. We follow the same principles as usual decidability proofs for problems on TA, using so-called *untiming* constructions like the one for the region graph. Let $\varphi \in$ TRCTL and $\mathcal{A}$ be a TA not using the clock $\mathtt{z}$. Let $\mathcal{A}^{\mathtt{z}}$ result from it by simply adding the clock $\mathtt{z}$ to it (which is not accessed or manipulated anywhere). Let $\mathcal{R}_{\mathcal{A}^{\mathtt{z}}}$ be the corresponding region graph. Note that its states are of the form $(\ell, [\eta])$, where ℓ is a location of $\mathcal{A}$ and $[\eta]$ is a region, i.e. an equivalence class of a clock evaluation η that is also defined on $\mathtt{z}$ now.

We construct a new LTS $\mathcal{R}^{\varphi}_{\mathcal{A}^{\mathtt{z}}}$ by extending $\mathcal{R}_{\mathcal{A}^{\mathtt{z}}} = (\mathcal{S}, \to, s_0, \lambda)$ in the following two ways:

- For each state $(\ell, [\eta])$ and each $c \leq m(\varphi)$, add new proposition $p_{\mathtt{z}\oplus c}$ to $\lambda(\ell, [\eta])$ if $\eta \models \mathtt{z}\oplus c$ for $\oplus \in \{\leq, <, \geq, >, =\}$.
- For each state $(\ell, [\eta])$ introduce a new state $s_{\ell,[\eta]}$ with the sole label $\{r_{\mathtt{z}}\}$, and add transitions $(\ell, [\eta]) \to s_{\ell,[\eta]} \to (\ell, [\eta|_{\{\mathtt{z}\}}])$.

This has introduced new traces in this region graph: at any moment, it is now possible to reset clock $\mathtt{z}$, and then continue some original trace. Moreover, the resetting of $\mathtt{z}$ becomes visible through the traversal of a state that satisfies $r_{\mathtt{z}}$. Since $\mathtt{z}$ is not used in $\mathcal{A}$, this is the only way that it is being reset. Moreover, the values of $\mathtt{z}$ are also accessible through propositions of the form $p_{\mathtt{z}\oplus c}$.

Next we rewrite φ so that it can make use of these propositions. The formula $\varphi^{\mathtt{z}}$ results from φ by replacing each subformula of the form

- $\mathtt{E}(\psi_1\, \mathtt{U}^{[c,d]}\, \psi_2)$ by $\mathtt{EX}(r_{\mathtt{z}} \wedge \mathtt{EXE}((\neg r_{\mathtt{z}} \wedge \psi_1)\, \mathtt{U}\, (\neg r_{\mathtt{z}} \wedge p_{\mathtt{z}\geq c} \wedge p_{\mathtt{z}\leq d} \wedge \psi_2)))$, resp.
- $\mathtt{A}(\psi_1\, \mathtt{U}^{[c,d]}\, \psi_2)$ by $\mathtt{EX}(r_{\mathtt{z}} \wedge \mathtt{EXA}((\neg r_{\mathtt{z}} \to \psi_1)\, \mathtt{U}\, (\neg r_{\mathtt{z}} \to p_{\mathtt{z}\geq c} \wedge p_{\mathtt{z}\leq d} \wedge \psi_2)))$.

For open intervals on one side, the p-propositions are amended accordingly to $p_{\mathtt{z}>c}$ etc.

The following forms the basis of an exponential reduction of TRCTL model checking to RecCTL model checking.

▶ **Lemma 9.** *Let $\mathcal{A}$ be a TA, $\varphi \in$ TRCTL.*

(a) *$\varphi^{\mathtt{z}}$ is a formula of (untimed) RecCTL and is constructible in time $\mathcal{O}(|\varphi|)$.*

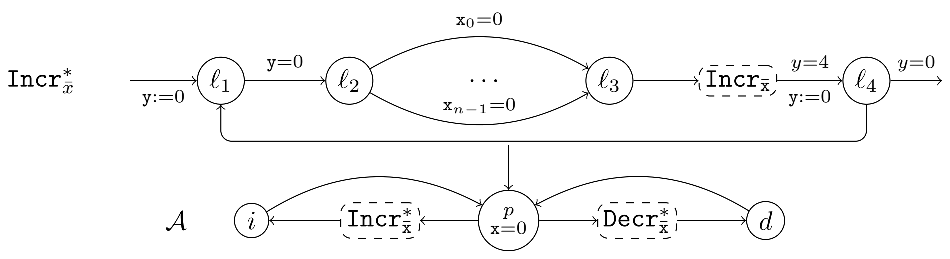
(b) *$\mathcal{R}^{\varphi}_{\mathcal{A}^{\mathtt{z}}}$ is an (untimed) LTS of size at most (singly) exponential in $|\mathcal{A}|$ and $m(\varphi)$ and also constructible in such time.*

(c) *$\mathcal{T}_{\mathcal{A}} \models \varphi$ iff $\mathcal{R}^{\varphi}_{\mathcal{A}^{\mathtt{z}}} \models \varphi^{\mathtt{z}}$.*

Parts (a) and (b) are easily checked. Part (c) can be proved by simple induction on the structure of φ using Prop. 4.

▶ **Theorem 10.** *The model checking problem for* TRCTL *over TA is decidable in* 2-EXPTIME*.*

Proof. Let a TA $\mathcal{A}$ and a TRCTL formula φ be given. To check whether $\mathcal{T}_{\mathcal{A}} \models \varphi$ holds, first construct $\mathcal{R}^{\varphi}_{\mathcal{A}^{\mathtt{z}}}$ and $\varphi^{\mathtt{z}}$. According to Lemma 9, this can be done in exponential time, and it suffices to check whether or not $\mathcal{R}^{\mathtt{z}}_{\mathcal{A}} \models \varphi^{\mathtt{z}}$ holds. According to Prop. 6, the latter can be solved in exponential time. Altogether, this gives a doubly exponential upper bound on the time complexity of model checking TRCTL over TA. ◀



$\mathtt{Incr}^*_{\bar{x}}$
y:=0
ℓ_1
y=0
ℓ_2
x_0=0
. . .
x_{n-1}=0
ℓ_3
$\mathtt{Incr}_{\bar{x}}$
y=4
y:=0
ℓ_4
y=0
$\mathcal{A}$
i
$\mathtt{Incr}^*_{\bar{x}}$
p
x=0
$\mathtt{Decr}^*_{\bar{x}}$
d

Figure 2 A gadget for arbitrary incrementation (upper part), the TA $\mathcal{A}$ (lower part).

Encoding Large Numbers. We now want to encode numbers in the range $[2^{2^n}]$ in TRCTL. We have seen in Obs. 3 how numbers in the range $[2^n]$ can be encoded in small counters and that there is a polynomially-sized gadget such that passing through (the TLTS generated by) that gadget increases the value encoded in the counter by 1. We now extend this to larger numbers. Clearly, increasing the number of clocks involved is not sufficient unless we use exponentially many clocks. However, note that a TA already generates an exponentially large TLTS through the values of its associated clocks, even after the region graph abstraction. This stems from the fact that both locations and clock values in a TA contribute to the TLTS, and with multiple clocks present, one location may generate many TLTS states. Informally put, the question whether some proposition holds at some location, and for which *small counter values*, already has exponentially many possible answers. We use small counters to represent the bits in binary numbers of exponential width, exactly enough to represent numbers of doubly exponential size. Hence, *large counters* are sets of TLTS states that agree on the location component, say ℓ. What varies are the clock values. Intuitively, we shall consider a bit $b \in [2^n]$ set in the representation of a number through such a set, if said set contains the state (ℓ, η) such that η is a small counter with $\langle \eta \rangle = b$.

▶ **Definition 11.** *Consider the TLTS $\mathcal{T}_\mathcal{A}$ generated by the TA $\mathcal{A}$, depicted in the upper part of Fig. 2. Let $\bar{\mathtt{x}} = (\mathtt{x}_0, \ldots, \mathtt{x}_{n-1})$. A set S of states of the form (p, η), where η is a small $\bar{\mathtt{x}}$-counter, is called a* large $(\bar{\mathtt{x}})$-counter. *Its* value *is $\langle S \rangle = \sum_{i=0}^{2^n-1} b_i \cdot 2^i$ where $b_i = 1$ if the small counter with value i belongs to S, and $b_i = 0$ otherwise.*

For example, the empty set encodes $m = 0$ since $(p, \eta) \in S$ for no η. Any set that contains all states of the form (p, η) encodes $m = 2^{2^n} - 1$, since $(p, \eta) \in S$ for all small counters η. On the other hand, a set that contains only (p, η) such that $\langle \eta \rangle = 0$ encodes $m = 1$, while a set that contains all states but those with $\langle \eta \rangle = 0$ encodes $m = 2^{2^n} - 2$. Note that the first two sets are expressible in TRCTL via $\mathtt{ff}$ and $\mathtt{tt}$. In fact, every TRCTL-formula of the form $p \wedge \psi$ defines a large counter, whence we write $\varphi = \langle m \rangle$ for a formula that defines a counter that encodes m.

Now assume that the gadget $\mathtt{Decr}^*_{\bar{\mathtt{x}}}$ is obtained from $\mathtt{Incr}^*_{\bar{\mathtt{x}}}$, by replacing $\mathtt{Incr}_{\bar{\mathtt{x}}}$ by $\mathtt{Decr}_{\bar{\mathtt{x}}}$ and by replacing the test for 0 by a test for 1. We use the following lemma.

▶ **Lemma 12.** *Consider $\mathcal{T}_\mathcal{A}$ where $\mathcal{A}$ is the TA in the lower part of Fig. 2. Let (p, η) and (p, η') be states in $\mathcal{T}_\mathcal{A}$ such that η, η' are small counters. If (p, η') is reachable from (p, η) by passing exactly once through the sub-TLTS generated by the gadget $\mathtt{Incr}^*_{\bar{\mathtt{x}}}$ and η encodes $m \in 2^n - 1$, then η' encodes a value in in $\{m+1, \ldots, 2^n - 1\}$. Moreover, for each such $m' \in \{m+1, \ldots, 2^n - 1\}$, there is a path through the gadget such that the η' encodes m', and there is no such path if $m = 2^n - 1$. The analogue holds for $\mathtt{Decr}^*_{\bar{\mathtt{x}}}$.*

Proof. In $\mathtt{Incr}^*_{\bar{\mathtt{x}}}$, time flows only between ℓ_3 and ℓ_4, namely for exactly 4 units in $\mathtt{Incr}_{\bar{\mathtt{x}}}$ (cf. Obs. 3). Now assume that the sub-TLTS generated by $\mathtt{Incr}^*_{\bar{\mathtt{x}}}$ is entered from (p, η). When passing directly from ℓ_3 to ℓ_4, the value encoded in η is increased by 1. Moreover, passing from ℓ_1 to ℓ_3 and, hence to the end of the gadget, is only possible if at least one of the $\mathtt{x}_i$ is 0, i.e. if η encodes a number less than $2^n - 1$. Finally, it is not hard to see that the return path from ℓ_4 to ℓ_1 can be taken without making it impossible to leave the gadget as long as at least one of the $\mathtt{x}_i$ is 0, whence the gadget can be left with any value in the range $\{m + 1, \dots, 2^n - 1\}$ encoded in η. ◀

We use Lem. 12 to manipulate large counters by TRCTL-formulas. Recall that, when incrementing a binary number m given as $b_0 \cdots b_{n-1}$ with least significant bit left, a bit is set in the representation of $m + 1$ iff it is already set in (the representation) of m, and there is a bit of lesser significance that is not set, or if it is not set in m, but all bits of lesser significance are. We now apply this to large counters. Let m be a number encoded in a large counter S. Then the set that encodes $m + 1$ comprises exactly all states (p, η) such that (p, η) is already included in X and there is η' with $\langle \eta'_{\bar{\mathtt{x}}} \rangle < \langle \eta_{\bar{\mathtt{x}}} \rangle$ such that $(p, \eta) \notin S$, and all such (p, η) such that $(p, \eta) \notin S$, but for all (p, η') with $\langle \eta'_{\bar{\mathtt{x}}} \rangle < \langle \eta_{\bar{\mathtt{x}}} \rangle$ we have $(p, \eta') \in S$. This, incrementation, and a test for 0 are expressed by the following formulas:

$$inc(X) = (X \land \mathtt{E}(\neg(p \lor i)\ \mathtt{U}^{>0}\ (p \land \neg X))) \lor (\neg X \land \mathtt{A}(\neg p\, \mathtt{U}^{>0}\ (i \lor p \land X)))$$
$$dec(X) = (X \land \mathtt{E}(\neg(p \lor i)\ \mathtt{U}^{>0}\ (p \land X))) \lor (\neg X \land \mathtt{A}(\neg p\, \mathtt{U}^{>0}\ (i \lor p \land \neg X)))$$
$$eq_0(X) = \neg X \land \mathtt{A}(\neg p\, \mathtt{U}^{>0}\ (p \land \neg X))$$

▶ **Lemma 13.** *Let S be a large counter in $\mathcal{T}_\mathcal{A}$ such that S encodes m for $m \in [2^{2^n}]$. Then $inc(S)$ encodes $m+1$ modulo 2^{2^n} and $dec(X)$ encodes $m-1$ modulo 2^{2^n}. Finally, $(p, \eta) \in eq_0(S)$ iff S encodes 0.*

Proof. We show the claim for $inc()$. Let S encode m. The left part of the disjunction concerns the case of a bit that is already set in the representation of m, i.e. $(p, \eta) \in S$ such that $\langle \eta \rangle$ is some $k \in [2^n]$. Then (p, η) is in the representation of $m + 1$ modulo 2^{2^n} iff there is $k' \in [k]$ such that $(p, \eta') \notin S$ if η' encodes k'. This is formalised in the $\mathtt{EU}$-formula, which requires the existence of a path of length different than 0 where, at the first occurrence of p after the initial state, $\neg S$ holds. Moreover, since i can also not hold, that path must go exactly once through the gadget $\mathtt{Decr}^*_{\bar{\mathtt{x}}}$. By Lem. 12, respectively its analogue for $\mathtt{Decr}^*_{\bar{\mathtt{x}}}$, we obtain that, for each $k' < k$, there is a path through this gadget such that $\bar{\mathtt{x}}$ encodes k' after leaving it, and no other paths through this gadget exist. Hence, the $\mathtt{EU}$-formula only holds if there is $k' \in [k]$ representing the bit of lesser significance than k not set in the representation of m. The other disjunct follows the same pattern, except here, the $\mathtt{AU}$-formula formalises the forall quantifier in the logic of binary incrementation, and the proposition i moves to formalise that each path either goes through i and, hence, is of no concern, or ends up in the location p such that the corresponding lower bit is set.

The formula for decrementation follows the same logic. Finally, a set S encodes 0 iff it contains no states (p, η) with η of any kind. In other words, (p, η) is in (the semantics of) $eq_0(S)$ iff it is not in S and, no matter which path is taken through one of the two gadgets, one ends up outside of S upon reaching the location p for the first time, i.e. no matter how the clocks in $\bar{\mathtt{x}}$ are changed, one cannot reach S. ◀

We add that the following are also expressible via the formulas given below: The fact that a large counter encodes 1, the fact that it encodes a number greater than 0 or 1, and the fact that it encodes a number less than $2^{2^n} - 1$:

$$eq_1(X) = eq_0(dec(X)) \quad eq_{2^{2^n}-1}(X) = eq_0(inc(X)) \quad gt_1(X) = \neg eq_0(X) \land \neg eq_0(dec(X))$$
$$gt_0(X) = \neg eq_0(X) \qquad lt_{2^{2^n}-1}(X) = \neg eq_0(dec(X))$$

Lower Bound. A matching lower bound can be obtained by a polynomial reduction from problem stated in Prop. 1. We construct, given such a DTM $\mathcal{M}$ and an $n \in \mathbb{N}$, a TA $\mathcal{A}_{\mathcal{M},n}$ and a TRCTL formula $\varphi_{\mathcal{M},n}$ each of polynomial size in $|\mathcal{M}|$ and n, such that $\mathcal{A}_{\mathcal{M},n} \models \varphi_{\mathcal{M},n}$ iff there is a 2^{2^n}-certificate for $\mathcal{M}$ and n. Given the previous work on encodings or large numbers, the existence of such a certificate is easily defined in TRCTL, as we will see below.

▶ **Theorem 14.** *The model checking problem for* TRCTL *over Timed Automata is* 2-EXPTIME*-hard.*

Proof. Let $\mathcal{M}$ and n be given. Let $\hat{\Gamma} = \{a_1, \dots, a_m\}$ and $\hat{\delta}$ be as defined in Sect. 2.1, resulting from $\mathcal{M}$'s state set, tape alphabet and transition function. Let

$$\varphi_{\mathcal{M},n} := \texttt{rec}_{(q_{\mathsf{acc}},\Box)} \begin{pmatrix} & \vdots & \\ \mathcal{C}_{a_i}(t,s) & . & chk_{a_i}(t,s) \lor \bigvee\limits_{(b_1,b_2,b_3,a_i)\in\hat{\delta}} nxt_{b_1,b_2,b_3}(t,s) \\ & \vdots & \end{pmatrix} (\langle 2^{2^n}-1\rangle, \langle 0\rangle)$$

where

$$chk_a(t,s) := \begin{cases} eq_0(s) \lor eq_{2^{2^n}-1}(s) & \text{, if } a = \# \\ eq_0(t) \land eq_1(s) & \text{, if } a = (q_0,\Box) \\ gt_1(s) \land lt_{2^{2^n}-1}(s) & \text{, if } a = \Box \\ \mathtt{ff} & \text{, otherwise} \end{cases}$$

$$nxt_{b_1,b_2,b_3}(t,s) := gt_0(t) \land gt_0(s) \land lt_{2^{2^n}-1}(s) \land \\ \mathcal{C}_{b_1}(dec(t),dec(s)) \land \mathcal{C}_{b_2}(dec(t),s) \land \mathcal{C}_{b_3}(dec(t),inc(s))$$

Let $\mathcal{A}$ be the TA from Fig. 2. Then $\mathcal{T}_{\mathcal{A}} \models \varphi_{\mathcal{M},n}$ iff there is a 2^{2^n}-certificate for $\mathcal{M}$. This follows from the fact that the definition of the $\mathcal{C}$ mirrors the pattern of the certificate *Cert* described in Sec. 2.1. The arithmetic used is described above. Note that $\varphi_{\mathcal{M},n}$ is well-defined w.r.t. monotonocity of the $\mathcal{C}$ since all of them occur only positively in $nxt_{b_1,b_2,b_3}(t,s)$. The variables s and t occur both positively and negatively but they are not recursion variables, so this is unproblematic. ◀

5 Conclusion & Further Work

We have introduced Timed Recursive Temporal Logic (TRCTL) and shown that its model-checking problem is 2-EXPTIME-complete. Its satisfiability problem is undecidable, this is inherited from Recursive Temporal Logic [10]. TRCTL is strictly stronger in expressive power than its two constituent parts RecCTL and TCTL since either can express properties that the other cannot, namely higher-order properties (cf. [10]) or real-time properties. A fine-grained analysis of the expressive power of TRCTL, i.e. which properties of e.g. TLTS become accessible that are not accessible in TCTL, is still to be done. It should be noted that our lower bounds already hold in the setting with just two clocks, the constructions from [17] carry over with few adaptations.

Further research concerns two angles: practicability and extensions in expressive power. With respect to the former, the 2-EXPTIME-complete model checking problem might seem prohibitive, yet higher-order algorithms are open to optimisations that can yield surprisingly competitive algorithms [9, 14]. The latter angle includes straightforward extensions such as propositions that test for the value of some clock that are unlikely to require new methods, but also more intricate ones like diagonal constraints etc. which, of course, are also likely to lead to undecidability [8].

References

1 R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. In *Proc. 5th Ann. IEEE Symp. on Logic in Computer Science, LICS'90*, pages 414–427. IEEE Computer Society Press, 1990. doi:10.1109/LICS.1990.113766.

2 R. Alur, C. Courcoubetis, and D. Dill. Model-checking in dense real-time. *Inform. and Comp.*, 104(1):2–34, 1993. doi:10.1006/inco.1993.1024.

3 R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994. doi:10.1016/0304-3975(94)90010-8.

4 A. Arnold and D. Niwiński. *Rudiments of μ-calculus*, volume 146 of *Studies in Logic and the Foundations of Mathematics.* North-Holland, 2001.

5 R. Axelsson, M. Lange, and R. Somla. The complexity of model checking higher-order fixpoint logic. *Logical Methods in Computer Science*, 3:1–33, 2007. doi:10.2168/LMCS-3(2:7)2007.

6 C. Baier and J.-P. Katoen. *Principles of model checking.* MIT Press, 2008.

7 H. Bekić. *Programming Languages and Their Definition, Selected Papers*, volume 177 of *LNCS*. Springer, 1984.

8 P. Bouyer, F. Laroussinie, N. Markey, J. Ouaknine, and J. Worrell. Timed temporal logics. In *Models, Algorithms, Logics and Tools - Essays Dedicated to Kim Guldstrand Larsen on the Occasion of His 60th Birthday*, volume 10460 of *LNCS*, pages 211–230. Springer, 2017. doi:10.1007/978-3-319-63121-9_11.

9 F. Bruse, J. Kreiker, M. Lange, and M. Sälzer. Local higher-order fixpoint iteration. In *Proc. 11th Int. Symp. on Games, Automata, Logics, and Formal Verification, GandALF'20*, volume 326 of *EPTCS*, pages 97–113, 2020. doi:10.4204/EPTCS.326.7.

10 F. Bruse and M. Lange. Temporal logic with recursion. In *Proc. 27th Int. Symp. on Temporal Representation and Reasoning, TIME'20*, volume 178 of *LIPIcs*, pages 6:1–6:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.TIME.2020.6.

11 A. K. Chandra, D. Kozen, and L. J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981. doi:10.1145/322234.322243.

12 E. A. Emerson and E. M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2(3):241–266, 1982. doi:10.1016/0167-6423(83)90017-5.

13 D. Harel, A. Pnueli, and J. Stavi. Propositional dynamic logic of nonregular programs. *Journal of Computer and System Sciences*, 26(2):222–243, 1983. doi:10.1016/0022-0000(83)90014-4.

14 Y. Hosoi, N. Kobayashi, and T. Tsukada. A type-based HFL model checking algorithm. In *Proc. 17th Asian Symp. on Programming Languages and Systems, APLAS'19*, volume 11893 of *NCS*, pages 136–155. Springer, 2019. doi:10.1007/978-3-030-34175-6_8.

15 M. Lange. Model checking propositional dynamic logic with all extras. *Journal of Applied Logic*, 4(1):39–49, 2005. doi:10.1016/j.jal.2005.08.002.

16 M. Lange and C. Stirling. Model checking fixed point logic with chop. In *Proc. 5th Conf. on Foundations of Software Science and Computation Structures, FOSSACS'02*, volume 2303 of *LNCS*, pages 250–263. Springer, 2002. doi:10.1007/3-540-45931-6_18.

17 F. Laroussinie, N. Markey, and P. Schnoebelen. Model checking timed automata with one or two clocks. In *Proc. 15th Int. Conf. on Concurrency Theory, CONCUR'04*, volume 3170 of *LNCS*, pages 387–401. Springer, 2004. doi:10.1007/978-3-540-28644-8_25.

18 M. Müller-Olm. A modal fixpoint logic with chop. In *Proc. 16th Symp. on Theoretical Aspects of Computer Science, STACS'99*, volume 1563 of *LNCS*, pages 510–520. Springer, 1999. doi:10.1007/3-540-49116-3_48.

19 A. Pnueli. The temporal logic of programs. In *Proc. 18th Symp. on Foundations of Computer Science, FOCS'77*, pages 46–57, Providence, RI, USA, 1977. IEEE. doi:10.1109/SFCS.1977.32.

20 A. Tarski. A lattice-theoretical fixpoint theorem and its application. *Pacific Journal of Mathematics*, 5:285–309, 1955. doi:10.2140/pjm.1955.5.285.

21 M. Viswanathan and R. Viswanathan. A higher order modal fixed point logic. In *CONCUR'04*, volume 3170 of *LNCS*, pages 512–528. Springer, 2004. doi:10.1007/978-3-540-28644-8_33.