Time-Bounded Verification[⋆]

Joël Ouaknine[1], Alexander Rabinovich[2], and James Worrell[1]

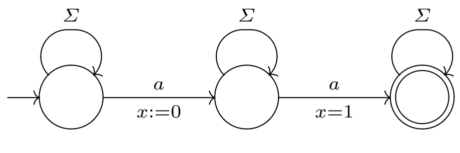
[1] Oxford University Computing Laboratory, UK
{joel,jbw}@comlab.ox.ac.uk
[2] School of Computer Science, Tel Aviv University, Israel
rabinoa@post.tau.ac.il

Abstract. We study the decidability and complexity of verification problems for timed automata over time intervals of fixed, bounded length. One of our main results is that time-bounded language inclusion for timed automata is 2EXPSPACE-complete. We also investigate the satisfiability and model-checking problems for Metric Temporal Logic (MTL), as well as monadic first- and second-order logics over the reals with order and the $+1$ function ($\mathsf{FO}(<, +1)$ and $\mathsf{MSO}(<, +1)$ respectively). We show that, over bounded time intervals, MTL satisfiability and model checking are EXPSPACE-complete, whereas these problems are decidable but non-elementary for the predicate logics. Nevertheless, we show that MTL and $\mathsf{FO}(<, +1)$ are equally expressive over bounded intervals, which can be viewed as an extension of Kamp's well-known theorem to metric logics.
It is worth recalling that, over unbounded time intervals, the various problems listed above are all undecidable.

1 Introduction

Timed automata were introduced by Alur and Dill in [2] as a natural and versatile model for real-time systems. They have been widely studied ever since, both by practitioners and theoreticians. A celebrated result concerning timed automata, which originally appeared in [1] in a slightly different context, is the PSPACE decidability of the *language emptiness* (or *reachability*) problem.

Unfortunately, the *language inclusion* problem—given two timed automata $\mathcal{A}$ and $\mathcal{B}$, is every timed word accepted by $\mathcal{A}$ also accepted by $\mathcal{B}$?—is known to be undecidable. A related phenomenon is the fact that timed automata are not closed under complementation. For example, the automaton below accepts every timed word in which there are two a-events separated by exactly one time unit.



⋆ This work was partially supported by the ESF Research Networking Programme Games and the UK's EPSRC.

The complement language consists of all timed words in which no two a-events are separated by precisely one time unit. Intuitively, this language is not expressible by a timed automaton, since such an automaton would need an unbounded number of clocks to keep track of the time delay from each a-event. (We refer the reader to [17] for a formal treatment of these considerations.)

The undecidability of language inclusion severely restricts the algorithmic analysis of timed automata, both from a practical and theoretical perspective, as many interesting questions can be phrased in terms of language inclusion. Over the past decade, several researchers have therefore attempted to circumvent this obstacle by investigating language inclusion, or closely related concepts, under various assumptions and restrictions. Among others, we note the use of (i) topological restrictions and digitisation techniques: [14, 10, 31, 28]; (ii) fuzzy semantics: [13, 15, 30, 7]; (iii) determinisable subclasses of timed automata: [4, 37]; (iv) timed simulation relations and homomorphisms: [43, 26, 23]; and (v) restrictions on the number of clocks: [32, 11].

The undecidability of language inclusion, first established in [2], derives from the undecidability of an even more fundamental problem, that of *universality*: does a given timed automaton accept every timed word? The proof of undecidability of universality in [2] uses in a crucial way the unboundedness of the time domain. Roughly speaking, this allows one to encode arbitrarily long computations of a Turing machine. On the other hand, many verification questions are naturally stated over bounded time domains. For example, a run of a communication protocol might normally be expected to have an *a priori* time bound, even if the total number of messages exchanged is potentially unbounded. Thus numerous researchers have considered the problem of time-bounded verification in the context of real-time systems [39, 8, 22]. This leads us to the question of the decidability of the time-bounded version of the language inclusion problem for timed automata. This problem asks, given timed automata $\mathcal{A}$ and $\mathcal{B}$ and a time bound N, whether all finite timed words of duration at most N that are accepted by $\mathcal{A}$ are also accepted by $\mathcal{B}$. One of the main results of this paper is that the time-bounded language inclusion problem is 2EXPSPACE-complete. It is worth noting that, since we are working with a dense model of time, time-bounded runs of a given automaton may contain arbitrarily many events. Moreover, the restriction to time boundedness does not alter the fact that timed automata are not closed under complement, and hence classical techniques for language inclusion do not trivially apply.

A second line of investigation in this paper concerns the relative expressiveness of monadic second-order and first-order metric logics over the reals. This direction is motivated by the celebrated result of Kamp [21] that Linear Temporal Logic (LTL) has the same expressiveness over the structure $(\mathbb{N}, <)$ as monadic first-order logic ($\mathsf{FO}(<)$). An influential consequence of Kamp's result is that LTL has emerged as the canonical temporal logic over the naturals. While a version of Kamp's result holds over the structure $(\mathbb{R}_{\geq 0}, <)$, the correspondence between predicate logics and temporal logics becomes considerably more complicated with the introduction of *metric* specifications. In practice this has led to a veritable babel of metric temporal logics over the reals [5].

A natural predicate logic in which to formalise metric specifications over the reals is the first-order monadic logic over the structure $(\mathbb{R}_{\geq 0}, <, +1)$. Given a set of uninterpreted monadic predicates $\mathbf{P}$, a model over $(\mathbb{R}_{\geq 0}, <, +1)$ is nothing but a function $f : \mathbb{R}_{\geq 0} \to 2^{\mathbf{P}}$ mapping each $x \in \mathbb{R}_{\geq 0}$ to the set of predicates that hold at x. Such a model is called a *flow* or *signal*, and naturally corresponds to the trajectory of a real-time system.

On the side of temporal logics, an appealing extension of LTL, called Metric Temporal Logic (MTL), was proposed by Koymans [24] almost twenty years ago. While MTL has been widely studied, it is well-known that there are first-order formulas over $(\mathbb{R}_{\geq 0}, <, +1)$ that cannot be expressed in MTL [20].

Our second main result is that, over bounded time domains, MTL has the same expressiveness as monadic first-order logic. Specifically, we show that MTL is as expressive as first-order logic over the structure $([0, N), <, +1)$, for any fixed bound N. Thus, as with language inclusion for timed automata, the restriction to time-boundedness leads to a better-behaved theory.

Finally, we relate automata and logics by showing decidability of the model-checking problem for timed automata against specifications expressed in MTL, first-order, and second-order monadic logics over $([0, N), <, +1)$. We also show decidability of the satisfiability problems for these logics. In contrast to the case of language inclusion between timed automata, the model-checking and satisfiability problems for monadic predicate logics all have non-elementary complexity, whereas these problems are EXPSPACE-complete in the case of MTL.

2 Timed Automata

Let X be a finite set of clocks, denoted x, y, z, etc. We define the set Φ_X of clock constraints over X via the following grammar, where $k \in \mathbb{N}$ stands for any non-negative integer, and $\bowtie \in \{=, \neq, <, >, \leq, \geq\}$ is a comparison operator:

$$\phi ::= \mathbf{true} \;\mid\; x \bowtie k \;\mid\; x - y \bowtie k \;\mid\; \phi_1 \wedge \phi_2 \;\mid\; \phi_1 \vee \phi_2 \,.$$

A ***timed automaton*** $\mathcal{A}$ is a six-tuple $(\Sigma, S, S_0, S_F, X, \Delta)$, where

- Σ is a finite set (alphabet) of events,
- S is a finite set of states,
- $S_0 \subseteq S$ is a set of initial states,
- $S_F \subseteq S$ is a set of accepting states,
- X is a finite set of clocks, and
- $\Delta \subseteq S \times S \times \Sigma \times \Phi_X \times 2^X$ is a finite set of transitions. A transition (s, s', a, ϕ, R) allows a jump from state s to s', consuming event $a \in \Sigma$ in the process, provided the constraint ϕ on clocks is met. Afterwards, the clocks in R are reset to zero, while all other clocks remain unchanged.

Given a timed automaton $\mathcal{A}$ as above, a *clock valuation* is a function $\nu : X \to \mathbb{R}_{\geq 0}$. If $t \in \mathbb{R}_{\geq 0}$, we let $\nu+t$ be the clock valuation such that $(\nu+t)(x) = \nu(x)+t$ for all $x \in X$.

A *configuration* of $\mathcal{A}$ is a pair (s, ν), where $s \in S$ is a state and ν is a clock valuation.

An *accepting run* of $\mathcal{A}$ is a finite alternating sequence of configurations and delayed transitions $\pi = (s_0, \nu_0) \xrightarrow{d_1,\theta_1} (s_1, \nu_1) \xrightarrow{d_2,\theta_2} \ldots \xrightarrow{d_n,\theta_n} (s_n, \nu_n)$, where each $d_i \in \mathbb{R}_{>0}$ and each $\theta_i = (s_{i-1}, s_i, a_i, \phi_i, R_i) \in \Delta$, subject to the following conditions:

1. $s_0 \in S_0$, and for all $x \in X$, $\nu_0(x) = 0$.
2. For all $0 \leq i \leq n-1$, $\nu_i + d_{i+1}$ satisfies ϕ_{i+1}.
3. For all $0 \leq i \leq n-1$, $\nu_{i+1}(x) = \nu_i(x) + d_{i+1}$ for all $x \in X \setminus R_{i+1}$, and $\nu_{i+1}(x) = 0$ for all $x \in R_{i+1}$.
4. $s_n \in S_F$.

Each d_i is interpreted as the (strictly positive[3]) time delay between the firing of transitions, and each configuration (s_i, ν_i), for $i \geq 1$, records the data immediately following transition θ_i. Abusing notation, we also write runs in the form $(s_0, \nu_0) \xrightarrow{d_1,a_1} (s_1, \nu_1) \xrightarrow{d_2,a_2} \ldots \xrightarrow{d_n,a_n} (s_n, \nu_n)$ to highlight the run's events.

A ***timed word*** is a pair (σ, τ), where $\sigma = \langle a_1 a_2 \ldots a_n \rangle \in \Sigma^*$ is a word and $\tau = \langle t_1 t_2 \ldots t_n \rangle \in (\mathbb{R}_{>0})^*$ is a strictly increasing sequence of real-valued timestamps of the same length.

Such a timed word is *accepted* by $\mathcal{A}$ if $\mathcal{A}$ has some accepting run of the form $\pi = (s_0, \nu_0) \xrightarrow{d_1,a_1} (s_1, \nu_1) \xrightarrow{d_2,a_2} \ldots \xrightarrow{d_n,a_n} (s_n, \nu_n)$ where, for each $1 \leq i \leq n$, $t_i = d_1 + d_2 + \ldots + d_i$.

In this paper, we are mainly concerned with behaviours over time domains of the form $[0, N)$, where $N \in \mathbb{N}$ is a positive integer.[4] Let us in general write $\mathbb{T}$ to denote either $[0, N)$ or $\mathbb{R}_{\geq 0}$. We then define $L_{\mathbb{T}}(\mathcal{A})$ to be the set of timed words accepted by $\mathcal{A}$ all of whose timestamps belong to $\mathbb{T}$.

Remark 1. Our timed automata have transitions that are labelled with instantaneous events; this is by far the most common model found in the literature. Alternatives include automata in which states are labelled with atomic propositions [3], or even mixed models in which states carry atomic propositions and transitions carry events. Other variants allow for silent transitions (invisible events), invariants on states, and combinations thereof. All the results presented in this paper carry over without difficulty to these more expressive models.

[3] This gives rise to the *strongly monotonic* semantics for timed automata; in contrast, the *weakly monotonic* semantics allows multiple events to happen ‘simultaneously’ (or, more precisely, with null-duration delays between them). The main results of this paper remain substantively the same under either semantics, although the weakly monotonic semantics causes some slight complications.

[4] All our results in fact carry over to the case in which N is chosen to be an arbitrary positive real number [29].

Note that we are focussing on *finite* words. Timed automata can be defined to accept infinite words (for example, by using Büchi acceptance conditions [2]), although over bounded time infinite words are automatically *Zeno* (and *ipso facto* ruled out from the accepted language by most researchers). Theorem 4 could nonetheless be extended to such infinite words, if desired [29].

3 Metric Logics

3.1 Syntax

Let **Var** be a set of *first-order variables*, denoted x, y, z, etc., ranging over non-negative real numbers. Let **MP** be a set of *monadic predicates*, denoted P, Q, R, etc. Monadic predicates will alternately be viewed as second-order variables over $\mathbb{R}_{\geq 0}$, i.e., ranging over sets of non-negative real numbers, and as atomic propositions holding at various points in time.

Second-order monadic formulas are obtained from the following grammar:

$$\varphi ::= \mathbf{true} \mid x < y \mid +1(x,y) \mid P(x) \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \forall x\, \varphi \mid \forall P\, \varphi\,,$$

where $+1$ is a binary relation symbol, with the intuitive interpretation of $+1(x,y)$ as '$x+1=y$'.[5] We refer to $\forall x$ and $\forall P$ as *first-order* and *second-order* quantifiers respectively. Existential quantifiers $\exists x$ and $\exists P$ are definable via standard dualities.

The ***monadic second-order metric logic of order***, written $\mathsf{MSO}(<,+1)$, comprises all second-order monadic formulas. Its first-order fragment, the ***(monadic) first-order metric logic of order***, written $\mathsf{FO}(<,+1)$, comprises all $\mathsf{MSO}(<,+1)$ formulas that do not contain any second-order quantifier; note that these formulas are however allowed free monadic predicates.

We also define two further purely order-theoretic sublogics, which are peripheral to our main concerns but necessary to express some key related results. The *monadic second-order logic of order*, $\mathsf{MSO}(<)$, comprises all second-order monadic formulas that do not make use of the $+1$ relation. Likewise, the *(monadic) first-order logic of order*, $\mathsf{FO}(<)$, comprises those $\mathsf{MSO}(<)$ formulas that do not figure second-order quantification.

Metric Temporal Logic, abbreviated MTL, comprises the following *temporal formulas*:

$$\theta ::= \mathbf{true} \mid P \mid \theta_1 \wedge \theta_2 \mid \theta_1 \vee \theta_2 \mid \neg\theta \mid \Diamond_I\theta \mid \Box_I\theta \mid \theta_1\, \mathcal{U}_I\, \theta_2\,,$$

where $P \in \mathbf{MP}$ is a monadic predicate, and $I \subseteq \mathbb{R}_{\geq 0}$ is an open, closed, or half-open interval with endpoints in $\mathbb{N} \cup \{\infty\}$. If $I = [0,\infty)$, then we omit the annotation I in the corresponding temporal operator. We also use pseudo-arithmetic

[5] The usual approach is of course to define $+1$ as a unary function symbol; this however necessitates an awkward treatment over bounded domains, as considered in this paper. We shall nonetheless abuse notation later on and invoke $+1$ as if it were a function, in the interest of clarity.

expressions to denote intervals. For example, the expression '≥ 1' denotes $[1, \infty)$ and '$=1$' denotes the singleton $\{1\}$.

Note that our version of MTL includes only *forwards* temporal operators, in keeping with the most common definition found in the literature. All our results extend straightforwardly to variants of MTL that make use of both forwards and backwards operators. It is also worth pointing out that the $\Diamond_I$ and $\Box_I$ operators are derivable from $\mathcal{U}_I$.

Finally, *Linear Temporal Logic*, written LTL, consists of those MTL formulas in which every indexing interval I on temporal operators is $[0, \infty)$ (and hence omitted).

Figure 3.1 pictorially summarises the syntactic inclusions and relative expressive powers of these various logics.

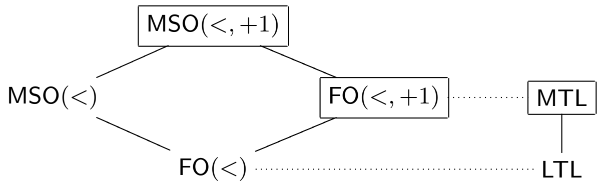


Fig. 1. Relative expressiveness among the various logics. Metric logics are enclosed in boxes. Straight lines denote syntactical inclusion, whereas dotted lines indicate expressive equivalence over bounded time domains (cf. Section 5).

3.2 Semantics

Let $\mathbf{P} \subseteq \mathbf{MP}$ be a finite set of monadic predicates, and let us again write $\mathbb{T}$ to denote either $[0, N)$ (for some fixed $N \in \mathbb{N}$) or $\mathbb{R}_{\geq 0}$. A ***flow*** (or *signal*) over $\mathbf{P}$ is a function $f : \mathbb{T} \to 2^{\mathbf{P}}$ that is *finitely variable.* Finite variability means that the restriction of f to any finite subinterval of $\mathbb{T}$ has only finitely many discontinuities.[6] Note that, unlike [44], we do not place any bound on the variability, other than requiring that it be finite.

A flow $f : \mathbb{T} \to 2^{\mathbf{P}}$ corresponds to an interpretation of the monadic predicates in $\mathbf{P}$: for any $P \in \mathbf{P}$, the interpretation of P as a subset of $\mathbb{T}$ is simply $\{t \in \mathbb{T} \mid P \in f(t)\}$. Conversely, any (finitely-variable) interpretation of all the predicates in $\mathbf{P}$ defines a unique flow $f : \mathbb{T} \to 2^{\mathbf{P}}$.

A timed word $(\langle a_1 \dots a_n \rangle, \langle t_1 \dots t_n \rangle)$ over alphabet Σ can be viewed as a (particular type of) flow, as follows. Let $\mathbb{T}$ be either $[0, N)$ (for some $N > t_n$) or

[6] The restriction to finitely-variable flows can be partially lifted, as we discuss in the full version of this paper [29]. Note however that from a computer-science perspective, infinitely-variable flows do not correspond to feasible computations, hence the widespread adoption of the present restriction in the literature.

$\mathbb{R}_{\geq 0}$, and let $\mathbf{P} = \Sigma$. Set $f(t_i) = \{a_i\}$, for $1 \leq i \leq n$, and $f(t) = \emptyset$ for all other values of $t \in \mathbb{T}$.

Fix a time domain $\mathbb{T}$, equipped with the standard order relation $<$ and the obvious binary relation $+1$, i.e., $+1(a, b)$ iff $a, b \in \mathbb{T}$ and $a+1 = b$. Given a formula φ of $\mathsf{MSO}(<, +1)$ or one of its sublogics, let $\mathbf{P}$ and $\{x_1, \ldots, x_n\}$ respectively be the sets of free monadic predicates and free first-order variables appearing in φ. For any flow $f : \mathbb{T} \to 2^{\mathbf{P}}$ and real numbers $a_1, \ldots, a_n \in \mathbb{T}$, the satisfaction relation $(f, a_1, \ldots, a_n) \models \varphi$ is defined inductively on the structure of φ in the standard way. For example:

- $(f, a) \models P(x)$ iff $P \in f(a)$.
- $(f, a_1, \ldots, a_n) \models \forall P\, \varphi$ iff for all finitely-variable flows $g : \mathbb{T} \to 2^{\mathbf{P} \cup \{P\}}$ extending f (i.e., such that $g{\restriction}_{\mathbf{P}} = f$), we have $(g, a_1, \ldots, a_n) \models \varphi$.
  (Here $\mathbf{P}$ is the set of free monadic predicates appearing in $\forall P\, \varphi$, and therefore does not contain P.)

And so on.

We shall particularly be interested in the special case in which φ is a *sentence*, i.e., a formula with no free first-order variable. In such instances, we simply write the satisfaction relation as $f \models \varphi$.

For θ an MTL or LTL formula, let $\mathbf{P}$ be the set of monadic predicates appearing in θ. Given a flow $f : \mathbb{T} \to 2^{\mathbf{P}}$ and $t \in \mathbb{T}$, the satisfaction relation $(f, t) \models \theta$ is defined inductively on the structure of θ, as follows:

- $(f, t) \models \mathbf{true}$.
- $(f, t) \models P$ iff $P \in f(t)$.
- $(f, t) \models \theta_1 \land \theta_2$ iff $(f, t) \models \theta_1$ and $(f, t) \models \theta_2$.
- $(f, t) \models \theta_1 \lor \theta_2$ iff $(f, t) \models \theta_1$ or $(f, t) \models \theta_2$.
- $(f, t) \models \neg\theta$ iff $(f, t) \not\models \theta$.
- $(f, t) \models \Diamond_I \theta$ iff there exists $u \in \mathbb{T}$ with $u > t$, $u - t \in I$, and $(f, u) \models \theta$.
- $(f, t) \models \Box_I \theta$ iff for all $u \in \mathbb{T}$ with $u > t$ and $u - t \in I$, $(f, u) \models \theta$.
- $(f, t) \models \theta_1\, \mathcal{U}_I\, \theta_2$ iff there exists $u \in \mathbb{T}$ with $u > t$, $u - t \in I$, $(f, u) \models \theta_2$, and for all $v \in (t, u)$, $(f, v) \models \theta_1$.

Finally, we write $f \models \theta$ iff $(f, 0) \models \theta$. This is sometimes referred to as the *initial semantics*.

Note that we have adopted a *strict* semantics, in which the present time t has no influence on the truth values of future temporal subformulas. Strictness is required for Theorem 2, but our other results hold under both the strict and non-strict semantics.

An important point concerning our semantics is that it is *continuous*, rather than *pointwise*: more precisely, the temporal operators quantify over all time points of the domain, as opposed to merely those time points at which a discontinuity occurs. Positive decidability results for satisfiability and model checking of MTL over unbounded time intervals have been obtained in the pointwise semantics [33, 34, 35]; it is worth noting that none of these results hold in the continuous semantics.

4 Satisfiability

The canonical time domain for interpreting the metric logics $\mathsf{MSO}(<,+1)$, $\mathsf{FO}(<,+1)$, and MTL is the non-negative real line $\mathbb{R}_{\geq 0}$. Unfortunately, none of these logics are decidable over $\mathbb{R}_{\geq 0}$ [5, 6, 19].

Our main focus in this paper is therefore on satisfiability over bounded time domains of the form $[0, N)$, for $N \in \mathbb{N}$. For each of the logics introduced in Section 3.1, one can consider the corresponding ***time-bounded satisfiability problem***: given a sentence φ over a set $\mathbf{P}$ of free monadic predicates, together with a time bound $N \in \mathbb{N}$, does there exist a flow $f : [0, N) \to 2^{\mathbf{P}}$ such that $f \models \varphi$?

One of our main results is the following:

Theorem 1. *The time-bounded satisfiability problems for the metric logics* $\mathsf{MSO}(<,+1)$*,* $\mathsf{FO}(<,+1)$*, and* MTL *are all decidable, with the following complexities:*[7]

$\mathsf{MSO}(<,+1)$	Non-elementary
$\mathsf{FO}(<,+1)$	Non-elementary
MTL	EXPSPACE-complete

Remark 2. It is worth noting that if one allows second-order quantification over predicates of *arbitrary* variability, then $\mathsf{MSO}(<)$ is undecidable over $\mathbb{R}_{\geq 0}$ [40]. Since any non-trivial interval of the form $[0, N)$ is order-isomorphic to $\mathbb{R}_{\geq 0}$, the undecidability of $\mathsf{MSO}(<)$ carries over to bounded time domains, and *a fortiori* to $\mathsf{MSO}(<,+1)$ over bounded time domains. $\mathsf{FO}(<,+1)$ and MTL however remain decidable over bounded time domains regardless of the variability of flows—see [29].

The proofs of all theorems in this paper are given in Appendix A.

5 Expressiveness

Fix a time domain $\mathbb{T}$ to be either $[0, N)$ (for some $N \in \mathbb{N}$) or $\mathbb{R}_{\geq 0}$. Let $\mathcal{L}$ and $\mathcal{J}$ be two logics. We say that $\mathcal{L}$ is ***at least as expressive as*** $\mathcal{J}$ if, for any sentence θ of $\mathcal{J}$, there exists a sentence φ of $\mathcal{L}$ such that θ and φ are satisfied by precisely the same set of flows over $\mathbb{T}$.

Two logics are then said to be ***equally expressive*** if each is at least as expressive as the other.

The following result can be viewed as an extension of Kamp's celebrated theorem [21, 12] to metric logics over bounded time domains:

Theorem 2. *For any fixed bounded time domain of the form* $[0, N)$*, with* $N \in \mathbb{N}$*, the metric logics* $\mathsf{FO}(<,+1)$ *and* MTL *are equally expressive. Moreover, this equivalence is effective.*

[7] All the complexity results in this paper assume that the time bound N is provided in binary.

Remark 3. Note that expressiveness here is relative to a *single* structure $\mathbb{T}$, rather than to a *class* of structures. In particular, although $\mathsf{FO}(<,+1)$ and MTL are equally expressive over any bounded time domain of the form $[0, N)$, the correspondence and witnessing formulas may very well vary according to the time domain.

It is interesting to note that $\mathsf{FO}(<,+1)$ is strictly more expressive than MTL over $\mathbb{R}_{\geq 0}$ [20]. For example, MTL is incapable of expressing the following formula (in slightly abusive but readable notation)

$$\exists x\, \exists y\, \exists z\, (x < y < z < x + 1 \wedge P(x) \wedge P(y) \wedge P(z))$$

over the non-negative reals. This formula asserts that, sometime in the future, P will hold at three distinct time points within a single time unit.

It is also worth noting that $\mathsf{MSO}(<,+1)$ is strictly more expressive than $\mathsf{FO}(<,+1)$—and hence MTL—over any time domain; see [29].

Finally, we point out that, in contrast to Kamp's theorem [21], but similarly to [12], Theorem 2 does not require backwards temporal operators for MTL (although adding these would be harmless). This may appear surprising in view of the main results of [18], and indeed backwards temporal operators for MTL *would* be required had we allowed flows to be infinitely variable [29].

6 Model Checking and Language Inclusion

We now turn to questions concerning the time-bounded behaviours of timed automata. Recall from Section 3.2 that timed words over an alphabet Σ can be viewed as flows from a sufficiently large time domain over the set of monadic predicates $\mathbf{P} = \Sigma$. The ***model-checking problem*** takes as inputs a timed automaton $\mathcal{A}$ with alphabet Σ, a sentence φ with set of free monadic predicates $\mathbf{P} = \Sigma$, and a time domain $\mathbb{T}$ (taken to be either $[0, N)$, for some $N \in \mathbb{N}$, or $\mathbb{R}_{\geq 0}$). The question is then whether every timed word (flow) in $L_{\mathbb{T}}(\mathcal{A})$ satisfies φ.

Unfortunately, the model-checking problem for timed automata and any of the metric logics introduced in this paper is undecidable over the non-negative real line $\mathbb{R}_{\geq 0}$ [5, 6]; this follows easily from the undecidability of the satisfiability problem for these logics over flows. We therefore focus on the ***time-bounded model-checking problem***, in which the time domain is required to be bounded. We have:

Theorem 3. *The time-bounded model-checking problems for timed automata against the metric logics* $\mathsf{MSO}(<,+1)$, $\mathsf{FO}(<,+1)$, *and* MTL *are all decidable, with the same complexities as the corresponding time-bounded satisfiability problems (cf. Theorem 1): non-elementary for* $\mathsf{MSO}(<,+1)$ *and* $\mathsf{FO}(<,+1)$, *and EXPSPACE-complete for* MTL.

The ***language inclusion problem*** takes as inputs two timed automata, $\mathcal{A}$ and $\mathcal{B}$, sharing a common alphabet, together with a time domain $\mathbb{T}$ (of the form

$[0,N)$, for some $N \in \mathbb{N}$, or $\mathbb{R}_{\geq 0}$). The question is then whether every timed word accepted by $\mathcal{A}$ over $\mathbb{T}$ is also accepted by $\mathcal{B}$.

As for model checking, language inclusion is unfortunately undecidable over $\mathbb{R}_{\geq 0}$ [2]. The ***time-bounded language inclusion problem*** circumvents this by restricting to bounded time domains. This leads us to our final main result, as follows:

Theorem 4. *The time-bounded language inclusion problem for timed automata is decidable and 2EXPSPACE-complete.*

A Proofs of Theorems

We begin by stating some relatively straightforward facts translating results about order-theoretic (i.e., non-metric) logics over the reals to bounded domains. For the purposes of Lemmas 1 and 2, let $N \in \mathbb{N}$ be a fixed positive integer.

Lemma 1. *Let φ be an $\mathsf{FO}(<)$ formula and ψ be an LTL formula that are satisfied by the same flows over $\mathbb{R}_{\geq 0}$. Then φ and ψ are satisfied by the same flows over $[0,N)$.*

Proof. Let $\mathbf{P}$ be the set of monadic predicates appearing in φ and ψ. We need to show that, for any flow $f : [0,N) \to 2^{\mathbf{P}}$, $f \models \varphi$ iff $f \models \psi$.

Let $\beta : [0,N) \to \mathbb{R}_{\geq 0}$ be an order isomorphism (monotone bijection), and let $f : [0,N) \to 2^{\mathbf{P}}$ be a finitely-variable flow. Observe that $f \circ \beta : \mathbb{R}_{\geq 0} \to 2^{\mathbf{P}}$ is then also finitely variable. We have

$$f \models \varphi \ \text{ iff } \ f \circ \beta \models \varphi \ \text{ iff } \ f \circ \beta \models \psi \ \text{ iff } \ f \models \psi\,,$$

as required. The first and third equivalences follow easily by induction on φ and ψ respectively, whereas the second equivalence holds by assumption. $\square$

Some of our results require the use of *Linear Temporal Logic with Past Operators*, $\mathsf{LTL{+}Past}$. This logic simply augments LTL with the backwards operators $\overleftarrow{\Diamond}$ ('sometimes in the past'), $\overleftarrow{\Box}$ ('always in the past'), and $\mathcal{S}$ ('since'); see [25].

Lemma 2. *The following all hold over the time domain $[0,N)$:*

1. $\mathsf{FO}(<)$ *and* LTL *are equally expressive.*
2. *Satisfiability for* $\mathsf{LTL{+}Past}$ *is decidable in PSPACE.*
3. *Satisfiability for* $\mathsf{MSO}(<)$ *is decidable.*
4. *Satisfiability for* $\mathsf{FO}(<)$ *is non-elementary.*

Proof.

1. It is known from [18] that $\mathsf{FO}(<)$ and LTL are equally expressive over $\mathbb{R}_{\geq 0}$ for finitely-variable flows.[8] It immediately follows from Lemma 1 that these logics are also equally expressive over $[0,N)$ (for finitely-variable flows).

[8] Note that the assumption of finite variability is crucial here: the same paper, [18], shows that $\mathsf{FO}(<)$ is strictly more expressive than LTL when flows of arbitrary variability are allowed.

2. Note that while $[0, N)$ and $\mathbb{R}_{\geq 0}$ are order-isomorphic, finitely-variable flows over $\mathbb{R}_{\geq 0}$ do not necessarily translate to finitely-variable flows over $[0, N)$. Rather, it is easily seen that the finitely-variable flows over $\mathbb{R}_{\geq 0}$ whose counterparts in $[0, N)$ are finitely variable are precisely those flows that are ultimately constant: every monadic predicate P is such that there is a time point beyond which P is either always true or always false.

The property of an $\mathbb{R}_{\geq 0}$-flow being ultimately constant is easily captured by an LTL formula such as $\bigwedge\{\Diamond(\Box P \vee \Box\neg P) \mid P \in \mathbf{P}\}$, where $\mathbf{P}$ is the relevant set of monadic predicates. Let us denote this formula by κ.

Given an LTL+Past formula φ, we therefore have that φ is satisfiable over $[0, N)$ iff φ is satisfiable over $\mathbb{R}_{\geq 0}$ by ultimately-constant flows iff $\varphi \wedge \kappa$ is satisfiable over $\mathbb{R}_{\geq 0}$. Since satisfiability of LTL+Past over $\mathbb{R}_{\geq 0}$ is in PSPACE [38], the desired result follows.

3. The decidability of MSO$(<)$ satisfiability over $[0, N)$ follows similarly to the case of LTL+Past. A sentence φ of MSO$(<)$ is satisfiable over $[0, N)$ iff the sentence $\widehat{\varphi}$ is satisfiable over $\mathbb{R}_{\geq 0}$, where $\widehat{\varphi}$ is obtained from φ by requiring all free monadic predicates to be ultimately constant and also syntactically restricting second-order quantification to ultimately-constant monadic predicates. The result follows from the decidability of satisfiability for MSO$(<)$ over $\mathbb{R}_{\geq 0}$ by finitely-variable flows [36].

4. Satisfiability of FO$(<)$ over $\mathbb{R}_{\geq 0}$ is known to be non-elementary [42, 27]. An examination of the proof shows that this result in fact also holds over (finitely-variable) flows that are ultimately constant. It immediately follows that satisfiability for FO$(<)$ over $[0, N)$ is non-elementary. $\square$

Theorem 1. *The time-bounded satisfiability problems for* MSO$(<, +1)$ *and* FO$(<, +1)$ *are decidable and non-elementary, whereas the time-bounded satisfiability problem for* MTL *is EXPSPACE-complete.*

Proof. Throughout this proof, let $N \in \mathbb{N}$ be fixed.

Decidability. An easy first observation is that any MTL formula can be translated into an equivalent FO$(<, +1)$ formula over $[0, N)$, following an approach similar to the translation of LTL formulas into FO$(<)$. For decidability, it therefore suffices to handle the case of MSO$(<, +1)$.

Let $\mathbf{P} \subseteq \mathbf{MP}$ be a finite set of monadic predicates. With each $P \in \mathbf{P}$, we associate a collection $P_0, \ldots, P_{N-1}$ of N fresh monadic predicates. We then let $\overline{\mathbf{P}} = \{P_i \mid P \in \mathbf{P}, 0 \leq i \leq N-1\}$.

Intuitively, each monadic predicate P_i represents P over the subinterval $[i, i+1)$. Indeed, there is an obvious 'stacking' bijection (indicated by overlining) between the set of flows $\{f : [0, N) \to 2^{\mathbf{P}}\}$ and the set of flows $\{\overline{f} : [0, 1) \to 2^{\overline{\mathbf{P}}}\}$.

Let φ be an MSO$(<, +1)$ sentence with set of free monadic predicates $\mathbf{P}$. We will define an MSO$(<)$ sentence $\overline{\varphi}$ such that, for any flow $f : [0, N) \to 2^{\mathbf{P}}$, $f \models \varphi$ iff $\overline{f} \models \overline{\varphi}$.

We can assume that φ does not contain any (first- or second-order) existential quantifiers, by replacing the latter with combinations of universal quantifiers and negations if need be. It is also convenient to rewrite φ into a formula that makes use of the constant $N-1$ as well as a family of unary functions $+k$ (for $k \in \mathbb{N}$) instead of the $+1$ relation. To this end, replace every occurrence of $+1(x, y)$ in φ by $(x < N-1 \wedge x+1 = y)$.

Next, recursively replace every instance of $\forall x\, \psi$ in φ by the formula

$$\forall x\, (\psi[x/x] \wedge \psi[x+1/x] \wedge \ldots \wedge \psi[x+(N-1)/x])\,,$$

where $\psi[t/x]$ denotes the formula resulting from substituting every free occurrence of the variable x in ψ by the term t. Intuitively, this transformation is legitimate since first-order variables in our target formula will range over $[0, 1)$ rather than $[0, N)$.

Having carried out these substitutions, rewrite every term in φ involving a variable into the form $x + k$, where x is the term's variable and $k \in \mathbb{N}$ is a non-negative integer constant.

Every inequality occurring in φ is now of the form $x+k < N-1$ or $x+k_1 < y+k_2$. Replace every inequality of the first kind by **true** if $k+2 \leq N$ and by $\neg$**true** otherwise, and replace every inequality of the second kind by (i) $x < y$, if $k_1 = k_2$; (ii) **true**, if $k_1 < k_2$; and (iii) $\neg$**true** otherwise.

Every equality occurring in φ is now of the form $x + k_1 = y + k_2$. Replace every such equality by $\neg(x < y \vee y < x)$—which of course is the same as $x = y$—if $k_1 = k_2$, and by $\neg$**true** otherwise.

Every occurrence of a monadic predicate in φ now has the form $P(x+k)$, for $k \leq N-1$. Replace every such predicate by $P_k(x)$.

Finally, recursively replace every occurrence of $\forall P\, \psi$ in φ by $\forall P_0\, \forall P_1 \ldots \forall P_{N-1}\, \psi$. The resulting formula is the desired $\overline{\varphi}$.

It is now straightforward to prove by induction that the set of $[0, N)$-flows satisfying the original φ are indeed in one-to-one correspondence with the set of $[0, 1)$-flows satisfying $\overline{\varphi}$.

Note that $\overline{\varphi}$ does not use any $+1$ or $+k$ functions or relations, and is therefore a purely order-theoretic sentence of $\mathsf{MSO}(<)$. Decidability therefore follows from Lemma 2.3.

Complexity. Since time-bounded satisfiability for $\mathsf{FO}(<)$ is non-elementary (Lemma 2.4), then *a fortiori* so are time-bounded satisfiability for $\mathsf{FO}(<, +1)$ and $\mathsf{MSO}(<, +1)$.

In [9], it is shown that the satisfiability problem for *Bounded Metric Temporal Logic* ($\mathsf{Bounded\text{-}MTL}$, which consists of MTL formulas in which all intervals I appearing as subscripts to temporal operators are bounded) is EXPSPACE-hard. An examination of that proof shows that it readily carries over to the time-bounded satisfiability problem for MTL, which is therefore EXPSPACE-hard as well.

Let θ be an MTL formula. We will sketch how to manufacture an exponential-size formula $\widetilde{\theta}$ of $\mathsf{LTL}{+}\mathsf{Past}$ such that θ is satisfiable over $[0, N)$ iff $\widetilde{\theta}$ is satisfiable over $[0,1)$. Since LTL+Past satisfiability is in PSPACE (Lemma 2.2), satisfiability for θ over $[0,N)$ can be decided in exponential space. Together with the previous paragraph, this will therefore establish EXPSPACE-completeness of time-bounded satisfiability for MTL.

The process for constructing $\widetilde{\theta}$ is similar in spirit to the reduction above of MSO$(<,+1)$ over $[0,N)$ to MSO$(<)$ over $[0,1)$. Once again, we imagine the interval $[0,N)$ decomposed into N vertically-stacked unit-length intervals.

Let $\mathrm{Sub}(\theta)$ denote the set of subformulas of θ.[9] For every subformula $\rho \in \mathrm{Sub}(\theta)$ and every $i \in \{0,\ldots,N{-}1\}$, we postulate an atomic proposition (monadic predicate) F_i^ρ of $\widetilde{\theta}$. Intuitively, $\widetilde{\theta}$ will be such that it forces F_i^ρ to hold at time $t \in [0,1)$ precisely when ρ holds at time $t+i \in [0,N)$. This is achieved by constructing, for each atomic proposition F_i^ρ, an LTL+Past formula ω_i^ρ that specifies F_i^ρ's value, at any time t, in relation to the behaviours—before time t, at time t, and after time t—of the atomic propositions associated with the subformulas of ρ.

The various clauses ω_i^ρ are produced according to the outermost connective of ρ, following an approach very similar to the reduction in [9] of Flat-MTL to LTL+Past. We give three examples below that should convey the general underlying idea:

- For P a monadic predicate, ω_i^P is the formula **true**, since P—being atomic—is not constrained by any subformula.
- $\omega_i^{\rho\wedge\eta}$ is the formula $F_i^{\rho\wedge\eta} \leftrightarrow (F_i^\rho \wedge F_i^\eta)$.
- $\omega_i^{\rho\,\mathcal{U}_{(2,3)}\,\eta}$ will depend on whether $i \leq N-4$, $i = N-3$, or $i \geq N-2$. Assuming, for example, that $i \leq N-4$, $\omega_i^{\rho\,\mathcal{U}_{(2,3)}\,\eta}$ is the formula

$$F_i^{\rho\,\mathcal{U}_{(2,3)}\,\eta} \leftrightarrow \left(\begin{array}{l} \Box F_i^\rho \wedge \overleftarrow{\Box} F_{i+1}^\rho \wedge F_{i+1}^\rho \wedge \Box F_{i+1}^\rho \wedge \overleftarrow{\Box} F_{i+2}^\rho \wedge F_{i+2}^\rho \wedge \\ \left(F_{i+2}^\rho \,\mathcal{U}\, F_{i+2}^\eta \vee \left(\Box F_{i+2}^\rho \wedge \overleftarrow{\Diamond}\left(F_{i+3}^\eta \wedge \overleftarrow{\Box} F_{i+3}^\rho\right)\right)\right) \end{array} \right).$$

Let us now write Ω to denote the conjunction

$$\bigwedge \{\omega_i^\rho \wedge \Box\omega_i^\rho \mid \rho \in \mathrm{Sub}(\theta), 0 \leq i \leq N-1\}\,.$$

Observe that Ω fully determines the behaviours of all the F_i^ρ's according to the behaviours of the F_i^P's (for P a monadic predicate of θ), throughout the time domain $[0,1)$.

The desired formula $\widetilde{\theta}$ is then $F_0^\theta \wedge \Omega$.

To establish the correctness of this construction, consider the set $\mathbf{P}$ of monadic predicates appearing in θ, and let $\overline{\mathbf{P}} = \{F_i^P \mid P \in \mathbf{P}, 0 \leq i \leq N-1\}$ be the set of atomic propositions associated with the monadic predicates of θ. There is once again a stacking bijection between the set of flows $\{f : [0,N) \to 2^{\mathbf{P}}\}$ and the set of flows $\{\overline{f} : [0,1) \to 2^{\overline{\mathbf{P}}}\}$. Writing $\mathbf{F} = \{F_i^\rho \mid \rho \in \mathrm{Sub}(\theta), 0 \leq i \leq N-1\}$ to denote the full set of monadic predicates appearing in $\widetilde{\theta}$, it is clear that any

[9] The subformulas of $P \vee (Q\,\mathcal{U}_{[1,3)}\,R)$, for example, are $P \vee (Q\,\mathcal{U}_{[1,3)}\,R)$, P, $Q\,\mathcal{U}_{[1,3)}\,R$, Q, R.

flow $\overline{f} : [0,1) \to 2^{\overline{\mathbf{P}}}$ can be uniquely extended into a flow $\widetilde{f} : [0,1) \to 2^{\mathbf{F}}$, by requiring that $\widetilde{f} \models \Omega$.

It is now straightforward to prove by induction on $\rho \in \mathrm{Sub}(\theta)$ that, for any flow $f : [0,N) \to 2^{\mathbf{P}}$, any $t \in [0,1)$, and any $i \in \{0,\ldots,N-1\}$, we have $(f, t+i) \models \rho$ iff $(\widetilde{f}, t) \models F_i^{\rho}$. The desired result follows by setting $t = 0$, $i = 0$, and $\rho = \theta$. □

Theorem 2. *For any fixed bounded time domain of the form* $[0,N)$*, with* $N \in \mathbb{N}$*, the metric logics* $\mathsf{FO}(<,+1)$ *and* MTL *are equally expressive. Moreover, this equivalence is effective.*

Proof. Throughout this proof, let $N \in \mathbb{N}$ be fixed.

The reduction from MTL to $\mathsf{FO}(<,+1)$ is straightforward, and therefore omitted.

For the other direction, let φ be an $\mathsf{FO}(<,+1)$ sentence with set of free monadic predicates $\mathbf{P} \subseteq \mathbf{MP}$. As in the proof of Theorem 1, let $\overline{\mathbf{P}} = \{P_i \mid P \in \mathbf{P}, 0 \leq i \leq N-1\}$ be a set of fresh monadic predicates. The stacking construction used in Theorem 1 yields an $\mathsf{FO}(<)$ sentence $\overline{\varphi}$ with set of free monadic predicates $\overline{\mathbf{P}}$, such that there is a bijection (indicated by overlining) from the set of $[0,N)$-flows over $\mathbf{P}$ satisfying φ to the set of $[0,1)$-flows over $\overline{\mathbf{P}}$ satisfying $\overline{\varphi}$.

According to [18], one can now construct an LTL formula ψ, with set of monadic predicates $\overline{\mathbf{P}}$, that defines precisely the same set of finitely-variable $\mathbb{R}_{\geq 0}$-flows as $\overline{\varphi}$. By Lemma 1, $\overline{\varphi}$ and ψ therefore also define precisely the same set of finitely-variable $[0,1)$-flows over $\overline{\mathbf{P}}$.

It therefore suffices to exhibit an MTL formula θ, over set of monadic predicates $\mathbf{P}$, such that, for any flow $f : [0,N) \to 2^{\mathbf{P}}$, $f \models \theta$ iff $\overline{f} \models \psi$.

To this end, write ι to denote the MTL formula $\Diamond_{=(N-1)}\mathbf{true}$. Note that, when interpreted within the time domain $[0,N)$, ι holds precisely over the time interval $[0,1)$. Perform the following substitutions on ψ to obtain the desired θ: (i) for each $P \in \mathbf{P}$, replace every occurrence of P_0 in ψ by P, and every occurrence of P_i in ψ (for $i \geq 1$) by $\Diamond_{=i}P$; (ii) recursively replace every occurrence of $\Diamond\gamma$ in ψ by $\Diamond(\iota \wedge \gamma)$; (iii) recursively replace every occurrence of $\Box\gamma$ in ψ by $\Box(\iota \to \gamma)$; (iv) recursively replace every occurrence of $\gamma_1 \,\mathcal{U}\, \gamma_2$ in ψ by $\gamma_1 \,\mathcal{U}\, (\iota \wedge \gamma_2)$.

Finally, show by induction on ψ that, for any flow $f : [0,N) \to 2^{\mathbf{P}}$ and any $t \in [0,1)$, one has $(f,t) \models \theta$ iff $(\overline{f}, t) \models \psi$. The desired result follows by setting $t = 0$. □

Theorem 3. *The time-bounded model-checking problems for timed automata against the metric logics* $\mathsf{MSO}(<,+1)$*,* $\mathsf{FO}(<,+1)$*, and* MTL *are all decidable, with the same complexities as the corresponding time-bounded satisfiability problems: non-elementary for* $\mathsf{MSO}(<,+1)$ *and* $\mathsf{FO}(<,+1)$*, and EXPSPACE-Complete for* MTL.

Proof. Fix $N \in \mathbb{N}$, and let $\mathcal{A}$ be a timed automaton over alphabet Σ. In [16], it is shown how to construct (in polynomial time) an MTL formula $\theta_{\mathcal{A}}$, over a potentially larger set of monadic predicates $\mathbf{P} \supseteq \Sigma$, such that, for any flow

$f : [0, N) \to 2^{\Sigma}$, $f \in L_{[0,N)}(\mathcal{A})$ iff there exists a flow $g : [0, N) \to 2^{\mathbf{P}}$ such that $g \models \theta_{\mathcal{A}}$ and $g\lceil_{\Sigma} = f$. Intuitively, the extra monadic predicates of $\theta_{\mathcal{A}}$ keep track of the (otherwise invisible) identity of transitions and clock resets that occur during runs of $\mathcal{A}$.

Of course, $\theta_{\mathcal{A}}$ can clearly instead be taken to be an $\mathsf{FO}(<, +1)$ or $\mathsf{MSO}(<, +1)$ formula, if desired. In all cases, given a metric formula φ, the model-checking problem for $\mathcal{A}$ and φ over $[0, N)$ boils down to whether $\theta_{\mathcal{A}} \wedge \neg\varphi$ is unsatisfiable over $[0, N)$ or not.

This shows that time-bounded model checking reduces to time-bounded satisfiability. For the converse, simply pick an automaton $\mathcal{A}$ that accepts every flow. □

Theorem 4. *The time-bounded language inclusion problem for timed automata is decidable and 2EXPSPACE-complete.*

Proof.

2EXPSPACE-membership. Fix $N \in \mathbb{N}$, and let $\mathcal{A}$ and $\mathcal{B}$ be timed automata over alphabet Σ. We give a procedure for deciding whether $L_{[0,N)}(\mathcal{A}) \subseteq L_{[0,N)}(\mathcal{B})$.

As in the proof of Theorem 3, let $\theta_{\mathcal{A}}$ be an MTL formula over set of monadic predicates $\mathbf{P} = \Sigma \cup \mathbf{U}$, with the property that each $[0, N)$-timed word over Σ accepted by $\mathcal{A}$ can be extended to a $[0, N)$-flow over $\mathbf{P}$ satisfying $\theta_{\mathcal{A}}$, and vice-versa. Likewise, let $\theta_{\mathcal{B}}$ be a similar MTL formula over set of monadic predicates $\mathbf{Q} = \Sigma \cup \mathbf{V}$ for the timed automaton $\mathcal{B}$. We assume that Σ, $\mathbf{U}$, and $\mathbf{V}$ are all pairwise disjoint.

Abusing notation, we see that $L_{[0,N)}(\mathcal{A}) \subseteq L_{[0,N)}(\mathcal{B})$ iff the following formula[10] holds over $[0, N)$:

$$\forall\Sigma\, \forall\mathbf{U}\, \exists\mathbf{V}\, (\neg\theta_{\mathcal{A}}(\Sigma, \mathbf{U}) \vee \theta_{\mathcal{B}}(\Sigma, \mathbf{V}))\,. \tag{1}$$

As in the proof of Theorem 1, we can transform the MTL formula $\neg\theta_{\mathcal{A}} \vee \theta_{\mathcal{B}}$ into an equisatisfiable but exponentially larger formula ψ of $\mathsf{LTL}{+}\mathsf{Past}$. More precisely, ψ has a different (and exponentially larger) set of monadic predicates $\mathbf{R} = \overline{\Sigma} \cup \overline{\mathbf{U}} \cup \overline{\mathbf{V}} \cup \mathbf{W}$, yet there is a one-to-one 'stacking' correspondence between the flows satisfying $\neg\theta_{\mathcal{A}} \vee \theta_{\mathcal{B}}$ and those satisfying $\exists\mathbf{W}\, \psi$. We adjust the outside quantifiers accordingly to transform Formula (1) into the equivalent formula

$$\forall\overline{\Sigma}\, \forall\overline{\mathbf{U}}\, \exists\overline{\mathbf{V}}\, \exists\mathbf{W}\, \psi(\overline{\Sigma}, \overline{\mathbf{U}}, \overline{\mathbf{V}}, \mathbf{W})\,.$$

Next, following [3] (as a special case), we transform ψ into an equivalent untimed finite-state automaton $\mathcal{C}$ whose transitions are labelled by subsets of $\overline{\Sigma} \cup \overline{\mathbf{U}} \cup \overline{\mathbf{V}} \cup \mathbf{W}$. This incurs a second exponential blowup.

Note that the existential quantifications $\exists\mathbf{W}$ and $\exists\overline{\mathbf{V}}$ simply correspond to relabelling all $\mathbf{W}$- and $\overline{\mathbf{V}}$-labelled transitions in $\mathcal{C}$; this can be carried out in

[10] Extensions of temporal logics by second-order quantification can be found in, e.g., [16, 41].

polynomial time. We are therefore asking whether the resulting automaton is universal over $\overline{\Sigma} \cup \overline{\mathbf{U}}$, i.e., accepts any string over this alphabet. Since universality is decidable in PSPACE, the overall procedure can be carried out in doubly-exponential space as claimed.

2EXPSPACE-hardness. We now establish 2EXPSPACE-hardness of the time-bounded language inclusion problem by reduction from the halting problem for 2EXPSPACE Turing machines. In fact we show 2EXPSPACE-hardness of the *time-bounded universality problem*: does a timed automaton accept every timed word over a given bounded time domain?

Suppose we are given a deterministic 2^{2^n}-space-bounded Turing machine $\mathcal{M}$, with a one-way infinite tape, together with input ξ of length n. We first describe a scheme to encode computations of $\mathcal{M}$ as untimed words. Let $\mathcal{M}$ have set of control states Q and tape alphabet Γ; then we consider untimed words over the alphabet $\Sigma \cup \{0, 1, \#\}$, where $\Sigma = (Q \times \Gamma) \cup \Gamma$. The idea is to encode a single tape cell as a word matching the regular expression $\Sigma(0+1)^{2^n}$. Here we think of the element of Σ as recording the contents of the tape cell[11] and the bit-string in $(0 + 1)^{2^n}$ as giving the address of the given tape cell in binary, measured from the leftmost tape cell. Now we encode a configuration of $\mathcal{M}$ as a string of 2^{2^n} tape cells followed by a single $\#$, and a computation of $\mathcal{M}$ as a string of configurations. For example, a configuration with tape contents $\gamma_0\gamma_1\gamma_2\gamma_3$, with control state q and with the read head pointing to γ_2, is represented by the word $w = \langle \gamma_0 00 \gamma_1 01 (q, \gamma_2) 10 \gamma_3 11 \# \rangle$.

Let us fix the time domain to be $[0, 2^n)$ for the remainder of this proof. We now invoke the above scheme to encode computations of $\mathcal{M}$ as timed words over $[0, 2^n)$. Define the timed language $L_{\mathcal{M},\xi}$ to consist of all those timed words $w = (\langle a_1 \ldots a_m \rangle, \langle t_1 \ldots t_m \rangle)$ over the alphabet $\Sigma \cup \{0, 1, \#\}$ such that:

(i) The untimed word $\langle a_1 \ldots a_k \rangle$ corresponding to the sequence of events in the time interval $[0, 1)$ in w encodes an accepting computation of $\mathcal{M}$ on input ξ according to the scheme presented above.
(ii) For all timestamps t_i in w, if $t_i < 2^n - 1$ then there exists a subsequent timestamp $t_j = t_i + 1$ such that $a_i = a_j$, and if $t_i \geq 1$ then there exists an earlier timestamp $t_j = t_i - 1$ such that $a_i = a_j$.

Condition (ii) essentially says that the sequence of timed events in the interval $[0, 1)$, which encodes the computation of $\mathcal{M}$, is copied in each subsequent unit-duration time interval.

It is clear that $L_{\mathcal{M},\xi} \neq \emptyset$ iff $\mathcal{M}$ accepts input ξ. Next we describe how to construct (in LOGSPACE) a timed automaton $\mathcal{A}_{\mathcal{M},\xi}$ whose accepted language is the complement of $L_{\mathcal{M},\xi}$. Thus $\mathcal{A}_{\mathcal{M},\xi}$ is universal if and only if $\mathcal{M}$ rejects ξ. Automaton $\mathcal{A}_{\mathcal{M},\xi}$ can be described as the union of several components, where each component accepts the set of timed words violating a requirement of one

[11] Symbol $\gamma \in \Gamma$ denotes a tape cell containing γ, whereas symbol $(q, \gamma) \in Q \times \Gamma$ denotes a tape cell containing γ that is pointed to by the read head while the Turing machine is in control state q.

of the above two conditions. It is well-known that there are simple automata accepting those timed words violating Condition (ii), see, e.g., [2]. We therefore focus on Condition (i).

The key idea for capturing timed words violating (i) is embodied in the automaton $\mathcal{A}_{\mathrm{bit}}$ in Figure 2. (Here we elide the classification of accepting states.) Consider a run of this automaton on a timed word w that matches the regular expression $((\Sigma(0+1)^*)^*\#)^*$ and that does not already violate Condition (ii). Recall that such a timed word copies the sequence of timed events in the time interval $[0,1)$ to all subsequent time intervals. In such a run, clocks x and y are initially reset during the first time unit immediately following respective events σ_1 and σ_2 in Σ, representing two adjacent tape cells in a given configuration of the computation of $\mathcal{M}$. Thereafter, whenever x or y reach 1, they are reset on the next event. By this device we can imagine that automaton $\mathcal{A}_{\mathrm{bit}}$ *scans* the respective bit strings that follow σ_1 and σ_2 in w, checking one bit every time unit. Using a variation on automaton $\mathcal{A}_{\mathrm{bit}}$ we can construct an automaton that accepts precisely when the bit string following σ_2 is not the successor of the bit string following σ_1. Note that this check requires in the worst case 2^n time units.

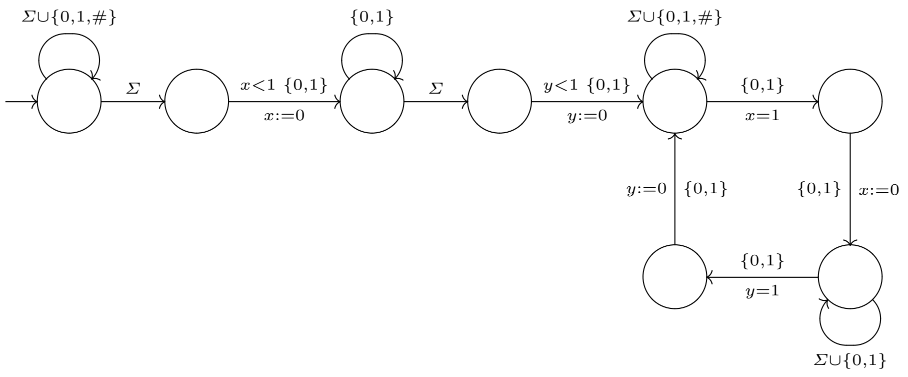


Fig. 2. The timed automaton $\mathcal{A}_{\mathrm{bit}}$.

Let us now assume that the bit strings following each Σ-event in w correctly give the address of the corresponding tape cell. We can devise an automaton that, among such timed words, accepts precisely those that do not correspond to valid accepting computations of $\mathcal{M}$. The automaton detects invalid computations by guessing two Σ-events in the first time unit, corresponding to the same tape cell in consecutive configurations of $\mathcal{M}$, and checking whether this tape cell is correctly updated according to the transition function of $\mathcal{M}$. One can verify that the two events do indeed correspond to the same tape cell using a device similar to that employed in the automaton $\mathcal{A}_{\mathrm{bit}}$: one scans the respective bit-string addresses of the two tape cells, checking that they match in each bit position.

□

References

[1] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. In *Proceedings of LICS*. IEEE Computer Society Press, 1990.

[2] R. Alur and D. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126, 1994.

[3] R. Alur, T. Feder, and T. A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1), 1996.

[4] R. Alur, L. Fix, and T. A. Henzinger. Event-clock automata: A determinizable class of timed automata. *Theor. Comput. Sci.*, 211, 1999.

[5] R. Alur and T. A. Henzinger. Logics and models of real time: A survey. In *REX Workshop*, volume 600 of *LNCS*. Springer, 1991.

[6] R. Alur and T. A. Henzinger. Real-time logics: Complexity and expressiveness. *Inf. Comput.*, 104(1), 1993.

[7] R. Alur, S. La Torre, and P. Madhusudan. Perturbed timed automata. In *Proceedings of HSCC*, volume 3414 of *LNCS*. Springer, 2005.

[8] C. Baier, H. Hermanns, J.-P. Katoen, and B. R. Haverkort. Efficient computation of time-bounded reachability probabilities in uniform continuous-time Markov decision processes. *Theor. Comput. Sci.*, 345(1), 2005.

[9] P. Bouyer, N. Markey, J. Ouaknine, and J. Worrell. On expressiveness and complexity in real-time model checking. In *Proceedings of ICALP*, volume 5126 of *LNCS*. Springer, 2008.

[10] D. Bošnački. Digitization of timed automata. In *Proceedings of FMICS*, 1999.

[11] M. Emmi and R. Majumdar. Decision problems for the verification of real-time software. In *Proceedings of HSCC*, volume 3927 of *LNCS*. Springer, 2006.

[12] D. M. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal basis of fairness. In *Proceedings of POPL*. ACM Press, 1980.

[13] V. Gupta, T. A. Henzinger, and R. Jagadeesan. Robust timed automata. In *Proceedings of HART*, volume 1201 of *LNCS*. Springer, 1997.

[14] T. A. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In *Proceedings of ICALP*, volume 623 of *LNCS*. Springer, 1992.

[15] T. A. Henzinger and J.-F. Raskin. Robust undecidability of timed and hybrid systems. In *Proceedings of HSCC*, volume 1790 of *LNCS*. Springer, 2000.

[16] T. A. Henzinger, J.-F. Raskin, and P.-Y. Schobbens. The regular real-time languages. In *Proceedings of ICALP*, volume 1443 of *LNCS*. Springer, 1998.

[17] P. Herrmann. Timed automata and recognizability. *Inf. Process. Lett.*, 65, 1998.

[18] Y. Hirshfeld and A. Rabinovich. Future temporal logic needs infinitely many modalities. *Inf. Comput.*, 187(2), 2003.

[19] Y. Hirshfeld and A. Rabinovich. Logics for real time: Decidability and complexity. *Fundam. Inform.*, 62(1), 2004.

[20] Y. Hirshfeld and A. Rabinovich. Expressiveness of metric modalities for continuous time. *Logical Methods in Computer Science*, 3(1), 2007.

[21] H. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, 1968.

[22] J.-P. Katoen and I. S. Zapreev. Safe on-the-fly steady-state detection for time-bounded reachability. In *Proceedings of QEST*. IEEE Computer Society Press, 2006.

[23] D. K. Kaynar, N. Lynch, R. Segala, and F. Vaandrager. Timed I/O Automata: A mathematical framework for modeling and analyzing real-time systems. In *Proceedings of RTSS*. IEEE Computer Society Press, 2003.

[24] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4), 1990.

[25] O. Lichtenstein, A. Pnueli, and L. D. Zuck. The glory of the past. In *Logic of Programs*, volume 193 of *LNCS*. Springer, 1985.

[26] N. A. Lynch and H. Attiya. Using mappings to prove timing properties. *Distributed Computing*, 6(2), 1992.

[27] A. R. Meyer. Weak monadic second-order theory of successor is not elementary-recursive. In *Logic Colloquium '72–73*, volume 453 of *LNM*. Springer, 1975.

[28] J. Ouaknine. Digitisation and full abstraction for dense-time model checking. In *Proceedings of TACAS*, volume 2280 of *LNCS*. Springer, 2002.

[29] J. Ouaknine, A. Rabinovich, and J. Worrell. Time-bounded verification (full version). In preparation, 2009.

[30] J. Ouaknine and J. Worrell. Revisiting digitization, robustness, and decidability for timed automata. In *Proceedings of LICS*. IEEE Computer Society Press, 2003.

[31] J. Ouaknine and J. Worrell. Universality and language inclusion for open and closed timed automata. In *Proceedings of HSCC*, volume 2623 of *LNCS*. Springer, 2003.

[32] J. Ouaknine and J. Worrell. On the language inclusion problem for timed automata: Closing a decidability gap. In *Proceedings of LICS*. IEEE Computer Society Press, 2004.

[33] J. Ouaknine and J. Worrell. On the decidability of Metric Temporal Logic. In *Proceedings of LICS*. IEEE Computer Society Press, 2005.

[34] J. Ouaknine and J. Worrell. Safety Metric Temporal Logic is fully decidable. In *Proceedings of TACAS*, volume 3920 of *LNCS*. Springer, 2006.

[35] J. Ouaknine and J. Worrell. On the decidability and complexity of Metric Temporal Logic over finite words. *Logical Methods in Computer Science*, 3(1), 2007.

[36] A. Rabinovich. Finite variability interpretation of monadic logic of order. *Theor. Comput. Sci.*, 275(1-2), 2002.

[37] J.-F. Raskin. *Logics, Automata and Classical Theories for Deciding Real Time.* PhD thesis, University of Namur, 1999.

[38] M. Reynolds. The complexity of temporal logic over the reals. Submitted, 2004.

[39] O. Roux and V. Rusu. Verifying time-bounded properties for ELECTRE reactive programs with stopwatch automata. In *Hybrid Systems*, volume 999 of *LNCS*. Springer, 1994.

[40] S. Shelah. The monadic theory of order. *Ann. Math.*, 102, 1975.

[41] A. P. Sistla, M. Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic (extended abstract). In *Proceedings of ICALP*, volume 194 of *LNCS*. Springer, 1985.

[42] L. J. Stockmeyer. *The complexity of decision problems in automata theory and logic*. PhD thesis, MIT, 1974.

[43] S. Taşiran, R. Alur, R. P. Kurshan, and R. K. Brayton. Verifying abstractions of timed systems. In *Proceedings of CONCUR 96*, volume 1119 of *LNCS*. Springer, 1996.

[44] T. Wilke. Specifying timed state sequences in powerful decidable logics and timed automata. In *FTRTFT*, volume 863 of *LNCS*. Springer, 1994.