On the Decidability of Metric Temporal Logic

Joël Ouaknine
Oxford University Computing Laboratory, Oxford, UK
Email: joel@comlab.ox.ac.uk

James Worrell
Department of Mathematics, Tulane University, USA
Email: jbw@math.tulane.edu

Abstract

Metric Temporal Logic (MTL) is a prominent specification formalism for real-time systems. In this paper, we show that the satisfiability problem for MTL over finite timed words is decidable, with non-primitive recursive complexity. We also consider the model-checking problem for MTL: whether all words accepted by a given Alur-Dill timed automaton satisfy a given MTL formula. We show that this problem is decidable over finite words. Over infinite words, we show that model checking the safety fragment of MTL—which includes invariance and time-bounded response properties—is also decidable. These results are quite surprising in that they contradict various claims to the contrary that have appeared in the literature. The question of the decidability of MTL over infinite words remains open.

1. Introduction

In the linear-temporal-logic approach to verification, an execution of a system is modelled by a sequence of states or events. This representation abstracts away from the precise times of the observations, retaining only their relative order. Such an approach is inadequate to express specifications for systems whose correct behaviour depends on quantitative timing requirements. To address this deficiency, much work has gone into adapting linear temporal logic to the real-time setting; see, e.g., [5], [7], [8], [20], [23], [25], [28].

Real-time logics feature explicit time references, usually by recording timestamps throughout computations. In this paper, we concentrate exclusively on the *dense-time*, or *real-time*, semantics, in which the timestamps are drawn from the set of real numbers.[1]

An important distinction among real-time models is whether one assumes that the system of interest is observed at every instant in time, leading to an *interval-based* semantics [5], [17], [25], or whether one only sees a (possibly countably infinite) sequence of snapshots of the system, leading to a *point-based* semantics [13], [7], [8], [15], [16], [28]. In this paper, we take the latter view: we model the executions of a system as a set of *timed state sequences*.

One of the earliest and most popular suggestions for extending temporal logic to the real-time setting is to replace the temporal operators by time-constrained versions; see [6] and references therein. *Metric Temporal Logic (MTL)*, introduced 15 years ago by Koymans [20], is a prominent and successful instance of this approach.[2] MTL extends Linear Temporal Logic by constraining the temporal operators by (bounded or unbounded) intervals of the real numbers. For example, the formula $\Diamond_{[3,4]}\varphi$ means that φ will become true within 3 to 4 time units from now.

Unfortunately, over the interval-based semantics, the satisfiability and model checking problems for MTL are undecidable [13]. This has led some researchers to consider various restrictions on MTL to recover decidability; see, e.g., [16], [28], [5]. Undecidability arises from the fact that MTL formulas can capture the computations of a Turing machine: configurations of the machine can be encoded within a single unit-duration time interval, since the density of time can accommodate arbitrarily large amounts of information. An MTL formula can then specify that the configurations be accurately propagated from one time interval to the next, in such a way that the timed words satisfying the formula correspond precisely to the halting computations of the Turing machine.

It turns out that the key ingredient required for this procedure to go through is *punctuality*: the ability to specify that a particular event is always followed exactly one time unit later by another one: $\Box(p \rightarrow \Diamond_{=1} q)$. It has in fact been claimed that, in the interval-based and the point-based semantics alike, any logic strong enough to express the above requirement will automatically be undecidable—see [6], [7], [15], among others.

While the claim is correct over the interval-based semantics, we show in this paper that it is erroneous in the point-based semantics. Indeed, we show that both satisfiability and model checking for MTL over finite timed words are decidable, albeit with non-primitive recursive complexity. Over infinite words, we show that model checking the safety fragment of MTL—which includes invariance and punctual time-bounded response properties—is also decidable.

Upon careful analysis, one sees that the undecidability argument breaks down because, over a point-based semantics, MTL is only able to encode *faulty* Turing machines, namely Turing machines suffering from insertion errors. Indeed, while the formula $\Box(p \rightarrow \Diamond_{=1} q)$ ensures that every p is followed exactly one time unit later by a q, there might be some q's that were *not* preceded one time unit earlier by a p. Intuitively, this problem does not occur over the interval-based semantics because the system there is assumed to be under observation at all instants

[1] By contrast, in *discrete-time* settings timestamps are usually integers, which yields more tractable theories that however correspond less closely to physical reality [16], [4].

[2] `http://scholar.google.com` lists over two hundred papers on the subject!

in time, and therefore any insertion error can automatically be detected.

MTL is also genuinely undecidable over a point-based semantics if in addition *past* temporal operators are allowed [7], [13]. Indeed, in this setting insertion errors can be detected by going backwards in time, and MTL formulas are therefore able to precisely capture the computations of perfect Turing machines.[3]

Existing decidability results for MTL involve restrictions either on the semantics or the syntax of the logic to circumvent the problem of punctuality. Alur and Henzinger [7] showed that the satisfiability and model checking problems for MTL relative to a discrete-time semantics are EXPSPACE-complete. Alur, Feder, and Henzinger [5] introduced *Metric Interval Temporal Logic (MITL)* as a fragment of MTL in which the temporal operators may only be constrained by *nonsingular* intervals. They showed that the satisfiability and model checking problems for MITL relative to a dense-time semantics are also EXPSPACE-complete.

The decidability results that we present in this paper are obtained by translating MTL formulas into *timed alternating automata*. These generalize Alur-Dill timed automata, and in particular are closed under complementation. Building on some of our previous work [24], we show that the finite-word language emptiness problem for one-clock timed alternating automata is decidable, which then entails the decidability of MTL satisfiability over finite timed words. We furthermore show how to extend these results to the model checking problems discussed earlier. In addition, we show that MTL formulas can capture the computations of insertion channel machines; using a result of Schnoebelen about the complexity of reachability for lossy channel machines [26], we are then able to give a non-primitive recursive lower bound for the complexity of MTL satisfiability.

We note that a very similar notion of timed alternating automaton has recently and independently been introduced by Lasota and Walukiewicz [21]. They also prove that the finite-word language emptiness problem is decidable for one-clock timed alternating automata, and likewise establish a non-primitive recursive complexity bound for this procedure. They do not, however, consider any questions related to MTL.

In our view, the most interesting unresolved question is whether our decidability results extend to MTL interpreted over *infinite* timed words. We shall briefly return to this in Section 7.

2. Preliminaries

A *time sequence* $\tau = \tau_0\tau_1\ldots$ is a finite or infinite sequence of time values $\tau_i \in \mathbb{R}_{\geq 0}$ satisfying the following constraints (where $|\tau|$ denotes the length of τ):

1) Initialization: $\tau_0 = 0$.
2) Monotonicity: $\tau_i \leq \tau_{i+1}$ for all $i < |\tau| - 1$.
3) Progress: If τ is infinite, then $\{\tau_i : i \in \mathbb{N}\}$ is unbounded.

A *timed word* over finite alphabet Σ is a pair $\rho = (\sigma, \tau)$, where $\sigma = \sigma_0\sigma_1\ldots$ is a word over Σ and τ is a time sequence of the same length. We also represent such a timed word as a sequence of *timed events* by writing $\rho = (\sigma_0, \tau_0)(\sigma_1, \tau_1)\ldots$. Given a timed word $\rho = (\sigma, \tau)$, let $\rho[0\ldots i]$ denote the subword $(\sigma_0, \tau_0)\ldots(\sigma_i, \tau_i)$. Finally, we write $T\Sigma^*$ for the set of finite timed words over alphabet Σ, and $T\Sigma^\omega$ for the set of infinite timed words over Σ. The requirement that the first event of a timed word occur at time 0 is quite natural in the present context since MTL formulas are insensitive to this time value.

A timed language is a set of timed words. A standard way of defining timed languages is via Alur-Dill timed automata [4]. A given Alur-Dill automaton $\mathcal{A}$ accepts a finite timed word iff it has a run over the word that ends in an accepting state. We write $L_f(\mathcal{A})$ for the language of finite timed words accepted by $\mathcal{A}$. We also define the language $L_\omega(\mathcal{A})$ of infinite timed words accepted by $\mathcal{A}$. In this case we assume a Büchi acceptance condition: the automaton accepts a word iff it has an infinite run over the word that visits an accepting state infinitely often.

3. Timed Alternating Automata

In this section we define timed alternating automata. These arise by extending alternating automata [9], [11], [27] with clock variables, in much the same way that Alur-Dill timed automata extend nondeterministic finite automata. A similar notion has independently been investigated by Lasota and Walukiewicz in a recent paper [21].

Timed alternating automata can in general be defined to have any number of clocks. Our goal, however, is to use them to represent metric temporal logic formulas, for which one clock suffices. Accordingly, we shall exclusively focus on one-clock timed alternating automata in this paper.[4] Note also that we only consider timed alternating automata over *finite* timed words.

Let S a finite set of *(control) locations*. The set of formulas $\Phi(S)$ is generated by the grammar:

$$\varphi ::= \top \mid \bot \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid s \mid x \bowtie k \mid x.\varphi,$$

where $k \in \mathbb{N}$, $\bowtie \in \{<, \leq, \geq, >\}$, and $s \in S$. A term of the form $x \bowtie k$ is called a *clock constraint*; here x represents the single clock of the automaton, and the expression $x.s$ is a binding construct corresponding to the operation of resetting the clock to 0.

In the definition of a timed alternating automaton, below, the transition function maps each location $s \in S$ and event $a \in \Sigma$ to an expression in $\Phi(S)$. Thus alternating automata allow two modes of branching: existential branching, represented by disjunction, and universal branching, represented by conjunction.

Definition 1: A timed alternating automaton is a tuple $\mathcal{A} = (\Sigma, S, s_0, F, \delta)$, where

[3]The original undecidability proof in [7] was carried out in a monadic second-order theory of timed state sequences, which subsumes both forward and past temporal operators.

[4]We note in passing that virtually all decision problems, and in particular language emptiness, are in general undecidable for timed alternating automata that have more than one clock.

- Σ is a finite alphabet
- S is a finite set of locations
- $s_0 \in S$ is the initial location
- $F \subseteq S$ is a set of accepting locations
- $\delta \colon S \times \Sigma \to \Phi(S)$ is the transition function.

Remark 2: It will later become apparent that these automata strictly generalize the one-clock Alur-Dill automata considered in [24]. (More generally, timed alternating automata equipped with several clocks are strictly more powerful than Alur-Dill automata with the same number of clocks.)

The notion of a run of a timed alternating automaton, defined below, is somewhat involved, so we first give an example.

Example 3: We define an automaton $\mathcal{A}$ over the singleton alphabet $\Sigma = \{a\}$ that accepts all those timed words in which no two events are separated by exactly one time unit. This language is known not to be expressible as the language of an Alur-Dill timed automaton [18]. The required automaton has set of locations $\{s_0, s_1\}$, with s_0 initial, and both s_0 and s_1 accepting. The transition function is defined by:

$$\begin{aligned} \delta(s_0, a) &= s_0 \wedge x.s_1 \\ \delta(s_1, a) &= s_1 \wedge x \neq 1. \end{aligned}$$

A run of $\mathcal{A}$ starts in location s_0. Every time an a-event occurs, the automaton makes a simultaneous transition to both s_0 and s_1, thus opening up a new thread of computation. The automaton resets a fresh copy of clock x whenever it transitions from location s_0 to s_1, and ensures that no event can happen when this clock equals 1. Every run of this automaton is accepting, since every location is accepting, but there is no run over any word in which two events are separated by exactly one time unit.

We now proceed to the formal definitions. A *state* of $\mathcal{A}$ is a pair (s, v), where $s \in S$ is a location and $v \in \mathbb{R}_{\geq 0}$ is a *clock valuation*. Write $Q = S \times \mathbb{R}_{\geq 0}$ for the set of all possible states.

A set of states $M \subseteq Q$ and a clock valuation $v \in \mathbb{R}_{\geq 0}$ defines a Boolean valuation on $\Phi(S)$ as follows:

- $M \models_v s$ iff $(s, v) \in M$
- $M \models_v x \bowtie k$ iff $v \bowtie k$
- $M \models_v x.\varphi$ iff $M \models_0 \varphi$.

(The Boolean connectives are handled in the expected way.) We say that M is a *minimal model of* $\varphi \in \Phi(S)$ *with respect to* v if $M \models_v \varphi$ and there is no proper subset $N \subset M$ with $N \models_v \varphi$. Also, if $\varphi \in \Phi(S)$ is a closed formula, i.e., every occurrence of x lies within the scope of a binding operator $x.-$, then the relation $M \models_v \varphi$ is independent of the value of v, and we feel free to omit it.

A *configuration* of $\mathcal{A}$ is a finite set of states; the set of configurations is denoted $\wp(Q)$. The *initial configuration* is $\{(s_0, 0)\}$ and a configuration is *accepting* if every location that it contains is accepting. Note in particular that the empty configuration is always accepting. The language accepted by a timed alternating automaton over finite words can be described in terms of a transition system of configurations, defined below.

Definition 4: Given a timed alternating automaton $\mathcal{A}$, we define the labelled transition system $\mathcal{T}_\mathcal{A} = (\wp(Q), \rightsquigarrow, \rightarrow)$ over the set of configurations as follows. The $(\mathbb{R}_{\geq 0})$-labelled transition relation $\rightsquigarrow \subseteq \wp(Q) \times \mathbb{R}_{\geq 0} \times \wp(Q)$ captures time evolutions, or *flow steps*, and is defined by

$$C \overset{t}{\rightsquigarrow} C' \text{ if } C' = \{(s, v + t) : (s, v) \in C\}.$$

The Σ-labelled transition relation $\rightarrow \subseteq \wp(Q) \times \Sigma \times \wp(Q)$ captures instantaneous changes in the locations, or *edge steps*. Let $C = \{(s_i, v_i)\}_{i \in I}$. Then $C \xrightarrow{a} C'$ if

$$C' = \bigcup_{i \in I} \{M_i : M_i \text{ is some minimal model of } \delta(s_i, a) \text{ with respect to } v_i\}.$$

Let $\rho = (\sigma, \tau)$ be a finite timed word with $|\rho| = n$. Write $d_i = \tau_{i+1} - \tau_i$ for the time delay between the $(i+1)$-st and i-th events. Define a *run* of $\mathcal{A}$ on ρ to be a finite alternating sequence of edge steps and flow steps in $\mathcal{T}_\mathcal{A}$:

$$C_0 \xrightarrow{\sigma_0} C_1 \overset{d_0}{\rightsquigarrow} C_2 \xrightarrow{\sigma_1} C_3 \overset{d_1}{\rightsquigarrow} \cdots \overset{d_{n-1}}{\rightsquigarrow} C_{2n} \xrightarrow{\sigma_n} C_{2n+1},$$

where C_0 is the initial configuration. The run is *accepting* if the last configuration C_{2n+1} is accepting, and the timed word ρ is *accepted* by $\mathcal{A}$ if there is some accepting run of $\mathcal{A}$ on ρ. We write $L_f(\mathcal{A}) \subseteq T\Sigma^*$ for the language of finite timed words accepted by $\mathcal{A}$.[5]

Example 5: A time-bounded response property such as 'for every a-event there is a b-event exactly one time unit later' can be expressed by the following automaton. Let $\mathcal{A}$ have two locations $\{s_0, s_1\}$ with s_0 the initial and only accepting location, and transition function δ given by the following table:

	a	b
s_0	$s_0 \wedge x.s_1$	s_0
s_1	s_1	$(x = 1) \vee s_1$

3.1. Duality and Complementation

The following derivation shows that the class of languages definable by timed alternating automata is closed under complement. Since it is straightforward to show that this class is also closed under union, timed alternating automata are closed under all Boolean operations. Note that the arguments presented here are similar to the untimed case [9], [11], and therefore we only sketch the proofs.

Given $\varphi \in \Phi(S)$ we define its dual $\overline{\varphi} \in \Phi(S)$ as follows. The dual of a clock constraint is its negation (e.g., $\overline{x < k} = x \geq k$), $\overline{x.\varphi} = x.\overline{\varphi}$ and $\overline{s} = s$. For the propositional connectives we have the usual de Morgan dualities: $\overline{\varphi_1 \vee \varphi_2} = \overline{\varphi_1} \wedge \overline{\varphi_2}$ and $\overline{\varphi_1 \wedge \varphi_2} = \overline{\varphi_1} \vee \overline{\varphi_2}$.

Let $\mathcal{A} = (\Sigma, S, s_0, F, \delta)$ be an alternating timed automaton. The complement automaton $\mathcal{A}^c$ is defined by $\mathcal{A}^c =$

[5] It is usual to define a run of an alternating automaton as a *tree* of states. However, over finite words one can equivalently define a run as a sequence of configurations, where each configuration represents a given level of the run tree.

$(\Sigma, S, s_0, S \setminus F, \overline{\delta})$, where $\overline{\delta}(s,a) = \overline{\delta(s,a)}$ for each $s \in S$ and $a \in \Sigma$.

Proposition 6: Let $\varphi \in \Phi(S)$, $v \in \mathbb{R}_{\geq 0}$, and let $R \subseteq Q$ be a set of states; then $R \models_v \varphi$ iff $Q \setminus R \not\models_v \overline{\varphi}$.

Proof: Straightforward structural induction on φ. ∎

Proposition 7: $L(\mathcal{A}) \cap L(\mathcal{A}^c) = \emptyset$.

Proof: Suppose there is a timed word ρ such that $\mathcal{A}$ and $\mathcal{A}^c$ both have runs on ρ. Using Proposition 6 one easily shows by induction on $|\rho|$ that the last configurations in each run have a state in common. It follows that the runs cannot both be accepting. ∎

Proposition 8: $L(\mathcal{A}) \cup L(\mathcal{A}^c) = T\Sigma^*$.

Proof: We claim that, given a finite timed word $\rho = (\sigma, \tau)$ and a set of states $R \subseteq Q$, either $\mathcal{A}$ has a run on ρ whose last configuration is a subset of R or $\mathcal{A}^c$ has a run on ρ whose last configuration is a subset of $Q \setminus R$. We prove this claim by induction on $|\rho|$ as follows.

Let $\rho = (\sigma, \tau)$ and $R \subseteq Q$ be given as in the claim, with $|\rho| = n + 1$. Also, let $d_n = \tau_{n+1} - \tau_n$ and write

$$R' = \{(s, v) : R \models_{v+d_n} \delta(s, \sigma_{n+1})\}.$$

By induction, either $\mathcal{A}$ has a run on $\rho[0 \ldots n-1]$ whose last configuration is a subset of R', or $\mathcal{A}^c$ has a run on $\rho[0 \ldots n-1]$ whose last configuration is a subset of $Q \setminus R'$. In the former case it is immediate that $\mathcal{A}$ has a run on ρ whose last configuration is a subset of R. In the latter case it follows from Proposition 6 that $\mathcal{A}^c$ has a run on ρ whose last configuration is a subset of $Q \setminus R$.

The proposition now follows from the claim by taking R to be the set of states whose underlying location is accepting. ∎

Summarizing:

Corollary 9: The class of languages definable by timed alternating automata is effectively closed under all Boolean operations.

3.2. Decidability of Language Emptiness

It is well known that the universality problem for Alur-Dill timed automata is undecidable [4]. Since the class of multi-clock timed alternating automata is closed under complement and includes the class of Alur-Dill automata, the language-emptiness problem for multi-clock timed alternating automata cannot be decidable. However, in [24] we showed that the universality problem for Alur-Dill automata that have at most one clock is decidable. Using similar techniques we now show that the language-emptiness problem for one-clock alternating automata is decidable.

The language-emptiness problem for a one-clock alternating automaton $\mathcal{A} = (\Sigma, S, s_0, F, \delta)$ is equivalent to the following reachability question on the derived transition system $\mathcal{T}_{\mathcal{A}}$: 'Is there a path from the initial configuration to an accepting configuration?'. Since $\mathcal{T}_{\mathcal{A}}$ has uncountably many states, and indeed each state has uncountably many successors under the flow-step relation, some abstraction is needed to explore the state space. The following definitions establish the groundwork for this abstraction.

Let k be a positive integer. Define an equivalence relation $\sim_k$ on $\mathbb{R}_{\geq 0}$ by $u \sim_k v$ if either $u, v > k$, or $\lceil u \rceil = \lceil v \rceil$ and $\lfloor u \rfloor = \lfloor v \rfloor$. The corresponding set of equivalence classes, or *regions*, is $REG_k = \{\mathsf{r}_0, \mathsf{r}_1, \ldots, \mathsf{r}_{2k+1}\}$, where $\mathsf{r}_{2i} = \{i\}$ for $i \leq k$, $\mathsf{r}_{2i+1} = (i, i+1)$ for $i < k$, and $\mathsf{r}_{2k+1} = (k, \infty)$. Let $reg_k(u)$ denote the equivalence class of $u \geq 0$. In practice we prefer to omit explicit reference to the threshold k in our notation, and infer it from the context. Thus we adopt the convention that whenever u, v are clock values of a timed automaton $\mathcal{A}$, then $u \sim v$ means $u \sim_k v$, where k is the largest constant appearing in $\mathcal{A}$.

The *fractional part* of a nonnegative real $x \in \mathbb{R}_{\geq 0}$ is $\mathit{frac}(x) = x - \lfloor x \rfloor$. Using this notion we define the relation $\approx$ on $(\mathbb{R}_{\geq 0})^n$—an n-dimensional analog of $\sim$, also depending on an invisible threshold k—by $\mathbf{u} \approx \mathbf{v}$ iff $u_i \sim v_i$ for each $i \in \{1, \ldots, n\}$ and $\mathit{frac}(u_i) \leq \mathit{frac}(u_j)$ iff $\mathit{frac}(v_i) \leq \mathit{frac}(v_j)$ for all $i, j \in \{1, \ldots, n\}$.

The following is a standard result; see, e.g., [4].

Proposition 10: Let $\mathbf{u}, \mathbf{v} \in (\mathbb{R}_{\geq 0})^n$ with $\mathbf{u} \approx \mathbf{v}$. Then for all $t \geq 0$ there exists $t' \geq 0$ such that $(\mathbf{u} + t) \approx (\mathbf{v} + t')$.

Definition 11: An equivalence relation $R \subseteq \wp(Q) \times \wp(Q)$ is a *time-abstract bisimulation* on $\mathcal{T}_{\mathcal{A}}$ if $p \; R \; q$ implies

- $(\forall a \in \Sigma)(p \xrightarrow{a} p'$ implies $\exists q'(q \xrightarrow{a} q'$ and $p' \; R \; q'))$
- $(\forall t \in \mathbb{R}_{\geq 0})(p \overset{t}{\rightsquigarrow} p'$ implies $\exists t' \exists q'(q \overset{t'}{\rightsquigarrow} q'$ and $p' \; R \; q'))$.

To better understand the notion of bisimulation in this particular setting, we take another look at the notion of minimal model underlying the edge-transition relation.

Any formula $\varphi \in \Phi(S)$ can be written in disjunctive normal form $\varphi \equiv \bigvee_{i \in I} \bigwedge A_i$, where each A_i is a set of terms of the form s, $x.s$, and $x \bowtie k$ (which we call *atoms*). The minimal models of φ can be read off from the disjunctive normal form as follows. For a set of atoms A and a clock valuation $v \in \mathbb{R}_{\geq 0}$, let $A[v] \subseteq Q$ be the set of states given by $A[v] = \{(s, v) : s \in A\} \cup \{(s, 0) : x.s \in A\}$. Then each minimal model M of φ with respect to v has the form $M = A_i[v]$ for some $i \in I$ where v satisfies all the clock constraints in A_i.

Lemma 12—Bisimulation Lemma: Define the relation $R \subseteq \wp(Q) \times \wp(Q)$ by $C \; R \; D$ iff there is a bijection $f\colon C \to D$ such that: (i) $f(s, u) = (t, v)$ implies $s = t$ and $u \sim v$; (ii) If $f(s, u) = (t, v)$ and $f(s', u') = (t', v')$, then $\mathit{frac}(u) \leq \mathit{frac}(u')$ iff $\mathit{frac}(v) \leq \mathit{frac}(v')$. Then R is a time-abstract bisimulation on $\mathcal{T}_{\mathcal{A}}$.

Proof: Suppose that $C = \{(s_i, u_i)\}_{i \in I}$ and $D = \{(t_i, v_i)\}_{i \in I}$ are configurations of $\mathcal{A}$, and that $f\colon C \to D$, where $f(s_i, u_i) = (t_i, v_i)$, is a bijection witnessing $C \; R \; D$.

Matching edge transitions: Suppose C makes an edge transition $C \xrightarrow{a} C'$ for some $a \in \Sigma$. By the above considerations on minimal models we know that $C' = \bigcup_{i \in I} A_i[u_i]$, where, for each $i \in I$, the set of atoms A_i is a clause in the disjunctive normal form expression for $\delta(s_i, a)$. Setting $D' = \bigcup_{i \in I} A_i[v_i]$ we have $D \xrightarrow{a} D'$ and $C' \; R \; D'$. (We leave it to the reader to construct a suitable bijection $f'\colon C' \to D'$ witnessing $C' \; R \; D'$.)

Matching flow transitions: Suppose C makes a flow transition $C \overset{t}{\rightsquigarrow} C'$ for some $t \in \mathbb{R}_{\geq 0}$. Writing $\mathbf{u} = (u_i)_{i \in I}$ and $\mathbf{v} = (v_i)_{i \in I}$, notice that $C \; R \; D$ implies that $\mathbf{u} \approx \mathbf{v}$ in the sense of Proposition 10. By that proposition there exists t' with $(\mathbf{u}{+}t) \approx (\mathbf{v}{+}t')$. Thus, writing $D' = D{+}t'$, we have $D \overset{t'}{\rightsquigarrow} D'$ and $C' \; R \; D'$. ∎

Motivated by the Bisimulation Lemma, we define an *abstract configuration* to be a finite word over the alphabet $\Lambda = \wp(S \times \mathit{REG})$. We also define an *abstraction function* H mapping each $\mathcal{A}$-configuration C to an abstract configuration $H(C) \in \Lambda^*$. The definition of H involves an auxiliary function $\mathit{Abs}\colon \wp(Q) \to \Lambda$, where $\mathit{Abs}(C) = \{(s, \mathit{reg}(u)) : (s, u) \in C\}$. Now given an $\mathcal{A}$-configuration C, to define $H(C)$, we partition C into a sequence of subsets $C_1, \ldots, C_n$, such that for all $(s, u) \in C_i$ and $(t, v) \in C_j$, $\mathit{frac}(u) \leq \mathit{frac}(v)$ iff $i \leq j$; then $H(C) = \mathit{Abs}(C_1) \ldots \mathit{Abs}(C_n) \in \Lambda^*$.

Proposition 13: If C and C' are $\mathcal{A}$-configurations with $H(C) = H(C')$, then C and C' are bisimilar in $\mathcal{T}_\mathcal{A}$.

Proof: The relation $\{(C, C') : H(C) = H(C')\}$ on configurations satisfies the hypotheses of the Bisimulation Lemma. ∎

Example 14: Suppose the largest constant appearing in $\mathcal{A}$ is 2, so that the corresponding set of clock regions is $\mathit{REG}_2 = \{\mathsf{r}_0, \mathsf{r}_1, \ldots, \mathsf{r}_5\}$. If $C = \{(s_0, 0.8), (s_1, 1), (s_1, 0.3), (s_1, 1.8), (s_2, 2.1)\}$ is an $\mathcal{A}$-configuration, then the corresponding abstract configuration is $H(C) = \{(s_1, \mathsf{r}_2)\}\{(s_2, \mathsf{r}_5)\}\{(s_1, \mathsf{r}_1)\}\{(s_0, \mathsf{r}_1), (s_1, \mathsf{r}_3)\}$.

Definition 15: The time-abstract transition system $\mathcal{W}_\mathcal{A} = (\Lambda^*, \rightsquigarrow, \rightarrow)$ is defined as a quotient of $\mathcal{T}_\mathcal{A}$ under H as follows. The state space is the set Λ^* of finite words over alphabet Λ. The *unlabelled* flow-step transition relation $\rightsquigarrow \subseteq \Lambda^* \times \Lambda^*$ is defined implicitly by $H(C) \rightsquigarrow H(C')$ iff there exists $t \geq 0$ with $C \overset{t}{\rightsquigarrow} C'$ in $\mathcal{T}_\mathcal{A}$. This is well-defined by Proposition 13. Similarly, we define the edge-step transition relation $\rightarrow \subseteq \Lambda^* \times \Sigma \times \Lambda^*$ implicitly by $H(C) \xrightarrow{a} H(C')$ iff $C \xrightarrow{a} C'$ in $\mathcal{T}_\mathcal{A}$—again well-defined by Proposition 13. A state of $\mathcal{W}_\mathcal{A} = (\Lambda^*, \rightsquigarrow, \rightarrow)$ is said to be *initial* (respectively, *accepting*) iff it is the image under H of an initial (respectively, accepting) state of $\mathcal{T}_\mathcal{A}$.

Although the above definition of the transition structure of $\mathcal{W}_\mathcal{A}$ is implicit, it is routine to show that each state $w \in \mathcal{W}_\mathcal{A}$ has only finitely many successors under edge steps and flow steps, and moreover these successors may be effectively calculated from w itself and the description of $\mathcal{A}$.

We have now reduced the language-emptiness problem for $\mathcal{A}$ to the following reachability question for $\mathcal{W}_\mathcal{A}$: 'Is there a path from the initial configuration to an accepting configuration?'. Given that $\mathcal{W}_\mathcal{A}$ has infinitely many states, it is not obvious that this problem is decidable. However, as we outline below, there is a fairly standard theory of *well-structured transition systems* [12] that can be applied to establish decidability.

An infinite sequence $w_1, w_2, w_3, \ldots$ in a partially ordered set $(W, \preccurlyeq)$ is said to be *saturating* if there exist indices $i < j$ such that $w_i \preccurlyeq w_j$. The partial order $\preccurlyeq$ is a *well-partial-order* (wpo for short) if every infinite sequence is saturating.

Let $\leqslant$ be a partial order on an alphabet Λ. Define the induced *monotone domination order* $\preccurlyeq$ on Λ^*, the set of finite words over Λ, by $a_1 \ldots a_m \preccurlyeq b_1 \ldots b_n$ if there exists a strictly increasing function $f\colon \{1 \ldots m\} \to \{1, \ldots, n\}$ such that $a_i \leqslant b_{f(i)}$ for all $i \in \{1, \ldots, m\}$.

Lemma 16—Higman's Lemma [19]: If $\leqslant$ is a wpo on Λ, then the induced monotone domination order $\preccurlyeq$ is a wpo on Λ^*.

Definition 17: A *well-structured transition system* is a triple $\mathcal{W} = (W, \preccurlyeq, \rightarrow)$, where $(W, \rightarrow)$ is a finitely-branching transition system equipped with a wpo $\preccurlyeq$ such that:

- $\preccurlyeq$ is a decidable relation
- $\mathit{Succ}(w) := \{w' : \exists l (w \xrightarrow{l} w')\}$ is effectively calculable for each $w \in W$
- $\preccurlyeq$ is downward compatible: if $w, v \in W$ with $w \preccurlyeq v$ then for any transition $v \xrightarrow{l} v'$ there exists a matching transition $w \xrightarrow{l} w'$ with $w' \preccurlyeq v'$.

Proposition 18: [12, Theorem 5.5] Let $\mathcal{W} = (W, \preccurlyeq, \rightarrow)$ be a WSTS. Given a state $u \in W$ and a subset $V \subseteq W$ that is downward-closed with respect to $\preccurlyeq$, it is decidable whether there is a sequence of transitions starting at u and ending in V.

Proposition 19: The transition system $\mathcal{W}_\mathcal{A}$ as defined above is a WSTS.

Proof: Recall that the state space is Λ^*, where $\Lambda = \wp(S \times \mathit{REG})$ with S the set of locations of $\mathcal{A}$. By Lemma 16, the induced monotone domination order $\preccurlyeq$ on Λ^* with respect to the set-inclusion order on Λ is a wpo. It remains to show that the edge-step and flow-step transition relations on $\mathcal{W}_\mathcal{A}$ are downward compatible. We consider the edge-step relation; the case for the flow-step relation is almost identical.

Suppose that $u, v, v' \in \Lambda^*$ are such that $u \preccurlyeq v$ and $v \xrightarrow{a} v'$ for some $a \in \Sigma$. Then, by definition of $\mathcal{W}_\mathcal{A}$, there exist configurations C, D, D' of $\mathcal{A}$ such that $H(C) = u, H(D) = v, H(D') = v', C \subseteq D$ and $D \xrightarrow{a} D'$. Since edge-steps between configurations are computed pointwise, the set-inclusion relation $\subseteq$ is clearly downward compatible with the transition structure on $\mathcal{T}_\mathcal{A}$: in other words, there exists a configuration C' with $C \xrightarrow{a} C'$ and $C' \subseteq D'$. Thus, writing $u' = H(C')$, we have $u' \preccurlyeq v'$ and $u \xrightarrow{a} u'$, as required. ∎

Since a state of $\mathcal{W}_\mathcal{A}$ is accepting if it only mentions accepting locations of $\mathcal{A}$, the set of accepting states of $\mathcal{W}_\mathcal{A}$ is downward-closed with respect to the monotone domination order on Λ^*. By Proposition 18, it follows that the language-emptiness problem is decidable for one-clock alternating automata. This proves the first assertion of Theorem 20 below. The second assertion can be proved in a similar way by making use of the abstractions introduced in [24] to prove the decidability of language inclusion for one-clock Alur-Dill automata; we omit the details. As noted earlier, these results have been independently obtained by Lasota and Walukiewicz [21].

Theorem 20: Let $\mathcal{A}$ range over the class of one-clock timed alternating automata and $\mathcal{B}$ over the class of Alur-Dill timed automata. The language-emptiness problem '$L_f(\mathcal{A}) = \emptyset$?' and the language-inclusion problem '$L_f(\mathcal{B}) \subseteq L_f(\mathcal{A})$?' are both decidable.

4. Metric Temporal Logic

In this section we define the syntax and semantics of Metric Temporal Logic (MTL). As discussed in the Introduction, there are two different dense-time semantics for MTL: *point-based* and *interval-based*, and for our concerns the difference is crucial. Following [13], [7], [8], [15], [16], [28], among others, we adopt a point-based semantics over timed words. A key observation about this semantics is that the temporal connectives quantify over a countable set of positions in a timed word. In contrast, the interval-based semantics, adopted in, e.g., [5], [17], [25], associates a state to each point in real time, and the temporal connectives quantify over the whole time domain. In the interval-based semantics one can use a formula of the type $\Box(p \rightarrow \Diamond_{=1} q)$ to specify a perfect channel, whereas in the point-based semantics the same formula only specifies a channel with insertion errors (see Section 5.2). This observation helps understand why MTL is undecidable under the interval-based semantics, whereas, at least over finite words, it is decidable in the point-based semantics (Theorem 24).

Given our use of the point-based semantics, it is quite natural to present MTL as a formalism for reasoning about sequences of *events* rather than sequences of *states*, although it should be noted that all our results readily carry over to the state-based setting.[6]

Definition 21: Given an alphabet Σ of atomic events, the formulas of MTL are built up from Σ by Boolean connectives and time-constrained versions of the *until* operator $\mathcal{U}$ as follows:

$$\varphi ::= \top \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid a \mid \varphi_1 \,\mathcal{U}_I\, \varphi_2$$

where $a \in \Sigma$, 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 $\mathcal{U}_I$. We also use pseudo-arithmetic expressions to denote intervals. For example, the expression '≥ 1' denotes $[1, \infty)$ and '$= 1$' denotes the singleton $\{1\}$.

Following [15], we give a *strict-future* interpretation to the temporal connectives $\mathcal{U}_I$. This choice does not affect the results of this paper since, as with Linear Temporal Logic, any property that can be expressed by an MTL formula under the non-strict semantics can also be expressed by an MTL formula under the strict semantics.[7] However the strict semantics allows a more uniform presentation of the tableau construction for formulas.

Definition 22: Given a (finite or infinite) timed word $\rho = (\sigma, \tau)$ and an MTL formula φ, the satisfaction relation $(\rho, i) \models \varphi$ (read ρ satisfies φ at position i) is defined inductively as follows:

- $(\rho, i) \models \top$
- $(\rho, i) \models \varphi_1 \wedge \varphi_2$ iff $(\rho, i) \models \varphi_1$ and $(\rho, i) \models \varphi_2$
- $(\rho, i) \models \neg\varphi$ iff $(\rho, i) \not\models \varphi$
- $(\rho, i) \models a$ iff $i < |\rho|$ and $\sigma_i = a$
- $(\rho, i) \models \varphi_1 \,\mathcal{U}_I\, \varphi_2$ iff there exists j such that $i < j < |\rho|$, $(\rho, j) \models \varphi_2$, $\tau_j - \tau_i \in I$, and $(\rho, k) \models \varphi_1$ for all k with $i < k < j$.

We say that ρ satisfies φ, denoted $\rho \models \varphi$, if $(\rho, 0) \models \varphi$. The set of finite models of an MTL formula φ is given by $L_f(\varphi) = \{\rho \in T\Sigma^* : \rho \models \varphi\}$. The set of infinite models of φ is given by $L_\omega(\varphi) = \{\rho \in T\Sigma^\omega : \rho \models \varphi\}$.

Additional temporal operators can be defined using the usual conventions. In particular, since we adopt a strict-future semantics for $\mathcal{U}_I$, we recover the *constrained next time* operator $\bigcirc_I \varphi \equiv \bot \,\mathcal{U}_I\, \varphi$. We have the *constrained eventually* operator $\Diamond_I \varphi \equiv \top \,\mathcal{U}_I\, \varphi$, and the *constrained always* operator $\Box_I \varphi \equiv \neg\Diamond_I \neg\varphi$. The *dual-until* operator is given by $\varphi_1 \,\widetilde{\mathcal{U}}_I\, \varphi_2 \equiv \neg((\neg\varphi_1) \,\mathcal{U}_I\, (\neg\varphi_2))$. Using the dual-until operator and the additional propositional connectives $\vee$ and $\bot$ we can rewrite every MTL formula into an equivalent formula in *positive normal form*, i.e., where negation is only applied to actions $a \in \Sigma$.

5. MTL over Finite Words

In this section we consider the *satisfiability problem* for MTL over finite words: 'Given an MTL formula φ, is $L_f(\varphi)$ nonempty?'. We also consider the following *model-checking problem*: 'Given an MTL formula φ and an Alur-Dill timed automaton $\mathcal{B}$, is it the case that $L_f(\mathcal{B}) \subseteq L_f(\varphi)$?'. In both cases we show decidability by translating the MTL formulas into equivalent one-clock timed alternating automata and invoking Theorem 20. We also show that both problems have non-primitive recursive complexity.

5.1. Decidability

Given an MTL formula φ in positive normal form, we define a one-clock alternating automaton $\mathcal{A}_\varphi$ such that $L_f(\mathcal{A}_\varphi) = L_f(\varphi)$. Since timed alternating automata are closed under union and intersection, and since it is clear how to define $\mathcal{A}_\varphi$ in case φ is an atomic formula or the negation of an atomic formula, without loss of generality we assume that the outermost connective in φ is $\mathcal{U}_I$ or $\widetilde{\mathcal{U}}_I$.

Define the *closure* of φ, denoted $cl(\varphi)$, to consist of all subformulas of φ whose outermost connective is $\mathcal{U}_I$ or $\widetilde{\mathcal{U}}_I$. We define $\mathcal{A}_\varphi = (\Sigma, S, s_0, F, \delta)$ as follows. The set of locations is $S = cl(\varphi) \cup \{\varphi^i\}$. We call φ^i the *initial copy of* φ: it is the initial location of $\mathcal{A}_\varphi$. A location $\psi \in S$ is accepting iff its outermost connective is $\widetilde{\mathcal{U}}_I$.

We define the transition function δ so that the presence of a fresh copy of ψ in a configuration during a run of $\mathcal{A}_\varphi$ ensures that the input word satisfies ψ at the current position. To enforce this requirement, when ψ is encountered the automaton

[6] In very rough terms, one can translate problems in a state-based setting over a set of atomic state propositions P into an equivalent problem in an event-based setting over the set of events 2^P.

[7] This statement is easily proved by induction on MTL formulas. Intuitively, the two semantics only differ in their interpretation of the until operator, for which the strict-future semantics does not place any obligations on the current state, only on subsequent ones. For example, the formula $a \,\mathcal{U}_{[0,3)}\, b$, interpreted under the *non-strict* semantics, is equivalent to $((a \,\mathcal{U}_{[0,3)}\, b) \wedge a) \vee b$.

starts a fresh clock and thereafter propagates ψ from configuration to configuration in the run until all the obligations that it stipulates are discharged. We first define an auxiliary function $init(\psi, a) \in \Phi(S)$ for each subformula ψ of φ and $a \in \Sigma$ as follows:

$$\begin{array}{rcl} init(\psi, a) & = & x.\psi, \text{ if } \psi \in cl(\varphi) \\ init(\psi_1 \wedge \psi_2, a) & = & init(\psi_1, a) \wedge init(\psi_2, a) \\ init(\psi_1 \vee \psi_2, a) & = & init(\psi_1, a) \vee init(\psi_2, a) \\ init(b, a) & = & \left\{ \begin{array}{ll} \top & \text{if } a = b \\ \bot & \text{if } a \neq b \end{array} \right. \text{ for } b \in \Sigma \\ init(\neg b, a) & = & \neg init(b, a). \end{array}$$

Now δ is defined by:

$$\begin{array}{rcl} \delta(\varphi^i, a) & = & \varphi \\ \delta(\psi_1 \, \mathcal{U}_I \, \psi_2, a) & = & (init(\psi_2, a) \wedge x \in I) \vee \\ & & (init(\psi_1, a) \wedge (\psi_1 \, \mathcal{U}_I \, \psi_2)) \\ \delta(\psi_1 \, \widetilde{\mathcal{U}}_I \, \psi_2, a) & = & (init(\psi_2, a) \vee x \notin I) \wedge \\ & & (init(\psi_1, a) \vee (\psi_1 \, \widetilde{\mathcal{U}}_I \, \psi_2)). \end{array}$$

Proposition 23: $L_f(\mathcal{A}_\varphi) = L_f(\varphi)$.

Proof: We first show that $L_f(\mathcal{A}_\varphi) \subseteq L_f(\varphi)$. To this end, let $\rho = (\sigma, \tau)$ be a timed word in $L_f(\mathcal{A}_\varphi)$. As usual, write $d_i = \tau_{i+1} - \tau_i$. Suppose that $\mathcal{A}_\varphi$ has an accepting run on ρ:

$$C_0 \xrightarrow{\sigma_0} C_1 \stackrel{d_0}{\rightsquigarrow} C_2 \xrightarrow{\sigma_1} C_3 \stackrel{d_1}{\rightsquigarrow} \cdots \xrightarrow{\sigma_n} C_{2n+1}.$$

We claim that for each subformula ψ of φ and each $i < |\rho|$, if $C_{2i+1} \models init(\psi, \sigma_i)$ then $(\rho, i) \models \psi$. We prove this claim by structural induction on ψ. The only non-trivial cases are when the outermost connective of ψ is a temporal modality. We consider the case $\psi \equiv \psi_1 \, \mathcal{U}_I \, \psi_2$ by way of example.

Suppose that $\psi \equiv \psi_1 \, \mathcal{U}_I \, \psi_2$ and $C_{2i+1} \models init(\psi, \sigma_i)$. Then $init(\psi, \sigma_i) = x.\psi$ and $(\psi, 0) \in C_{2i+1}$. Analyzing the definition of the transition function δ we can show that for each successive value of $j \geq i$ we have that $C_{2j+1} \models init(\psi_1, \sigma_j)$ and $(\psi, \tau_j - \tau_i) \in C_{2j+1}$ until at some point $C_{2j+1} \models init(\psi_2, \sigma_j)$ and $\tau_j - \tau_i \in I$. (Note that the latter must eventually occur since ψ is not an accepting location.) From the induction hypothesis it is clear that this implies that $(\rho, i) \models \psi$.

Having proved the claim we observe that $(\varphi, 0) \in C_1$, and so $C_1 \models init(\varphi, \sigma_0)$. Thus, applying the claim in the case $i = 0$ and $\psi \equiv \varphi$, we immediately get that $\rho \models \varphi$ whenever $\mathcal{A}_\varphi$ has an accepting run on ρ. This completes the proof that $L_f(\mathcal{A}_\varphi) \subseteq L_f(\varphi)$.

It remains to show the converse inclusion: $L_f(\varphi) \subseteq L_f(\mathcal{A}_\varphi)$. But this follows from $L_f(\mathcal{A}_\varphi) \subseteq L_f(\varphi)$ and the observation that $\mathcal{A}_{\neg\varphi} = (\mathcal{A}_\varphi)^c$. ■

In conjunction with Theorem 20, Proposition 23 immediately yields:

Theorem 24: The satisfiability and the model-checking problems for MTL over finite words are both decidable.

5.2. Complexity

Using a result of Schnoebelen [26] about channel systems, we prove that the satisfiability problem for MTL has non-primitive recursive complexity.

A *channel machine* consists of a finite-state automaton acting on an unbounded fifo channel. More precisely, a channel machine is a tuple $\mathcal{C} = (S, M, \Delta)$, where S is a finite set of *control states*, M is a finite set of *messages*, and $\Delta \subseteq S \times \Sigma \times S$ is the transition relation over label set $\Sigma = \{m!, m? : m \in M\}$. A transition labelled $m!$ writes message m to the channel, and a transition labelled $m?$ reads message m from the channel.

We define an operational semantics for channel machines as follows. A *global state* of $\mathcal{C}$ is a pair $\gamma = (s, x)$, where $s \in S$ is the control state and $x \in M^*$ represents the contents of the channel. The rules in Δ induce a Σ-labelled transition relation on the set of global states as follows: $(s, m!, t) \in \Delta$ yields a transition $(s, x) \xrightarrow{m!} (t, x \cdot m)$ that writes $m \in M$ to the tail of the channel, and $(s, m?, t) \in \Delta$ yields a transition $(s, m \cdot x) \xrightarrow{m?} (t, x)$ that reads $m \in M$ from the head of the channel. If we only allow the transitions indicated above, then we call $\mathcal{C}$ an *error-free* channel machine. A *computation* of such a machine is a finite sequence of transitions between global states

$$(s_0, x_0) \xrightarrow{\alpha_0} (s_1, x_1) \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_{n-1}} (s_n, x_n).$$

We also consider channel machines that operate with *insertion errors*. Given $x, y \in M^*$, write $x \sqsubseteq y$ if x can be obtained from y by deleting any number of letters, e.g., sub $\sqsubseteq$ <u>s</u>t<u>ub</u>born, as indicated by the underlining. (This is an instance of the monotone domination order introduced earlier.) Following [26] we enable *insertion errors* by extending the transition relation on global states with the following clause: if $(s, x) \xrightarrow{\alpha} (t, y)$, $x' \sqsubseteq x$ and $y \sqsubseteq y'$, then $(s, x') \xrightarrow{\alpha} (t, y')$. Dually, we define *lossy channel machines* by adding a clause: if $(s, x) \xrightarrow{\alpha} (t, y)$, $x \sqsubseteq x'$ and $y' \sqsubseteq y$, then $(s, x') \xrightarrow{\alpha} (t, y')$. The notion of a computation of a channel machine with insertion errors or lossiness errors is defined analogously to the error-free case.

The *control-state reachability problem* asks, given a channel machine $\mathcal{C} = (S, M, \Delta)$ and two distinct control states $s_{init}, s_{fin} \in S$, whether there is a finite computation of $\mathcal{C}$ starting in global state (s_{init}, ε) and ending in global state (s_{fin}, x) for some $x \in M^*$. This problem was proved to be decidable for lossy channel machines by Abdulla and Jonsson [3]. Later Schnoebelen [26] showed that it has non-primitive recursive complexity. The *dual control-state reachability problem* asks, given a channel machine $\mathcal{C} = (S, M, \Delta)$ and two distinct control states $s_{init}, s_{fin} \in S$, whether there is a finite computation of $\mathcal{C}$ starting in control state (s_{fin}, x) and ending in state (s_{init}, ε), for some initial channel contents $x \in M^*$.

Proposition 25: The dual control-state reachability problem for channel machines with insertion errors has non-primitive recursive complexity.

Proof: Given a channel machine $\mathcal{C} = (S, M, \Delta)$, the *opposite channel machine* is defined by $\mathcal{C}^{\mathrm{op}} = (S, M, \Delta^{\mathrm{op}})$

where

$$\Delta^{\mathrm{op}} = \{(s, m!, t) : (t, m?, s) \in \Delta\} \cup \{(s, m?, t) : (t, m!, s) \in \Delta\}.$$

Note that $\mathcal{C}$ has a computation from (s, x) to (t, y) with lossiness errors iff $\mathcal{C}^{\mathrm{op}}$ has a computation from (t, y^{op}) to (s, x^{op}) with insertion errors, where $(-)^{\mathrm{op}} : M^* \to M^*$ reverses the order of a word. Thus the dual control-state reachability problem for $\mathcal{C}$ with insertion errors is clearly equivalent to the control-state reachability problem for $\mathcal{C}^{\mathrm{op}}$ with lossiness errors. But, as we mentioned above, this last problem is known to be decidable with non-primitive recursive complexity. ∎

Theorem 26: The satisfiability and model checking problems for MTL over finite words have non-primitive recursive complexity.

Proof: We give a reduction of the dual control-state reachability problem for channel machines with insertion errors to the satisfiability problem for MTL[8]. For this reduction it is helpful to introduce the *non-strict henceforth* operator $\boxdot\varphi \equiv \varphi \wedge \Box\varphi$.

Let $\mathcal{C} = (S, M, \Delta)$ and $s_{init}, s_{fin} \in S$ be an instance of the dual control-state reachability problem. We consider MTL formulas over the alphabet $\Sigma = S \cup \{m!, m? : m \in M\}$. We use the formula φ_{CHAN} below to capture the channel discipline: every write-event is followed one time unit later by a matching read-event. However, there is no guarantee that every read-event is *preceded* one time unit earlier by a write-event, so the channel may have insertion errors.

$$\varphi_{CHAN} \equiv \textstyle\bigwedge_{m \in M} \boxdot(m! \to \Diamond_{=1} m?).$$

In order that there be no confusion in terms of matching write-events with their corresponding subsequent read-events, we require that time be strongly monotonic (no two events can occur at the same time):

$$\varphi_{SM} \equiv \boxdot\, \Box_{=0} \perp.$$

Note that this formula achieves its aim thanks to our strict-future semantics for $\Box$ which places no obligations on the current state but only on subsequent ones.

We encode the finite control of $\mathcal{C}$ using the formula φ_{CONT}:

$$\varphi_{CONT} \equiv \bigwedge_{s \in S} (s \to \bigvee_{(s,\mu,t) \in \Delta} (\bigcirc\mu \wedge \bigcirc\bigcirc t)).$$

We then use φ_{RUN} to assert that a run must start in control state s_{fin} and obey the discrete controller until it terminates, in control state s_{init}:

$$\varphi_{RUN} \equiv s_{fin} \wedge \varphi_{CONT} \wedge (\varphi_{CONT}\ \mathcal{U}\ (s_{init} \wedge \Box\perp)).$$

We combine all these requirements into φ_{REACH}:

$$\varphi_{REACH} \equiv \varphi_{CHAN} \wedge \varphi_{SM} \wedge \varphi_{RUN}.$$

[8]Recall that we are using the strict-future semantics, though it is straightforward to modify the proof to accommodate the non-strict semantics.

Suppose we are given a timed word ρ satisfying φ_{REACH}; then we can construct a computation of $\mathcal{C}$ as follows. Note that ρ consists of an alternating sequence of events from S and events from $\{m!, m? : m \in M\}$. This gives the sequence of control states and transitions in the desired computation; it remains to construct the contents of the channel at each control state. Suppose event $s \in S$ occurs at some point along ρ with timestamp t. Then the channel contents associated to this occurrence of s is the sequence of read-events occurring in ρ in the time interval $(t, t+1)$. Observe how this definition ensures that a message can only be read from the head of the channel, and how each write-event adds a message to the tail of the channel. Note also that in this reconstruction of a computation of $\mathcal{C}$, all insertion errors add messages to the tail of the channel. Finally, observe that any timed word satisfying φ_{REACH} must have s_{init} as its last event; this ensures that the channel is empty at that point.

Conversely, suppose we are given a computation of $\mathcal{C}$,

$$(s_0, x_0) \xrightarrow{\alpha_0} (s_1, x_1) \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_{n-1}} (s_n, x_n)$$

with $s_0 = s_{fin}$, $s_n = s_{init}$ and $x_n = \varepsilon$. We then derive a timed word $\rho = (\sigma, \tau)$ that satisfies φ_{REACH}. We define $\sigma = s_0\alpha_0 s_1\alpha_1 \ldots s_n$; this guarantees that ρ satisfies φ_{CONT}. It remains to choose a sequence of timestamps τ such that $\varphi_{CHAN} \wedge \varphi_{SM}$ is also satisfied.

Since the given computation of $\mathcal{C}$ ends with the empty channel, every message that is written to the channel is eventually read from the channel. Thus for each write event $m!$ in σ there is a 'matching' read event $m?$ later on. We choose the timestamps τ so that each such matching pair is separated by one time unit. Formally we choose the τ_i sequentially, starting with $\tau_0 = 0$ and maintaining the following invariant: τ_i is chosen such that for each matching pair $\sigma_j = m!$ and $\sigma_k = m?$, if $j < k = i$ then $\tau_i - \tau_j = 1$, and if $j < i < k$ then $\tau_i - \tau_j < 1$. It is clearly possible to do this using the density of time.

Thus a channel machine $\mathcal{C} = (S, M, \Delta)$ and pair of control states $s_{init}, s_{fin} \in S$ is a positive instance of the dual reachability problem iff the formula φ_{REACH} is satisfiable.

Finally, note that the model checking and satisfiability problems for MTL are equivalent, since MTL is closed under negation. ∎

6. Infinite Words: Safety MTL

In this section we use the techniques developed in Section 3 to prove the decidability of the model-checking problem over infinite words for a subset of MTL, called *Safety MTL.* Safety MTL consists of those MTL formulas in positive normal form that only include instances of the constrained until operator $\mathcal{U}_I$ in which interval I has bounded length. Note that no restrictions are placed on the dual-until operator $\widetilde{\mathcal{U}}_I$.

Safety MTL can express time-bounded response properties, but not arbitrary response formulas. For instance, the formulas $\varphi_1 \equiv \Box(a \to \Diamond_{=1} b)$ and $\varphi_2 \equiv \Box(a \to \Diamond_{\leq 5}(b \wedge \Diamond_{=1} c))$

are in Safety MTL. Note in passing that, intuitively, there is a qualitative difference in the difficulty of model checking φ_1 compared with model checking φ_2. To find a counterexample to φ_1 one need only guess an a-event, and check that there is no b-event one time unit later—a task requiring only one clock. On the other hand, to find a counterexample to φ_2 one must not only guess an a-event, but also check that every b-event in the ensuing five time units fails to have a matching c event one time unit later—a task requiring a potentially unbounded number of clocks.

To explain the name Safety MTL, recall from [14] that $L \subseteq T\Sigma^\omega$ is a *safety property relative to the divergence of time* if for every $\rho \notin L$ there exists $n \in \mathbb{N}$ such that no infinite timed word in $T\Sigma^\omega$ extending $\rho[0 \dots n-1]$ is contained in L. In this case we say that $\rho[0 \dots n-1]$ is a *bad prefix* of ρ.

Proposition 27: For every Safety MTL formula φ, $L_\omega(\varphi)$ is a safety property relative to the divergence of time.

Proof: This is a straightforward structural induction on φ. ■

To model check φ on an Alur-Dill timed automaton $\mathcal{B}$ we need only check whether any of the bad prefixes of φ are prefixes of words accepted by $\mathcal{B}$. We can do this using a variant of the idea used in the proof of Theorem 20. To set up this decision prodecure we first define a translation of φ into a one-clock alternating automaton $\mathcal{A}_\varphi^{\mathrm{safe}}$ in which every location is accepting.

We define $\mathcal{A}_\varphi^{\mathrm{safe}}$ as a modification of the automaton $\mathcal{A}_\varphi$ from Subsection 5.1. $\mathcal{A}_\varphi^{\mathrm{safe}}$ has the same alphabet, locations and initial location as $\mathcal{A}_\varphi$, but we declare every location of $\mathcal{A}_\varphi^{\mathrm{safe}}$ to be accepting. To compensate for this last change, we slightly modify the definition of the transition function δ—the clause for $\varphi_1 \ \mathcal{U}_I \ \varphi_2$ is altered as we indicate below. Here we use the notation $x \le I$ to mean $x < k$ if I is right-open with right endpoint k, and $x \le k$ if I is right-closed with right endpoint k.

$$\begin{aligned}\delta(\psi_1 \, \mathcal{U}_I \, \psi_2, a) = {} & (\mathit{init}(\psi_2, a) \wedge x \in I) \vee \\ & (\mathit{init}(\psi_1, a) \wedge x \le I \wedge (\psi_1 \, \mathcal{U}_I \, \psi_2)).\end{aligned}$$

Thus we use a timeout rather than an acceptance condition to ensure that the second argument of $\mathcal{U}_I$ eventually becomes true. This strategy works because of the assumption that time diverges in infinite timed words.

Note that so far we have only considered alternating automata on finite words. In order to state the correctness of the definition of $\mathcal{A}_\varphi^{\mathrm{safe}}$ we first define a run of $\mathcal{A}_\varphi^{\mathrm{safe}}$ on an *infinite* timed word $\rho = (\sigma, \tau)$ to be an infinite alternating sequence of edge steps and flow steps in $\mathcal{T}_{\mathcal{A}_\varphi^{\mathrm{safe}}}$:

$$C_0 \xrightarrow{\sigma_0} C_1 \overset{d_0}{\rightsquigarrow} C_2 \xrightarrow{\sigma_1} C_3 \overset{d_1}{\rightsquigarrow} \cdots \overset{d_{n-1}}{\rightsquigarrow} C_{2n} \xrightarrow{\sigma_n} \cdots ,$$

where C_0 is the initial configuration and $d_i = \tau_{i+1} - \tau_i$. Next we define $L_\omega(\mathcal{A}_\varphi^{\mathrm{safe}})$, the *infinitary language of* φ, to be the set of $\rho \in T\Sigma^\omega$ over which $\mathcal{A}_\varphi^{\mathrm{safe}}$ has a run. (Since every state of $\mathcal{A}_\varphi^{\mathrm{safe}}$ is accepting there is no need to consider an acceptance condition here.)

Proposition 28: $L_\omega(\varphi) = L_\omega(\mathcal{A}_\varphi^{\mathrm{safe}})$ for each Safety MTL formula φ.

Proof: The proof of Proposition 28 is a straightforward modification of the proof of Proposition 23. ■

As a corollary we get that the complement automaton $(\mathcal{A}_\varphi^{\mathrm{safe}})^c$ (as defined in Subsection 3.1) accepts a bad prefix of each timed word that does not satisfy φ. Note that this automaton does not have any accepting locations—it accepts by moving to an empty configuration.

Corollary 29: Let φ be a Safety MTL formula. For all $\rho \in T\Sigma^\omega$, $\rho \not\models \varphi$ iff $\exists n \in \mathbb{N}$ such that $\rho[0 \dots n-1] \in L_f((\mathcal{A}_\varphi^{\mathrm{safe}})^c)$.

Proof: We sketch the only-if direction. If $\rho \not\models \varphi$ then, by Proposition 28, $\mathcal{A}_\varphi^{\mathrm{safe}}$ does not have a run on ρ. By König's lemma there exists $n \in \mathbb{N}$ such that $\mathcal{A}_\varphi^{\mathrm{safe}}$ does not have a run on the finite word $\rho[0 \dots n-1]$. Thus the complement automaton $(\mathcal{A}_\varphi^{\mathrm{safe}})^c$ accepts $\rho[0 \dots n-1]$. ■

Given an Alur-Dill timed automaton $\mathcal{B}$, we now sketch a decision procedure for the model-checking problem '$L_\omega(\mathcal{B}) \subseteq L_\omega(\varphi)$?'. This uses the same techniques as the proof in Section 3 to the effect that emptiness is decidable for one-clock alternating automata. We assume in the discussion below that the reader is familiar with the operational semantics of Alur-Dill timed automata.

Theorem 30: The model-checking problem for Safety MTL over infinite words is decidable: given a timed automaton $\mathcal{B}$ and a Safety MTL formula φ, there is an algorithm to decide whether or not $L_\omega(\mathcal{B}) \subseteq L_\omega(\varphi)$.

Proof: Suppose that $\mathcal{B}$ has n clocks. A state of $\mathcal{B}$ is a pair $\gamma = (s, \mathbf{v})$, where s is a location of $\mathcal{B}$ and $\mathbf{v} \in (\mathbb{R}_{\ge 0})^n$ is a vector of values for the clocks of $\mathcal{B}$. Following the pattern of Definition 15 we define a labelled transition system $\mathcal{T}_{\mathcal{B},\varphi}$ representing $\mathcal{B}$ and $(\mathcal{A}_\varphi^{\mathrm{safe}})^c$ executing in parallel. The states of $\mathcal{T}_{\mathcal{B},\varphi}$ are pairs (γ, C), where γ is a state of $\mathcal{B}$ and C is a configuration of $(\mathcal{A}_\varphi^{\mathrm{safe}})^c$. As in Definition 15 we define an $(\mathbb{R}_{\ge 0})$-labelled flow-step transition relation by $(\gamma, C) \overset{t}{\rightsquigarrow} (\gamma + t, C + t)$ for $t \ge 0$, and a Σ-labelled edge-step transition relation by $(\gamma, C) \xrightarrow{a} (\gamma', C')$ if $\gamma \xrightarrow{a} \gamma'$ and $C \xrightarrow{a} C'$ for some $a \in \Sigma$. A state $((s, \mathbf{0}), C)$ of $\mathcal{T}_{\mathcal{B},\varphi}$ is said to *initial* if s is an initial location of $\mathcal{B}$ and C is the initial configuration of $(\mathcal{A}_\varphi^{\mathrm{safe}})^c$.

Since $(\mathcal{A}_\varphi^{\mathrm{safe}})^c$ can only accept by moving to the empty configuration, a timed word $\rho \in L_\omega(\mathcal{B})$ fails to satisfy φ iff there is a computation of $(\mathcal{A}_\varphi^{\mathrm{safe}})^c$ on a finite prefix of ρ that reaches $\emptyset$. Motivated by this observation, we say that a state (γ, C) of $\mathcal{T}_{\mathcal{B},\varphi}$ is *doomed* if $C = \emptyset$, i.e., $(\mathcal{A}_\varphi^{\mathrm{safe}})^c$ has reached an accepting configuration, and if $\mathcal{B}$ can accept some infinite time-divergent word starting in state γ. Then $L_\omega(\mathcal{B}) \not\subseteq L_\omega(\varphi)$ iff there is a doomed state $(\gamma, \emptyset)$ that is reachable from the initial state of $\mathcal{T}_{\mathcal{B},\varphi}$. Below we sketch how we can use Proposition 18 to prove that this reachability problem is decidable.

To set up the application of Proposition 18 we can reuse constructions from Section 3 to show that $\mathcal{T}_{\mathcal{B},\varphi}$ admits a quotient transition system $\mathcal{W}_{\mathcal{B},\varphi}$ that is a WSTS. Suppose that $\mathcal{B}$ has set of locations S and $(\mathcal{A}_\varphi^{\mathrm{safe}})^c$ has set of locations T, where S and T are disjoint. Define a finite alphabet $\Lambda =$

$\wp(((S \times \{1, \ldots, n\}) \cup T) \times REG_k)$, where k is the maximum constant mentioned in the clock constraints of $\mathcal{B}$ and $(\mathcal{A}_\varphi^{\text{safe}})^c$. The set Λ^* of finite words over Λ is the state space of $\mathcal{W}_{\mathcal{B},\varphi}$. We reuse the function H from Section 3 to encode states of $\mathcal{T}_{\mathcal{B},\varphi}$ as states of $\mathcal{W}_{\mathcal{B},\varphi}$. Encode a state $((s, \mathbf{v}), C)$ of $\mathcal{T}_{\mathcal{B},\varphi}$ as a word $H(\{((s, 1), v_1), \ldots, ((s, n), v_n)\} \cup C)$. From this word we can reconstruct all clock values in $((s, \mathbf{v}), C)$ up to the nearest integer and also the relative order of the fractional parts of the clocks. Similarly to the Bisimulation Lemma, Lemma 12, we use this observation to prove that the kernel of H is a time-abstract bisimulation on $\mathcal{T}_{\mathcal{B},\varphi}$, and thus induces a quotient transition structure on $\mathcal{W}_{\mathcal{B},\varphi}$.

We define a word $\mathcal{W}_{\mathcal{B},\varphi}$ to be initial (respectively doomed) if it is the image under H of an initial (respectively doomed) state of $\mathcal{T}_{\mathcal{B},\varphi}$. Since $\mathcal{W}_{\mathcal{B},\varphi}$ is a quotient of $\mathcal{T}_{\mathcal{B},\varphi}$, the inclusion $L_\omega(\mathcal{B}) \subseteq L_\omega(\varphi)$ holds iff it is not possible to reach a doomed state from the initial state in $\mathcal{W}_{\mathcal{B},\varphi}$. Finally, since the set of doomed states in $\mathcal{W}_{\mathcal{B},\varphi}$ is trivially downward-closed with respect to the monotone domination order, and since it is decidable whether a state is doomed or not (thanks to [4]), Proposition 18 gives us a decision procedure for the language inclusion question '$L_\omega(\mathcal{B}) \subseteq L_\omega(\varphi)$?'. ■

7. Conclusion

In this paper, we have shown that Metric Temporal Logic is decidable over finite timed words in its standard dense-time, point-based semantics, with non-primitive recursive complexity. Over infinite words, we have shown that the important safety fragment of Metric Temporal Logic can be model checked, although we do not know the complexity of this problem.

As future work, we would like to settle the question of the decidability of the full logic MTL over infinite words. In this context it is natural to consider the language-emptiness problem for one-clock timed alternating automata over infinite words. Indeed Proposition 23, which asserts the correctness of the translation from MTL formulas to automata, extends to infinite words provided that we endow the automata with weak parity acceptance conditions.

We have recently shown that the universality problem for one-clock Alur-Dill timed automata with Büchi acceptance conditions is undecidable[9] [1]. It follows that the language-emptiness problem for one-clock timed alternating automata with co-Büchi acceptance conditions is undecidable. However these appear to be more general than weak parity acceptance conditions, so that the problem remains open.

Acknowledgements. We thank Tom Henzinger for clarifying some of the undecidability results for MTL, and for asking about the relationship between single-clock timed automata and real-time temporal logics. We also thank the anonymous referees for their careful reading and helpful suggestions.

[9]This result was discovered concurrently by Lasota and Walukiewicz [22].

References

[1] P. A. Abdulla, J. Deneux, J. Ouaknine and J. Worrell. Decidability and complexity results for timed automata via channel systems. In *Proceedings of ICALP 05*, LNCS, to appear, 2005.

[2] P. A. Abdulla and B. Jonsson. Undecidable verification problems with unreliable channels. *Information and Computation*, 130:71–90, 1996.

[3] P. A. Abdulla, B. Jonsson. Model checking of systems with many identical timed processes. *Theoretical Computer Science*. 290(1):241–264, 2003.

[4] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[5] R. Alur, T. Feder and T. A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43:116–146, 1996.

[6] R. Alur and T. A. Henzinger. Logics and models of real time: A survey. In *Proceedings of Real Time: Theory in Practice*, LNCS 600, 1992.

[7] R. Alur and T. A. Henzinger. Real-time logics: complexity and expressiveness. *Information and Computation*, 104:35–77, 1993.

[8] R. Alur and T. A. Henzinger. A really temporal logic. *Journal of the ACM*, 41:181–204, 1994.

[9] J. A. Brzozowski and E. Leiss. Finite automata and sequential networks. *Theoretical Computer Science*, 10:19–35, 1980.

[10] G. Cécé, A. Finkel and S. Purushothaman Iyer. Unreliable channels are easier to verify than perfect channels. *Information and Computation*, 124:20–31, 1996.

[11] A. K. Chandra, D. C. Kozen and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[12] A. Finkel and P. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1-2):63–92, 2001.

[13] T. A. Henzinger. The temporal specification and verification of real-time systems. *Ph.D. Thesis*, Technical Report STAN-CS-91-1380, Stanford University, 1991.

[14] T. A. Henzinger. Sooner is safer than later. *Information Processing Letters*, 43:135–141, 1992.

[15] T. A. Henzinger. It's about time: Real-time logics reviewed. In *Proceedings of CONCUR 98*, LNCS 1466, 1998.

[16] T. A. Henzinger, Z. Manna and A. Pnueli. What good are digital clocks? In *Proceedings of ICALP 92*, LNCS 623, 1992.

[17] T. A. Henzinger, J.-F. Raskin, and P.-Y. Shobbens. The regular real-time languages. In *Proceedings of ICALP 98*, LNCS 1443, 1998.

[18] P. Hermann. Timed automata and recognizability. *Information Processing Letters*, 65:313–318, 1998.

[19] G. Higman. Ordering by divisibility in abstract algebras. *Proc. of the London Mathematical Society*, 2:236–366, 1952.

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

[21] S. Lasota and I. Walukiewicz. Alternating timed automata. In *Proceedings of FOSSACS 05*, LNCS 3441, 2005.

[22] S. Lasota and I. Walukiewicz. Personal communication, 2005.

[23] J. Ostroff. Temporal logic of real-time systems. Research Studies Press, Taunton, England.

[24] J. Ouaknine and J. Worrell. On the language inclusion problem for timed automata: Closing a decidability gap. In *Proceedings of LICS 04*, IEEE Computer Society Press, 2004.

[25] J.-F. Raskin and P.-Y. Schobbens. State-clock logic: a decidable real-time logic. In *Proceedings of HART 97*, LNCS 1201, 1997.

[26] P. Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity. *Information Processing Letters*, 83(5):251–261, 2002.

[27] M. Vardi. Alternating automata: Unifying truth and validity checking for temporal logics. In *Proceedings of CADE 97*, LNCS 1249, 1997.

[28] T. Wilke. Specifying timed state sequences in powerful decidable logics and timed automata. *Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 863, 1994.