Safety Metric Temporal Logic is Fully Decidable

Joël Ouaknine and James Worrell

Oxford University Computing Laboratory, UK
{joel,jbw}@comlab.ox.ac.uk

Abstract. Metric Temporal Logic (MTL) is a widely-studied real-time extension of Linear Temporal Logic. In this paper we consider a fragment of MTL, called Safety MTL, capable of expressing properties such as invariance and time-bounded response. Our main result is that the satisfiability problem for Safety MTL is decidable. This is the first positive decidability result for MTL over timed ω-words that does not involve restricting the precision of the timing constraints, or the granularity of the semantics; the proof heavily uses the techniques of infinite-state verification. Combining this result with some of our previous work, we conclude that Safety MTL is *fully decidable* in that its satisfiability, model checking, and refinement problems are all decidable.

1 Introduction

Timed automata and real-time temporal logics provide the foundation for several well-known and mature tools for verifying timed and hybrid systems [21]. Despite this success in practice, certain aspects of the real-time theory are notably less well-behaved than in the untimed case. In particular, timed automata are not determinisable, and their language inclusion problem is undecidable [4]. In similar fashion, the model-checking problems for (linear-time) real-time logics such as *Metric Temporal Logic* and *Timed Propositional Temporal Logic* are also undecidable [5, 6, 17].

For this reason, much interest has focused on *fully decidable* real-time specification formalisms. We explain this term in the present context as follows. We represent a computation of a real-time system as a *timed word*: a sequence of instantaneous events, together with their associated timestamps. A specification denotes a *timed language*: a set of allowable timed words. Then a formalism (a logic or class of automata) is fully decidable if it defines a class of timed languages that is closed under finite unions and intersections and has a decidable language-inclusion problem[1]. Note that language emptiness and universality are special cases of language inclusion.

[1] This phrase was coined in [12] with a slightly more general meaning: a specification formalism closed under finite unions, finite intersections and complementation, and for which language emptiness is decidable. However, since the main use of complementation in this context is in deciding language inclusion, we feel that our definition is in the same spirit.

In this paper we are concerned in particular with Metric Temporal Logic (MTL), one of the most widely known real-time logics. MTL is a variant of Linear Temporal Logic in which the temporal operators are replaced by time-constrained versions. For example, the formula $\Box_{[0,5]}\varphi$ expresses that φ holds for the next 5 time units. Until recently, the only positive decidability results for MTL involved placing syntactic restrictions on the precision of the timing constraints, or restricting the granularity of the semantics. For example, [5, 12, 19] ban punctual timing constraints, such as $\Diamond_{=1}\varphi$ (φ is true in exactly one time unit). Semantic restrictions include adopting an integer-time model, as in [6, 11], or a bounded-variation dense-time model, as in [22]. These restrictions guarantee that a formula has a finite tableau: in fact they yield decision procedures for model checking and satisfiability that use exponential space in the size of the formula. However, both the satisfiability and model checking problems are undecidable in the unrestricted logic, cf. [5, 17].

The main contribution of this paper is to identify a new fully decidable fragment of MTL, called *Safety MTL*. Safety MTL consists of those MTL formulas which, when expressed in negation normal form, are such that the interval I is bounded in every instance of the constrained until operator $\mathcal{U}_I$ and the constrained eventually operator $\Diamond_I$. For example, the time-bounded response formula $\Box(a \to \Diamond_{=1} b)$ (every a-event is followed after one time unit by a b-event) is in Safety MTL, but not $\Box(a \to \Diamond_{(1,\infty)} b)$. Because we place no limit on the precision of the timing constraints or the granularity of the semantics, the tableau of a Safety MTL formula may have infinitely many states. However, using techniques from infinite-state verification, we show that the restriction to safety properties facilitates an effective analysis.

In [16] we already gave a procedure for model checking Alur-Dill timed automata against Safety MTL formulas. As a special case we obtained the decidability of the *validity* problem for Safety MTL ('Is a given formula satisfied by every timed word?'). The two main contributions of the present paper complement this result, and show that Safety MTL is fully decidable. We show the decidability of the *satisfiability problem* ('Is a given Safety MTL formula satisfied by some timed word?') and, more generally, we claim decidability of the *refinement problem* ('Given two Safety MTL formulas φ_1 and φ_2, does every timed word that satisfies φ_1 also satisfy φ_2?'). Note that Safety MTL is not closed under negation, so neither of these results follow trivially from the decidability of validity.

Closely related to MTL are *timed alternating automata*, introduced in [15, 16]. Both cited works show that the language-emptiness problem for one-clock timed alternating automata over finite timed words is decidable. This result is the foundation of the above-mentioned model-checking procedure for Safety MTL. The procedure involves translating the negation of a Safety MTL formula φ into a one-clock timed alternating automaton over finite words that accepts all the *bad prefixes* of φ. (Every infinite timed word that fails to satisfy a Safety MTL formula φ has a *finite bad prefix*, that is, a finite prefix none of whose extensions satisfies φ.) In contrast, the results in the present paper involve considering timed alternating automata over infinite timed words.

Our main technical contribution is to show the decidability of language-emptiness over infinite timed words for a class of timed alternating automata rich enough to capture Safety MTL formulas. A key restriction is that we only consider automata in which every state is accepting. We have recently shown that language emptiness is undecidable for one-clock alternating automata with Büchi or even weak parity acceptance conditions [17]. Thus the restriction to safety properties is crucial.

As in [16], we make use of the notion of a *well-structured transition system (WSTS)* [9] to give our decision procedure. However, whereas the algorithm in [16] involved reduction to a reachability problem on a WSTS, here we reduce to a fair nontermination problem on a WSTS. The fairness requirement is connected to the assumption that timed words are non-Zeno. Indeed, we remark that our results provide a rare example of a decidable nontermination problem on an infinite-state system with a nontrivial fairness condition. For comparison, undecidability results for nontermination under various different fairness conditions for Lossy Channel Systems, Timed Networks, and Timed Petri Nets can be found in [2, 3].

Related Work. An important distinction among real-time models is whether one records the *state* of the system of interest at every instant in time, leading to an *interval semantics* [5, 12, 19], or whether one only sees a countable sequence of instantaneous *events*, leading to a *point-based* or *trace semantics* [4, 6, 7, 10, 11, 22]. In the interval semantics the temporal operators of MTL quantify over the whole time domain, whereas in the point-based semantics they quantify over a countable set of positions in a timed word. For this reason the interval semantics is more natural for reasoning about states, whereas the point-based semantics is more natural for reasoning about events. In this paper we adopt the latter.

MTL and Safety MTL do not differ in terms of their decidability in the interval semantics: Alur, Feder, and Henzinger [5] showed that the satisfiability problem for MTL is undecidable, and it is easy to see that their proof directly carries over to Safety MTL. We pointed out in [16] that the same proof does not apply in the point-based semantics, and we recently gave a different argument to show that MTL is undecidable in this setting. However, our proof crucially uses a 'liveness formula' of the form $\Box\Diamond p$, and it does not apply to Safety MTL. The results in this paper confirm that by excising such formulas we obtain a fully decidable logic in the point-based setting.

2 Metric Temporal Logic

In this section we define the syntax and semantics of Metric Temporal Logic (MTL). As discussed above, we adopt a point-based semantics over timed words.

A *time sequence* $\tau = \tau_0\tau_1\tau_2\ldots$ is an infinite nondecreasing sequence of time values $\tau_i \in \mathbb{R}_{\geq 0}$. Here it is helpful to adopt the convention that $\tau_{-1} = 0$. If $\{\tau_i : i \in \mathbb{N}\}$ is bounded then we say that τ is *Zeno*, otherwise we say that

τ is *non-Zeno*. A *timed word* over finite alphabet Σ is a pair $\rho = (\sigma, \tau)$, where $\sigma = \sigma_0\sigma_1\ldots$ is an infinite word over Σ and τ is a time sequence. We also represent a timed word as a sequence of *timed events* by writing $\rho = (\sigma_0, \tau_0)(\sigma_1, \tau_1)\ldots$. Finally, we write $T\Sigma^\omega$ for the set of non-Zeno timed words over Σ.

Definition 1. *Given an alphabet Σ of atomic events, the formulas of MTL are built up from Σ by monotone Boolean connectives and time-constrained versions of the* ***next*** *operator $\bigcirc$,* ***until*** *operator $\mathcal{U}$ and the* ***dual until*** *operator $\widetilde{\mathcal{U}}$ as follows:*

$$\varphi ::= \top \mid \bot \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid a \mid \bigcirc_I \varphi \mid \varphi_1 \,\mathcal{U}_I\, \varphi_2 \mid \varphi_1 \,\widetilde{\mathcal{U}}_I\, \varphi_2$$

where $a \in \Sigma$, and $I \subseteq \mathbb{R}_{\geq 0}$ is an open, closed, or half-open interval with end-points in $\mathbb{N} \cup \{\infty\}$.

Safety MTL is the fragment of MTL obtained by requiring that the interval I in each 'until' operator $\mathcal{U}_I$ have finite length. (Note that no restriction is placed on the dual until operators $\widetilde{\mathcal{U}}_I$ or next operators $\bigcirc_I$.)

Additional temporal operators are defined using the usual conventions. We have the *constrained eventually* operator $\Diamond_I \varphi \equiv \top \,\mathcal{U}_I\, \varphi$, and the *constrained always* operator $\Box_I\, \varphi \equiv \bot \,\widetilde{\mathcal{U}}_I\, \varphi$. We use pseudo-arithmetic expressions to denote intervals. For example, the expression '$= 1$' denotes the interval $[1, 1]$. In case $I = [0, \infty)$ we simply omit the annotation I on temporal operators. Finally, given $a \in \Sigma$, we write $\neg a$ for $\bigvee_{b \in \Sigma \setminus \{a\}} b$.

Definition 2. *Given a 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 as follows:*

- $(\rho, i) \models a$ *iff* $\sigma_i = a$
- $(\rho, i) \models \varphi_1 \wedge \varphi_2$ *iff* $(\rho, i) \models \varphi_1$ *and* $(\rho, i) \models \varphi_2$
- $(\rho, i) \models \varphi_1 \vee \varphi_2$ *iff* $(\rho, i) \models \varphi_1$ *or* $(\rho, i) \models \varphi_2$
- $(\rho, i) \models \bigcirc_I\, \varphi$ *iff* $\tau_{i+1} - \tau_i \in I$ *and* $(\rho, i+1) \models \varphi$
- $(\rho, i) \models \varphi_1 \,\mathcal{U}_I\, \varphi_2$ *iff there exists* $j \geqslant i$ *such that* $(\rho, j) \models \varphi_2$, $\tau_j - \tau_i \in I$, *and* $(\rho, k) \models \varphi_1$ *for all* k *with* $i \leqslant k < j$.
- $(\rho, i) \models \varphi_1 \,\widetilde{\mathcal{U}}_I\, \varphi_2$ *iff for all* $j \geqslant i$ *such that* $\tau_j - \tau_i \in I$, *either* $(\rho, j) \models \varphi_2$ *or there exists* k *with* $i \leqslant k < j$ *and* $(\rho, k) \models \varphi_1$.

We say that ρ satisfies φ, denoted $\rho \models \varphi$, if $(\rho, 0) \models \varphi$. The ***language*** *of φ is the set $L(\varphi) = \{\rho \in T\Sigma^\omega : \rho \models \varphi\}$ of non-Zeno words that satisfy φ.*

Example 1. Consider an alphabet $\Sigma = \{req_i, aq_i, rel_i : i = X, Y\}$ denoting the actions of two processes X and Y that request, acquire, and release a lock. The following formulas are all in Safety MTL.

- $\Box(aq_X \rightarrow \Box_{<3} \neg aq_Y)$ says that Y cannot acquire the lock less than 3 seconds after X acquires the lock.
- $\Box(aq_X \rightarrow rel_X \,\widetilde{\mathcal{U}}_{<3}\, \neg aq_Y)$ says that Y cannot acquire the lock less than 3 seconds after X acquires the lock, unless X first releases it.
- $\Box(req_X \rightarrow \Diamond_{<2}(aq_X \wedge \Diamond_{=1} rel_X))$ says that whenever X requests the lock, it acquires the lock within 2 seconds and releases it exactly one second later.

3 Timed Alternating Automata

In this paper, following [15, 16], a timed alternating automaton is an alternating automaton augmented with a single clock variable[2].

We use x to denote the single clock variable of an automaton. A *clock constraint* is a term of the form $x \bowtie c$, where $c \in \mathbb{N}$ and $\bowtie \in \{<, \leqslant, \geqslant, >\}$. Given a set S of *locations*, $\Phi(S)$ denotes the set of formulas generated from S and the set of clock constraints by positive Boolean connectives and variable binding. Thus $\Phi(S)$ is generated by the grammar

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

where $s \in S$ and $x.\varphi$ binds x to 0 in φ.

In the definition of a timed alternating automaton, below, the transition function δ 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. Variable binding corresponds to the automaton resetting x to 0. For example, $\delta(s, a) = (x < 1) \wedge s \wedge x.t$ means that when the automaton is in location s with clock value less than 1, it can make a simultaneous a-labelled transition to locations s and t, resetting the clock as it enters t.

Definition 3. *A timed alternating automaton is a tuple $\mathcal{A} = (\Sigma, S, s_0, \delta)$, where*

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

We consider all locations of $\mathcal{A}$ to be accepting.

The following example illustrates how a timed alternating automaton accepts a language of timed words.

Example 2. We define an automaton $\mathcal{A}$ over the alphabet $\Sigma = \{a, b\}$ that accepts all those timed words in which every a-event is followed one time unit later by a b-event. $\mathcal{A}$ has three locations $\{s, t, u\}$, with s the initial location. The transition function δ is given by the following table:

	a	b
s	$s \wedge x.t$	s
t	$t \wedge (x \leqslant 1)$	$(t \wedge (x < 1)) \vee (u \wedge (x = 1))$
u	u	u

[2] Virtually all decision problems, and in particular language emptiness, are undecidable for alternating automata with more than one clock.

A run of $\mathcal{A}$ starts in location s. Every time an a-event occurs, the automaton makes a simultaneous transition to both s and t, thus opening up a new thread of computation. The automaton resets a fresh copy of clock x when it moves from location s to t, and in location t it only performs transitions as long as the clock does not exceed one. Therefore if location t is entered at some point in a non-Zeno run, it must eventually be exited. Inspecting the transition table, we see that the only way for this to happen is if a b-event occurs exactly one time unit after the a-event that spawned the t-state.

Next we proceed to the formal definition of a run.

Define a *tree* to be a directed acyclic graph (V, E) with a distinguished *root node* such that every node is reachable by a unique finite path from the root. It is clear that every tree admits a stratification, $\mathit{level} : V \to \mathbb{N}$, such that $v\ E\ v'$ implies $\mathit{level}(v') = \mathit{level}(v) + 1$ and the root has level 0.

Let $\mathcal{A} = (\Sigma, S, s_0, \delta)$ be an automaton. A *state* of $\mathcal{A}$ is a pair (s, ν), where $s \in S$ is a location and $\nu \in \mathbb{R}_{\geq 0}$ is the clock value. Write $Q = S \times \mathbb{R}_{\geq 0}$ for the set of all states. A finite set of states is a *configuration*. Given a clock value ν, we define a satisfaction relation $\models_\nu$ between configurations and formulas in $\Phi(S)$ according to the intuition that state (s, ν) can make an a-transition to configuration C if $C \models_\nu \delta(s, a)$. The definition of $C \models_\nu \varphi$ is given by induction on $\varphi \in \Phi(S)$ as follows: $C \models_\nu t$ if $(t, \nu) \in C$, $C \models_\nu x \bowtie c$ if $\nu \bowtie c$, $C \models_\nu x.\varphi$ if $C \models_0 \varphi$, and we handle the Boolean connectives in $\Phi(S)$ in the obvious way.

Definition 4. *A* ***run*** Δ *of* $\mathcal{A}$ *on a timed word* (σ, τ) *consists of a tree* (V, E) *and a labelling function* $l : V \to Q$ *such that if* $l(v) = (s, \nu)$ *for some level-*n *node* $v \in V$*, then* $\{l(v') \mid v\ E\ v'\} \models_{\nu'} \delta(s, \sigma_n)$*, where* $\nu' = \nu + (\tau_n - \tau_{n-1})$*.*

The ***language*** *of* $\mathcal{A}$*, denoted* $L(\mathcal{A})$*, consists of all non-Zeno words over which* $\mathcal{A}$ *has a run whose root is labelled* $(s_0, 0)$*.*

Figure 1 depicts part of a run of the automaton $\mathcal{A}$ from Example 2 on the timed word $\langle (a, 0.3), (b, 0.5), (a, 0.8), (b, 1.3), (b, 1.8) \ldots \rangle$.

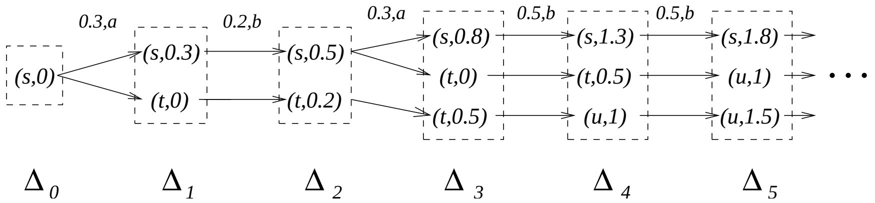

Fig. 1. Consecutive levels in a run of $\mathcal{A}$.

One typically applies the acceptance condition in an alternating automaton to all paths in the run tree [20]. In the present context, since every location is accepting, the tree structure plays no role in the definition of acceptance; in this respect, a run could be viewed simply as a sequence of configurations. This motivates the following definition.

Definition 5. *Given a run $\Delta = ((V, E), l)$ of $\mathcal{A}$, for each $n \in \mathbb{N}$ the configuration $\Delta_n = \{l(v) \mid v \in V, level(v) = n\}$ consists of the states at level n in Δ (cf. the dashed boxes in Figure 1).*

The reader may wonder why we mention trees at all in Definition 4. The reason is quite subtle: the tree structure is convenient for expressing a certain fairness property (cf. Lemma 2) that allows a Zeno run to be transformed into a non-Zeno run by inserting extra time delays.

Definition 4 only allows runs that start in a single state. More generally, we allow runs that start in an arbitrary configuration $C = \{(s_i, \nu_i)\}_{i \in I}$. Such a run is a *forest* consisting of $|I|$ different run trees, where the i-th run starts at (s_i, ν_i).

3.1 Translating Safety MTL into Timed Automata

Given a Safety MTL formula φ, one can define a timed alternating automaton $\mathcal{A}_\varphi$ such that $L(\mathcal{A}_\varphi) = L(\varphi)$. Since space is restricted, and since we have already given a similar translation in [16], we refer the reader to [18] for details. However, we draw the reader's attention to two important points. First, it is the restriction to timed-bounded until operators combined with the adoption of a non-Zeno semantics that allows us to translate a Safety MTL formula into an automaton in which every location is accepting; this is illustrated in Example 2, where location t, corresponding to the response formula $\Diamond_{=1} b$, is accepting. Secondly, we point out that each automaton $\mathcal{A}_\varphi$ is *local* according to the definition below. This last observation is important because it is the class of local automata for which Section 5 shows decidability of language emptiness.

Definition 6. *An automaton $\mathcal{A} = (\Sigma, S, s_0, \delta)$ is **local** if for each $s \in S$ and $a \in \Sigma$, each location $t \neq s$ appearing in $\delta(s, a)$ lies within the scope of a reset quantifier $x.(-)$, i.e., the automaton resets the clock whenever it changes location.*

We call such automata local because the static and dynamic scope of any reset quantification agree, i.e., the scope does not 'extend' across transitions to different locations. An investigation of the different expressiveness of local and non-local temporal logics is carried out in [8].

4 The Region Automaton

Throughout this section let $\mathcal{A} = (\Sigma, S, s_0, \delta)$ be a timed alternating automaton, and let c_{max} be the maximum constant appearing in a clock constraint in $\mathcal{A}$.

4.1 Abstract Configurations

We partition the set $\mathbb{R}_{\geq 0}$ of nonnegative real numbers into the set $REG = \{\mathsf{r}_0, \mathsf{r}_1, \ldots, \mathsf{r}_{2c_{max}+1}\}$ of *regions*, where $\mathsf{r}_{2i} = \{i\}$ for $i \leqslant c_{max}$, $\mathsf{r}_{2i+1} = (i, i+1)$ for $i < c_{max}$, and $\mathsf{r}_{2c_{max}+1} = (c_{max}, \infty)$. The *successor* of each region is given by

$succ(\mathsf{r}_i) = \mathsf{r}_{i+1}$ for $i < 2c_{max} + 1$ and $succ(\mathsf{r}_{2c_{max}+1}) = \mathsf{r}_{2c_{max}+1}$. Henceforth let r_{max} denote $\mathsf{r}_{2c_{max}+1}$ and write $reg(u)$ to denote the region containing $u \in \mathbb{R}_{\geq 0}$.

The *fractional part* of a nonnegative real $x \in \mathbb{R}_{\geq 0}$ is $frac(x) = x - \lfloor x \rfloor$.

We use the regions to define a discrete representation of configurations that abstracts away from precise clock values, recording only their values to the nearest integer and the relative order of their fractional parts, cf. [4].

Definition 7. *An **abstract configuration** is a finite word over the alphabet $\Lambda = \wp(S \times REG)$ of nonempty finite subsets of $S \times REG$.*

Define an *abstraction function* $H : \wp(Q) \to \Lambda^*$, yielding an abstract configuration $H(C)$ for each configuration C as follows. First, lift the function *reg* to configurations by $reg(C) = \{(s, reg(\nu)) : (s, \nu) \in C\}$. Now given a configuration C, partition C into a sequence of subsets $C_1, \ldots, C_n$, such that for all $(s, \nu) \in C_i$ and $(t, \nu') \in C_j$, $frac(\nu) \leqslant frac(\nu')$ iff $i \leqslant j$ (so (s, ν) and (t, ν') are in the same block C_i iff ν and ν' have the same fractional part). Then define $H(C) = \langle reg(C_1), \ldots, reg(C_n) \rangle \in \Lambda^*$.

Example 3. Consider the automaton $\mathcal{A}$ from Example 1. The maximum clock constant appearing in $\mathcal{A}$ is 1, and the corresponding regions are $\mathsf{r}_0 = \{0\}$, $\mathsf{r}_1 = (0, 1)$, $\mathsf{r}_2 = \{1\}$ and $\mathsf{r}_3 = (1, \infty)$. Given a concrete configuration $C = \{(s, 1), (t, 0.4), (s, 1.4), (t, 0.8)\}$, the corresponding abstract configuration $H(C)$ is $\langle \{(s, \mathsf{r}_2)\}, \{(t, \mathsf{r}_1), (s, \mathsf{r}_3)\}, \{(t, \mathsf{r}_1)\} \rangle$.

The image of the function H, which is a proper subset of Λ^*, is the set of *well-formed* words according to the following definition.

Definition 8. *Say that an abstract configuration $w \in \Lambda^*$ is **well-formed** if it is empty or if both of the following hold.*

- *The only letter of w containing a pair (s, r) with r a singular region is the first letter w_0.*
- *Whenever w_0 contains a singular region, the only nonsingular region that also appears in w_0 is r_{max}.*

Write $W \subseteq \Lambda^*$ for the set of well formed words.

We model the progression of time by introducing the notion of the *time successor* of an abstract configuration. We first illustrate the idea informally with concrete configurations.

Example 4. Consider a configuration $C = \{(s, 1.2), (t, 2.5), (s, 0.8)\}$. Intuitively, the time successor of C is $C' = \{(s, 1.4), (t, 2.7), (s, 1)\}$, where time has advanced 0.2 units and the clock value in C with largest fractional part has moved to a new region. On the other hand, a time successor of $C = \{(s, 1), (t, 0.5)\}$ is obtained after any time evolution δ, with $0 < \delta < 0.5$, so that the clock value with zero fractional part moves to a new region, while all other clock values remain in the same region. (Different values of δ lead to different configurations, but the underlying abstract configuration is the same.)

The definition below formally introduces the time successor of an abstract configuration. The two clauses correspond to the two different cases in Example 4. The first clause models the case where a clock with zero fractional part advances to the next region, while the second clause models the case where the clock with maximum fractional part advances to the next region.

Definition 9. *Let $w = w_0 \cdots w_n \in W$ be an abstract configuration. We say that w is **transient** if w_0 contains a pair (s, r) with r singular.*

- *If $w = w_0 \cdots w_n$ is transient, then its **time successor** is $w_0' w_1 \cdots w_n$, where $w_0' = \{(s, succ(\mathsf{r})) : (s, \mathsf{r}) \in w_0\}$.*
- *If $w = w_0 \cdots w_n$ is not transient, then its **time successor** is $w_n' w_0 \cdots w_{n-1}$, where $w_n' = \{(s, succ(\mathsf{r})) : (s, \mathsf{r}) \in w_n\}$.*

4.2 Definition of $R(\mathcal{A})$

The *region automaton* $R(\mathcal{A})$ is a nondeterministic infinite-state untimed automaton (with ε-transitions) that mimics $\mathcal{A}$. The states of $R(\mathcal{A})$ are abstract configurations, representing levels in a run of $\mathcal{A}$, and the transition relation contains those pairs of states representing consecutive levels in a run. We partition the transitions into two classes: *conservative* and *progressive*. Intuitively, a transition is progressive if it cycles the fractional order of the clock values in a configuration. This notion will play a role in our analysis of non-Zenoness in Section 5.

The definition of $R(\mathcal{A})$ is as follows:

- **Alphabet.** The alphabet of $R(\mathcal{A})$ is Σ.
- **States.** The set of states of $R(\mathcal{A})$ is the set $W \subseteq \Lambda^*$ of well-formed words over alphabet $\Lambda = \wp(S \times REG)$. The initial state is $\{(s_0, \mathsf{r}_0)\}$.
- **ε-transitions.** If $w \in W$ has time successor $w' \neq w$, then we include a transition $w \xrightarrow{\varepsilon} w'$ (excluding self-loops here is a technical convenience). This transition is classified as conservative if w is transient, otherwise it is progressive.
- **Labelled transitions.** Σ-labelled transitions in $R(\mathcal{A})$ represent instantaneous transitions of $\mathcal{A}$. Given $a \in \Sigma$, we include a transition $w \xrightarrow{a} w'$ in $R(\mathcal{A})$ if there exist $\mathcal{A}$-configurations C and C' with $H(C) = w$, $H(C') = w'$, $C = \{(s_i, \nu_i)\}_{i \in I}$ and
$$C' = \bigcup_{i \in I} \{M_i : \ M_i \models_{\nu_i} \delta(s_i, a)\} .$$
We say that this transition is progressive if $C' = \emptyset$ or
$$\max\{frac(\nu) : (s, \nu) \in C'\} < \max\{frac(\nu) : (s, \nu) \in C\} , \qquad (1)$$
otherwise we say that the transition is conservative. Note that (1) says that the clocks in C with maximal fractional part get reset in the course of the transition.

The above definition of the Σ-labelled transition relation (as a quotient) is meant to be succinct and intuitive. However, it is straightforward to compute the successors of each state $w \in W$ directly from the transition function δ of $\mathcal{A}$. For example, if $\delta(s, a) = s \wedge x.t$ then we include a transition $\langle \{(s, \mathsf{r}_1)\} \rangle \xrightarrow{a} \langle \{(t, \mathsf{r}_0)\}, \{(s, \mathsf{r}_1)\} \rangle$ in $R(\mathcal{A})$.

Given $a \in \Sigma$, write $w \overset{a}{\Longrightarrow} w'$ if w' can be reached from w by a sequence of ε-transitions, followed by a single a-transition. The following is a variant of [16, Definition 15].

Lemma 1. *Let Δ be a run of $\mathcal{A}$ on a timed word (σ, τ), and recall that $\Delta_n \subseteq Q$ is the set of states labelling the n-th level of Δ. Then $R(\mathcal{A})$ has a run*

$$[\Delta] : \ H(\Delta_0) \overset{\sigma_0}{\Longrightarrow} H(\Delta_1) \overset{\sigma_1}{\Longrightarrow} H(\Delta_2) \overset{\sigma_2}{\Longrightarrow} \cdots$$

on the untimed word $\sigma \in \Sigma^\omega$.

Conversely, if $R(\mathcal{A})$ has an infinite run r on $\sigma \in \Sigma^\omega$, then there is a time sequence τ and a run Δ of $\mathcal{A}$ on (σ, τ) such that $[\Delta] = r$.

Lemma 1 is a first step towards reducing the language-emptiness problem for $\mathcal{A}$ to the language-emptiness problem for $R(\mathcal{A})$. What is lacking is a characterisation of non-Zeno runs of $\mathcal{A}$ in terms of $R(\mathcal{A})$. Also, since $R(\mathcal{A})$ has infinitely many states, its own language-emptiness problem is nontrivial. We deal with both these issues in Section 5.

5 A Decision Procedure for Satisfiability

Let $\mathcal{A}$ be a local timed alternating automaton. We give a procedure for determining whether $\mathcal{A}$ has nonempty language. The key ideas are as follows. We define the notion of a *progressive* run of the region automaton $R(\mathcal{A})$, such that $R(\mathcal{A})$ has a progressive run iff $\mathcal{A}$ has a non-Zeno run. We then use a backward-reachability analysis to determine the set of states of $R(\mathcal{A})$ from which there is a progressive run. The effectiveness of this analysis depends on a well-quasi-order on the states of $R(\mathcal{A})$.

5.1 Background on Well-quasi-orders

Recall that a *quasi-order* on a set Q is a reflexive and transitive relation $\preccurlyeq \subseteq Q \times Q$. Given such an order we say that $L \subseteq Q$ is a *lower set* if $x \in Q$, $y \in L$ and $x \preccurlyeq y$ implies $x \in L$. The notion of an *upper set* is similarly defined. We define the *upward closure* of $S \subseteq Q$, denoted $\uparrow S$, to be $\{x \mid \exists y \in S : y \preccurlyeq x\}$. This is the smallest upper set that contains S. A *basis* of an upper set U is a subset $U_b \subseteq U$ such that $U = \uparrow U_b$. A *cobasis* of a lower set L is a basis of the upper set $Q \setminus L$.

Definition 10. *A* ***well-quasi-order*** *(wqo) is a quasi-order $(Q, \preccurlyeq)$ such that for any infinite sequence $q_0, q_1, q_2, \ldots$ in Q, there exist indices $i < j$ such that $q_i \preccurlyeq q_j$.*

Example 5. Let $\leqslant$ be a quasi-order on a finite 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 $f : \{1 \ldots m\} \to \{1, \ldots, n\}$ such that $a_i \leqslant b_{f(i)}$ for all $i \in \{1, \ldots, m\}$. Higman's Lemma states that if $\leqslant$ is a wqo on Λ, then the induced monotone domination order $\preccurlyeq$ is a wqo on Λ^*.

Proposition 1. *[9, Lemma 2.4] Let $(Q, \preccurlyeq)$ be a wqo. Then*

1. *each lower set $L \subseteq Q$ has a finite cobasis;*
2. *each infinite decreasing sequence $L_0 \supseteq L_1 \supseteq L_2 \supseteq \cdots$ of lower sets eventually stabilises, i.e., there exists $k \in \mathbb{N}$ such that $L_n = L_k$ for all $n \geqslant k$.*

5.2 Progressive Runs

Definition 11. *Overloading terminology, we say that a run $r : w \longrightarrow w' \longrightarrow w'' \longrightarrow \cdots$ of $R(\mathcal{A})$ is* ***progressive*** *if it contains infinitely many progressive transitions.*

The above definition is motivated by the notion of a progressive run of an (ordinary) timed automaton [4, Definition 4.11]. However our definition is more primitive. In particular, Lemma 2, which for us is a *property* of progressive runs, is the actual analog of Alur and Dill's *definition* of a progressive run.

Lemma 2. *Suppose Δ is a run of $\mathcal{A}$ over (σ, τ) such that the corresponding run $[\Delta]$ of $R(\mathcal{A})$ is progressive. Then there exists an infinite sequence of integers $n_0 < n_1 < \cdots$ such that $\tau_{n_0} < \tau_{n_1} < \cdots$ and every path in Δ running from a level-n_i node to a level-n_{i+1} node contains a node (s, ν) in which $\nu = 0$ or $\nu > c_{max}$.*

We use Lemma 2 in the proof of Theorem 1 below, which closely follows [4, Lemma 4.13].

Theorem 1. *$\mathcal{A}$ has a non-Zeno run iff $R(\mathcal{A})$ has a progressive run.*

Proof (sketch). It is straightforward that if Δ is a non-Zeno run of $\mathcal{A}$, then $[\Delta]$ is a progressive run of $R(\mathcal{A})$. The interesting direction is the converse.

Suppose that $R(\mathcal{A})$ has a progressive run r on a word $\sigma \in \Sigma^\omega$. Then by Lemma 1 there is a time sequence τ and a run Δ of $\mathcal{A}$ over (σ, τ) such that $[\Delta] = r$. If τ is non-Zeno then there is nothing to prove. We therefore suppose that τ is Zeno, and show how to modify Δ by inserting extra time delays to obtain a non-Zeno run Δ'.

Since τ is Zeno there exists $N \in \mathbb{N}$ such that $\tau_j - \tau_i < 1/4$ for all $i, j \geqslant N$. Let $n_0 < n_1 < \cdots$ be the sequence of integers in Lemma 2 where, without loss of generality, $N < n_0$. Define a new time sequence τ' by inserting extra delays in τ as follows:

$$\tau'_{i+1} - \tau'_i = \begin{cases} \tau_{i+1} - \tau_i & \text{if } i \notin \{n_1, n_2, \ldots\} \\ 1/2 & \text{if } i \in \{n_1, n_2, \ldots\}. \end{cases}$$

Clearly τ' is non-Zeno. We claim that a run Δ' over the timed word (σ, τ') can be constructed by appropriately modifying the clock values of the states occurring in Δ to account for the extra delay. What needs to be checked here is that the modified clock values remain in the same region.

Consider a path π through Δ, and let $\pi[m, n]$ denote the segment of π from level m to level n in Δ. If the clock x does not get reset in the segment $\pi[n_0, n_i]$ for some i, then, by Lemma 2, it is continuously greater than c_{max} along the segment $\pi[n_1, n_i]$: so the extra delay in Δ' is harmless on this part of π. Now if x gets reset in the segment $\pi[n_i, n_{i+1}]$ for some i, it can thereafter never exceed $1/4$ along π. Thus, by Lemma 2, it must get reset at least once in every segment $\pi[n_j, n_{j+1}]$ for $j \geqslant i$. In this case the extra delay in Δ' is again harmless. $\square$

5.3 Fixed-Point Characterisation

Let $PR \subseteq W$ denote the set of states of $R(\mathcal{A})$ from which a progressive run can originate. In order to compute PR we first characterise it as a fixed-point.

Definition 12. *Let $I \subseteq W$ be a set of states of $R(\mathcal{A})$. Define $Pred_+(I)$ to consist of those $w \in W$ such that there is a (possibly empty) sequence of conservative transitions $w \longrightarrow w' \longrightarrow w'' \longrightarrow \cdots \longrightarrow w^{(n)}$, followed by a single progressive transition $w^{(n)} \longrightarrow w^{(n+1)}$, such that $w^{(n+1)} \in I$.*

It is straightforward that PR is the greatest fixed point of $Pred_+(-) : 2^W \to 2^W$ with respect to the set-inclusion order[3]. Given this characterisation, one idea to compute PR is via the following decreasing chain of approximations:

$$W \supseteq Pred_+(W) \supseteq (Pred_+)^2(W) \supseteq \cdots . \tag{2}$$

But it turns out that we have to refine this idea a little to get an effective procedure. We start by observing the existence of a well-quasi-order on W.

Definition 13. *Define the quasi-order $\preccurlyeq$ on $W \subseteq \Lambda^*$ to be the monotone domination order over Λ (cf. Example 5).*

We might hope to use Proposition 1 to show that the chain (2) stabilises after finitely many steps. However $Pred_+$ does not map lower sets to lower sets in general. This reflects a failure of the progressive-transition relation to be downwards compatible with $\preccurlyeq$ in the sense of [9]. (This is not surprising—the possibility of $w \in W$ performing a progressive transition depends on its first and last letters.)

Example 6. Consider the automaton $\mathcal{A}$ in Example 2, with associated regions including $\mathsf{r}_0 = \{0\}$, $\mathsf{r}_1 = (0, 1)$ and $\mathsf{r}_2 = \{1\}$. Then, in $R(\mathcal{A})$, $w = \langle\{(s, \mathsf{r}_1)\}, \{(t, \mathsf{r}_1)\}\rangle$ makes a progressive ε-transition to $w' = \langle\{(t, \mathsf{r}_2)\}, \{(s, \mathsf{r}_1)\}\rangle$. However, $\langle\{(s, \mathsf{r}_1)\}\rangle$, which is a subword of w, does not belong to $Pred_+(\downarrow w')$. Indeed, any state reachable from $\langle\{(s, \mathsf{r}_1)\}\rangle$ by a sequence of conservative transitions followed by a single progressive transition must contain the letter $\{(s, \mathsf{r}_2)\}$.

[3] It is not possible for w to belong to the greatest fixed point of $Pred_+$ merely by virtue of being able to perform an infinite consecutive sequence of ε-transitions that includes infinitely many progressive ε-transitions. The reason is that once all the clock values in a configuration have advanced beyond the maximum of clock constant c_{max}, then the configuration is no longer capable of performing ε-transitions (cf. Section 4.2.)

Although $Pred_+$ fails to enjoy one-step compatibility with $\preccurlyeq$, it satisfies a kind of infinitary compatibility. More precisely, even though $Pred_+$ does not map lower sets to lower sets, its greatest fixed point is a lower set.

Proposition 2. *PR is a lower set.*

Proof. We exploit the correspondence between non-Zeno runs of $\mathcal{A}$ and progressive runs of $R(\mathcal{A})$, as given in Proposition 1.

Suppose $w' \in PR$ and $w \preccurlyeq w'$. Then there exist $\mathcal{A}$-configurations C, C' such that $C \subseteq C'$, $H(C) = w$ and $H(C') = w'$. Since $w' \in PR$, by Proposition 1 $\mathcal{A}$ has a run Δ' on some non-Zeno word ρ such that $\Delta'_0 = C'$. Now let Δ be the subgraph of Δ' consisting of all nodes reachable from those level-0 nodes of Δ' labelled by elements of $C \subseteq C'$. Then Δ is also a run of $\mathcal{A}$ on ρ, so $w \in PR$ by Proposition 1 again. $\square$

In anticipation of applying Proposition 2, we make the following definition.

Definition 14. *Define $\Psi : 2^W \to 2^W$ by $\Psi(I) = W \setminus \uparrow (W \setminus Pred_+(I))$.*

By construction, Ψ maps lower sets to lower sets. Also, being a monotone self-map of $(2^W, \subseteq)$, it has a greatest fixed point, denoted $gfp(\Psi)$.

Proposition 3. *PR is the greatest fixed point of Ψ.*

Proof. Since PR is both a fixed point of $Pred_+$ and a lower set we have:

$$\begin{aligned}
\Psi(PR) &= W \setminus \uparrow (W \setminus Pred_+(PR)) \\
&= W \setminus \uparrow (W \setminus PR) \\
&= W \setminus (W \setminus PR) \\
&= PR\,.
\end{aligned}$$

That is, PR is a fixed point of Ψ. It follows that $PR \subseteq gfp(\Psi)$.

The reverse inclusion, $gfp(\Psi) \subseteq PR$ follows easily from the fact that $\Psi(I) \subseteq Pred_+(I)$ for all $I \subseteq W$. $\square$

Next we assert that Ψ is computable.

Proposition 4. *Given a finite cobasis of a lower set $L \subseteq W$, there is a procedure to compute a finite cobasis of $\Psi(L)$.*

Proposition 4 is nontrivial since the definition of Ψ involves $Pred_+$, which refers to multi-step reachability (by conservative transitions), not just single-step reachability. We refer the reader to [18] for a detailed proof. The proof exploits the fact that conservative transitions on local automata have a very restricted ability to transform a configuration—for instance, the only way they can change the order of the fractional values of the clocks is by resetting some clocks to 0.

5.4 Main Results

Theorem 2. *The satisfiability problem for Safety MTL is decidable.*

Proof. Since every Safety MTL formula can be translated into a local automaton, it suffices to show that language emptiness is decidable for local automata.

Given a local automaton $\mathcal{A}$, let Ψ be as in Definition 14. Since Ψ is monotone and maps lower sets to lower sets, $W \supseteq \Psi(W) \supseteq \Psi^2(W) \supseteq \cdots$ is a decreasing sequence of lower sets in $(W, \preccurlyeq)$. By Proposition 1 this sequence stabilises after some finite number of iterations. By construction, the stabilising value is the greatest fixed point of Ψ, which by Proposition 3 is the set PR. Furthermore, using Proposition 4 we can compute a finite cobasis of each successive iterate $\Psi^n(W)$ until we eventually obtain a cobasis for PR. We can then decide whether the initial state of $R(\mathcal{A})$ is in PR which, by Theorem 1, holds iff $\mathcal{A}$ has nonempty language. □

We leave the complexity of the satisfiability problem for future work. The argument used to derive the nonprimitive recursive lower bound for MTL satisfiability over finite timed words [16] does not apply here.

By combining the techniques used to prove Theorem 2 with the techniques used in [16] to show that the model-checking problem is decidable for Safety MTL, one can show the decidability of the refinement problem: 'Given two Safety MTL formulas φ_1 and φ_2, does every word satisfying φ_1 also satisfy φ_2?'

Theorem 3. *The refinement problem for Safety MTL is decidable.*

6 Conclusion

It is folklore that extending linear temporal logic in any way that enables expressing the *punctual* specification 'in one time unit φ will hold' yields an undecidable logic over a dense-time semantics. Together with [17], this paper reveals that there is an unexpected factor affecting the truth or falsity of this belief. While [17] shows that Metric Temporal Logic is undecidable over timed ω-words, the proof depends on being able to express liveness properties, such as $\Box\Diamond p$. On the other hand, this paper shows that the safety fragment of MTL remains fully decidable in the presence of punctual timing constraints. This fragment is not closed under complement, and the decision procedures for satisfiability and model checking are quite different. The algorithm for satisfiability solves a nontermination problem on a well-structured transition system by iterated backward reachability, while the algorithm for model checking, given in a previous paper [16], used forward reachability.

Acknowledgement. The authors would like to thank the anonymous referees for providing many helpful suggestions to improve the presentation of the paper.

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 3580, 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. Real-time logics: complexity and expressiveness. *Information and Computation*, 104:35–77, 1993.
7. R. Alur and T. A. Henzinger. A really temporal logic. *Journal of the ACM*, 41:181–204, 1994.
8. P. Bouyer, F. Chevalier and N. Markey. On the expressiveness of TPTL and MTL. *Research report LSV-2005-05*, Lab. Spécification et Vérification, May 2005.
9. A. Finkel and P. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1-2):63–92, 2001.
10. T. A. Henzinger. It's about time: Real-time logics reviewed. In *Proceedings of CONCUR 98*, LNCS 1466, 1998.
11. T. A. Henzinger, Z. Manna and A. Pnueli. What good are digital clocks? In *Proceedings of ICALP 92*, LNCS 623, 1992.
12. T. A. Henzinger, J.-F. Raskin, and P.-Y. Schobbens. The regular real-time languages. In *Proceedings of ICALP 98*, LNCS 1443, 1998.
13. G. Higman. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, 2:236–366, 1952.
14. R. Koymans. Specifying real-time properties with metric temporal logic. *Real-time Systems*, 2(4):255–299, 1990.
15. S. Lasota and I. Walukiewicz. Alternating timed automata. In *Proceedings of FOSSACS 05*, LNCS 3441, 2005.
16. J. Ouaknine and J. Worrell. On the decidability of Metric Temporal Logic. In *Proceedings of LICS 05*, IEEE Computer Society Press, 2005.
17. J. Ouaknine and J. Worrell. Metric temporal logic and faulty Turing machines. Proceedings of FOSSACS 06, LNCS, 2006.
18. J. Ouaknine and J. Worrell. Safety MTL is fully decidable. Oxford University Programming Research Group Research Report RR-06-02.
19. J.-F. Raskin and P.-Y. Schobbens. State-clock logic: a decidable real-time logic. In *Proceedings of HART 97*, LNCS 1201, 1997.
20. M. Vardi. Alternating automata: Unifying truth and validity checking for temporal logics. In *Proceedings of CADE 97*, LNCS 1249, 1997.
21. F. Wang. Formal Verification of Timed Systems: A Survey and Perspective. *Proceedings of the IEEE*, 92(8):1283–1307, 2004.
22. 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.