Some Recent Results in Metric Temporal Logic

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 survey results about the complexity of the satisfiability and model checking problems for fragments of MTL with respect to different semantic models. We show that these fragments have widely differing complexities: from polynomial space to non-primitive recursive and even undecidable. However we show that the most commonly occurring real-time properties, such as invariance and bounded response, can be expressed in fragments of MTL for which model checking, if not satisfiability, can be decided in polynomial or exponential space.

1 Introduction

Linear temporal logic (LTL) is a popular formalism for the specification and verification of concurrent and reactive systems [28]. Most approaches that use LTL adopt a discrete model of time, where a run of a system produces a sequence of observations. Such a model is inadequate for real-time systems, where a run of a system is modelled either as a sequence of events that are time-stamped with reals or as a trajectory with domain the set $\mathbb{R}_+$ of non-negative reals.

In fact, interpretations of LTL on the reals were considered long before temporal logic became popular in verification. For example, the celebrated result of Kamp [20] that LTL with the "until" and "since" modalities is expressively complete for the first-order monadic logic over $(\mathbb{N}, <)$ also holds for the structure $(\mathbb{R}_+, <)$. A more recent development has been the extension of LTL to allow specifying quantitative or metric properties over the reals [21, 23]. For example, when specifying the behaviour of a real-time system one may want to stipulate deadlines between environment events and corresponding system responses: every *alarm* is followed by a *shutdown* event in 10 time units unless *all clear* is sounded first.

The most widely known such extension is *Metric Temporal Logic (MTL)* in which the modalities of LTL are augmented with timing constraints [21]. For example, the informally stated property above might be rendered

$$\Box(\mathit{alarm} \rightarrow (\Diamond_{(0,10)}\mathit{allclear} \vee \Diamond_{\{10\}}\mathit{shutdown}))$$

in MTL. Here $\Diamond_{(0,10)}$ means *sometime in the next* 10 *time units*, while $\Diamond_{\{10\}}$ means *in exactly* 10 *time units*.

An alternative approach to extending LTL is embodied in *Timed Propositional Temporal Logic (TPTL)* [6]. TPTL is a version of first-order temporal logic in which first-order variables range over the time domain and there is a restricted form of quantification, called *freeze quantification*, in which every variable is bound to the time of a particular state. In TPTL the above property could be written

$$\Box x.(\mathit{alarm} \rightarrow (\Diamond y.(\mathit{allclear} \wedge y - x < 10) \vee \Diamond z.(\mathit{shutdown} \wedge z - x = 10))\,.$$

Here x is bound to the time of the *alarm* event, y is bound to the time of the *allclear* event, and z is bound to the time of the *shutdown* event.

The relative expressiveness of MTL and TPTL is investigated in [8]. In particular it was shown there that MTL corresponds to a proper subset of the two-variable fragment of TPTL. In general it seems that the extra expressiveness of TPTL compared to MTL makes it harder to identify interesting decidable fragments of the former.

Yet another approach to reasoning about metric properties of computations is to work within the framework of monadic predicate logic [5, 17, 18, 31]. For example, Hirshfeld and Rabinovich [17] introduce the *Quantitative Monadic Logic of Order (QMLO)*, a fragment of first-order monadic logic over $(\mathbb{R}_+, <, +1)$. QMLO carefully restricts the use of the $+1$ function to a type of *bounded quantification*. For example, given a QMLO formula φ with one free variable,

$$(\exists t)^{<t_0+1}_{>t_0}\varphi \equiv \exists t(t_0 < t < t_0 + 1 \wedge \varphi(t))$$

denotes a formula with free variable t_0. It turns out that QMLO has the same expressiveness as a well-known decidable subset of MTL [17].

In this paper we concentrate on Metric Temporal Logic, although naturally many of the ideas we develop apply more widely. We survey a wide variety of complexity and decidability results for fragments MTL and show that these fragments have widely differing complexities: from PSPACE to non-primitive recursive and even undecidable. Reinforcing the message of [15], our objective is to illustrate that great care must be exercised in extending LTL to handle metric properties if one is to avoid a blow-up in the complexity of verification.

2 Metric Temporal Logic

Given a set P of atomic propositions, the formulas of MTL are built from P using Boolean connectives, and time-constrained versions of the *until* operator U as follows:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \, U_I \, \varphi\,,$$

where $I \subseteq (0, \infty)$ is an interval of reals with endpoints in $\mathbb{N} \cup \{\infty\}$. We sometimes abbreviate $U_{(0,\infty)}$ to U, calling this the *unconstrained* until operator.

Further connectives can be defined following standard conventions. In addition to propositions $\top$ (true) and $\bot$ (false), and to disjunction $\vee$, we have the *constrained eventually* operator $\Diamond_I \varphi \equiv \top \, U_I \, \varphi$, the *constrained always* operator $\Box_I \varphi \equiv \neg \Diamond_I \neg \varphi$, and the *constrained dual until* operator $\varphi_1 \, \widetilde{U}_I \, \varphi_2 \equiv \neg((\neg\varphi_1) \, U_I \, (\neg\varphi_2))$. Admitting only $\widetilde{U}_I$ as an extra connective one can transform any MTL formula into an equivalent *negation normal form*, in which negation is only applied to propositional variables.

Sometimes MTL is presented with past connectives (e.g., constrained versions of the "since" connective from LTL) as well future connectives [5]. However we do not consider past connectives in this paper.

Next we describe two commonly adopted semantics for MTL.

Continuous Semantics. Denote by $\mathbb{R}_+$ the set of nonnegative real numbers. Given a set of propositions P, a *signal* is a function $f \colon \mathbb{R}_+ \to 2^P$ mapping $t \in \mathbb{R}_+$ to the set $f(t)$ of propositions holding at time t. We say that f has *finite variability* if its set of discontinuities has no accumulation points. Given an MTL formula φ over the set of propositional variables P, the satisfaction relation $f \models \varphi$ is defined inductively, with the classical rules for atomic propositions and Boolean operators, and with the following rule for the "until" modality, where f^t denotes the signal $f^t(s) = f(t+s)$:

$$f \models \varphi_1 \, U_I \, \varphi_2 \text{ iff for some } t \in I,\ f^t \models \varphi_2 \text{ and } f^u \models \varphi_1 \text{ for all } u \in (0,t).$$

Pointwise Semantics. In the *pointwise semantics* MTL formulas are interpreted over *timed words*. Given an alphabet of events Σ, a timed word ρ is a finite or infinite sequence $(\sigma_0, \tau_0)(\sigma_1, \tau_1)\ldots$ where $\sigma_i \in \Sigma$ and $\tau_i \in \mathbb{R}_+$, such that the sequence (τ_i) is strictly increasing and *non-Zeno* (i.e., it is either finite or it diverges to infinity). The requirement of non-Zenoness is closely related to the condition of finite variability in the continuous semantics. It reflects the intuition that a system has only finitely many state changes in bounded time interval.

Given a (finite or infinite) timed word $\rho = (\sigma, \tau)$ over alphabet 2^P and an MTL formula φ, the satisfaction relation $\rho, i \models \varphi$ (read ρ satisfies φ at position i) is defined inductively, with the classical rules for Boolean operators, and with the following rule for the "until" modality:

$$\rho, i \models \varphi_1 \, \mathcal{U}_I \, \varphi_2 \text{ iff there exists } j \text{ such that } i < j < |\rho|,\ \rho, j \models \varphi_2,\ \tau_j - \tau_i \in I, \text{ and } \rho, k \models \varphi_1 \text{ for all } k \text{ with } i < k < j.$$

The pointwise semantics is less natural if one thinks of temporal logics as encoding fragments of monadic logic over the reals. On the other hand it seems more suitable when considering MTL formulas as specifications on timed automata. In this vein, when adopting the pointwise semantics it is natural to think of atomic propositions in MTL as referring to events (corresponding to state changes) rather than to states themselves.

For example, consider the specification of a traffic light. In the continuous semantics one might introduce propositions such as *green* and *red* to denote the *state* of the light. Then one could write a formula $\Box(\mathit{green} \to (\mathit{green} \ U_{(0,5)} \ \mathit{red}))$ to say that whenever the light is green it stays green until turning red in at most five time units. By contrast, in the pointwise semantics one would introduce propositions referring to *events*. For example, suppose the propositions *green* and *red* hold of events that respectively turn the traffic light green and red. Then the specification $\Box(green \rightarrow (\neg red\ U_{\{5\}}\ red))$ says that after the traffic light turns green it next becomes red after exactly 5 time units.

Decision Problems. This paper focuses on the following two fundamental decision problems:

- The *satisfiability problem*, asking whether a given MTL formula φ is satisfiable by some signal (or timed word).
- The *model-checking problem*, asking whether a given timed automaton A satisfies a given MTL formula φ, *i.e.*, whether all signals (or timed words) accepted by A satisfy φ (see [3] for details).

We consider satisfiability and model checking for various fragments of MTL, relative to the continuous semantics and both the finite-word and infinite-word variants of the pointwise semantics.

3 Alternating Timed Automata

In this section we review the notion of *alternating timed automata*. This class of automata is closely related to MTL and plays a key role in decision procedures for the latter over the pointwise semantics. (By contrast, over the continuous semantics it generally seems possible to avoid the use of automata in decision procedures [17].)

Following [22, 27] we define an alternating timed automaton to be an alternating automaton augmented with a single clock variable[1], which is denoted x. Given a finite set S of *locations* we define a set of formulas $\Phi(S, x)$ by the grammar:

$$\varphi ::= \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid s \mid x \sim c \mid x.\varphi\,,$$

where $c \in \mathbb{N}$, $\sim \in \{<, \leq, \geq, >\}$, and $s \in S$. A term of the form $x \sim c$ should be thought of as a *clock constraint*, whereas the expression $x.\varphi$ is a binding construct corresponding to the operation of resetting the clock x to 0.

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

Formally an *alternating timed automaton* is a tuple $A = (\Sigma, S, s_0, F, \delta)$, where

- 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

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

– $\delta : S \times \Sigma \to \Phi(S, x)$ is the transition function.

Before stating the formal definition of a run of an alternating timed automaton, we give an example of how an MTL formula can be translated into an equivalent automaton.

Example 1. The MTL formula $\Box(a \to \Diamond_{\{1\}} b)$ ('for every a-event there is a b-event exactly one time unit later') can be expressed by the following automaton A. Let A have two locations $\{s, t\}$ with s the initial and only accepting location, and transition function δ given by the following table:

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

Location s represents an invariant. When an a-event occurs, the conjunction in the definition of $\delta(s, a)$ results in the creation of a new thread of computation, starting in location t. Since this location is not accepting, the automaton must eventually leave it. This is only possible if a b-event happens exactly one time unit after the new thread was spawned.

We now proceed to formally define a run of an alternating timed automaton A. A *state* of A is a pair (s, v), where $s \in S$ is a location and $v \in \mathbb{R}_+$ is a *clock value*. Write $Q = S \times \mathbb{R}_+$ for the set of all states of A and define a *configuration* to be a finite subset of Q. A configuration $M \subseteq Q$ and a clock value $v \in \mathbb{R}_+$ defines a Boolean valuation on $\Phi(S, x)$ as follows:

– $M \models_v \varphi_1 \wedge \varphi_2$ if $M \models_v \varphi_1$ and $M \models_v \varphi_2$
– $M \models_v \varphi_1 \vee \varphi_2$ if $M \models_v \varphi_1$ or $M \models_v \varphi_2$
– $M \models_v s$ if $(s, v) \in M$
– $M \models_v x \sim c$ if $v \sim c$
– $M \models_v x.\varphi$ if $M \models_0 \varphi$.

A *tree* is a non-empty prefix closed set of *nodes* $T \subseteq \mathbb{N}^*$. A run of an alternating timed automaton A on a timed word $\rho = (\sigma, \tau)$ consists of a tree T and a *labelling* $l : T \to Q$ of the nodes of T by states of A such that: *(i)* $l(\varepsilon) = (s_0, 0)$ (the root is labelled by the initial state); and *(ii)* for each node $t \in T$ with $l(t) = (s, v)$, we have that $M \models_{v'} \delta(s, \sigma_n)$, where $n = |t|$ is the depth of t, $v' = v + (\tau_n - \tau_{n-1})$ and $M = \{l(t \cdot n) : t \cdot n \in T, n \in \mathbb{N}\}$ is the set of labels of the children of t. Finally, an infinite run is accepting if every infinite branch contains infinitely many accepting locations, while a run on a finite word ρ is accepting if every node at depth $|\rho|$ is accepting.

Example 1 can be generalised to show that for each MTL formula φ there is an alternating timed automaton A_φ such that $\{\rho : \rho \models \varphi\}$ is the set of timed words accepted by A. We refer the reader to [27] for details.

Well-quasi-order on Configurations. Recall that a *quasi-order* $(W, \preccurlyeq)$ consists of a set W together with a reflexive, transitive relation $\preccurlyeq$. An infinite sequence $w_1, w_2, w_3, \ldots$ in $(W, \preccurlyeq)$ is said to be *saturating* if there exist indices $i < j$ such that $w_i \preccurlyeq w_j$. $(W, \preccurlyeq)$ is said to be a *well-quasi-order (wqo)* if every infinite sequence is saturating.

Recall that the set of states of a classical (non-deterministic) timed automaton admits a finite quotient: the so called *clock regions construction* [1, 2]. In the case of alternating timed automata this generalises to a well-quasi-order on the set of configurations as we shortly explain. This well-quasi-order is important in establishing the termination of several decision procedures for MTL, e.g., in [26, 27].

Given an alternating timed automaton A, let c_{max} be the maximum clock constant in the description of A. Given configurations C and D, define $C \preccurlyeq D$ if there is an injection $f : C \to D$ such that: *(i)* $f(s, u) = (t, v)$ implies $s = t$ and either $\lfloor u \rfloor = \lfloor v \rfloor$ or $\lfloor u \rfloor, \lfloor v \rfloor > c_{max}$; *(ii)* if $f(s, u) = (s, u')$ and $f(s, v) = (t, v')$, then $\mathit{frac}(u) \leq \mathit{frac}(v)$ iff $\mathit{frac}(u') \leq \mathit{frac}(v')$. This quasi-order can be shown to be a well-quasi-order using Higman's Lemma [19, 27].

4 Decidable Sublogics: Continuous Semantics

It is well known that both model checking and satisfiability for MTL in the continuous semantics are highly undecidable (Σ_1^1-complete) [3]. In this section we explain how this undecidability arises, we discuss some of the syntactic restrictions that have been imposed to recover decidability, and we state the complexity of model checking and satisfiability for the resulting fragments of MTL.

4.1 Punctuality

From one point of view, the source of undecidability in MTL is the excessive precision of the timing constraints. In particular, MTL allows so-called *punctual* formulas, such as $\Diamond_{\{1\}} p$, in which the constraint is a singleton interval. Using such punctual formulas, given an arbitrary Turing machine M with input X, one can construct an MTL formula $\varphi_{M,X}$ such that the signals satisfying $\varphi_{M,X}$ encode accepting computations of M on X. Thus one reduces the halting problem to the MTL satisfiability problem.

Assume that the set of atomic propositions P includes a proposition for each tape symbol and control state of M. A configuration of M is then encoded by a sequence of propositions holding in a unit-length time interval in a given signal. The formula $\varphi_{M,X}$ includes a component φ_{INIT} to ensure that the first configuration agrees with X and a component φ_{TRAN} that ensures that successive configurations respect the transition function of M. For example, the punctual formula $p \leftrightarrow \Diamond_{\{1\}} p$ is used in φ_{TRAN} to indicate that a given tape cell is unchanged from one configuration to the next. Since there is no *a priori* bound on the length of M's computations φ_{TRAN} appears in $\varphi_{M,X}$ under the scope of the "always" operator $\Box$.

Researchers were thus led to propose syntactic subsets of MTL in which punctual formulas are not expressible. For example, the sub-logic $\mathsf{MTL}_{0,\infty}$ [3, 15] arises by requiring that the constraining interval I in any temporal modality either has left endpoint 0 or right endpoint ∞. This logic allows one to speak about the earliest or latest times that a formula becomes true; for example $\Box(p \rightarrow \Diamond_{(0,5)} q)$ is an $\mathsf{MTL}_{0,\infty}$ formula saying that every p-state is followed by a q-state within 5 time units. Satisfiability and model checking for $\mathsf{MTL}_{0,\infty}$ are both PSPACE-complete. Thus the addition of upper- and lower-bound timing constraints to LTL incurs no complexity blow-up.

A more general fragment of MTL that prohibits punctual specifications is *Metric Interval Temporal Logic (MITL)*. This is the subset of MTL in which the constraining interval I in any temporal modality is required to be non-singular. Alur, Feder and Henzinger [3] describe an exponential translation of MITL formulas into equivalent non-deterministic timed automata, leading to an EXPSPACE decision procedure for both model checking and satisfiability. It was shown in [18] that over the continuous semantics $\mathsf{MTL}_{0,\infty}$ and MITL are equally expressive, although the latter is exponentially more succinct. Over the pointwise semantics, however, MITL is strictly more expressive than $\mathsf{MTL}_{0,\infty}$ [15].

A version of MITL called *Quantitative Temporal Logic (QTL)* has been introduced by Hirshfeld and Rabinovich [17]. This logic simply augments LTL with the modality $\Diamond_{(0,1)}$ (and the correspond past modality). They show that QTL has the same expressiveness as the version of MITL with both "until" and "since" and give a PSPACE decision procedure for the satisfiability problem. In contrast to the approach of [3] this procedure does not involve automata, but rather goes via a satisfiability preserving translation of QTL into LTL. For each QTL formula φ using set of atomic propositions P one can define an LTL formula $\widetilde{\varphi}$, over an augmented set of propositions $P \cup Q$, such that a signal $f : \mathbb{R}_+ \rightarrow 2^{P \cup Q}$ satisfies $\widetilde{\varphi}$ iff there exists a piecewise-linear monotone bijection $g : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ such that $f \circ g$ satisfies φ. Intuitively, sets of signals that are definable in QTL (or MITL) are also definable in LTL up to some stretching.

4.2 Boundedness

A complementary route to obtaining decidable subsets of MTL has recently been propounded in [9, 10]. The idea is that rather than ban constraining intervals that are *too small*, one bans constraining intervals that are *too big*. (Note in this regard that the undecidability proof for MTL described above involved both the punctual "eventually" connective $\Diamond_{\{1\}}$ and the unbounded "always" connective $\Box$.) Thus [10] defines BMTL to be the subset of MTL in which all constraining intervals have finite length. For example, $\varphi \equiv \Box_{(0,10)}(p \leftrightarrow \Diamond_{\{1\}} q)$ is a BMTL formula.

As with MITL, the satisfiability and model checking problems for BMTL are EXPSPACE-complete. However, unlike MITL, it is not the case that BMTL formulas can be translated into equivalent timed automata. Indeed a variation on a well-known result tells us that the set of signals satisfying the example formula φ above cannot be the language of a timed automaton [2, 7].

Note that the above-defined encoding of Turing machines in MTL does not, when restricted to BMTL, yield EXPSPACE-hardness. Since we encode Turing-machine configurations in unit-length intervals, a BMTL version of the formula $\varphi_{M,X}$, described above, that has size polynomial in $|X|$ can only encode computations for which the number of steps is exponential in $|X|$. To achieve EXPSPACE-hardness requires a slightly different idea, although still crucially using punctuality.

Given a 2^n-space-bounded Turing machine M with input X, we construct in logarithmic space a BMTL formula $\varphi_{M,X}$ that is satisfiable if and only if M accepts X. The definition of $\varphi_{M,X}$ involves a set of atomic propositions $P \cup \dot{P}$, where P is as in the undecidability proof for MTL and $\dot{P} = \{\dot{p} : p \in P\}$. The dot is used as a pointer to aid in simulating M. The idea is to encode the entire computation of M in a single time unit, rather than encoding one configuration per time unit. In any signal satisfying $\varphi_{M,X}$ the sequence of propositions holding in the time interval $[0, 1)$ is meant to encode the computation history of M on input X. In this time interval we assume that the dot superscript decorates the first tape cell of each configuration of the computation.

The definition of $\varphi_{M,X}$ involves a formula

$$\begin{aligned}\varphi_{COPY} = &\bigwedge_{p \in P} \Box_{[0,2^{|X|}]}(p \rightarrow \Diamond_{\{1\}}(p \vee \dot{p})) \\ &\wedge \bigwedge_{p,q \in P} \Box_{[0,2^{|X|}]}((\dot{p}\, U_{(0,1)}\, q) \leftrightarrow \Diamond_{\{1\}}(p\, U_{(0,1)}\, \dot{q}))\,,\end{aligned}$$

that copies the sequence of propositions holding in each unit-duration time interval into the subsequent time interval, at the same time moving the dot superscript 'one place to the right'. Thus the sequence of propositions holding in each subsequent time interval $[k, k+1)$, $k = 1, \ldots, 2^{|X|} - 1$, should also represent the computation history of M on X. The only difference is that in the interval $[k, k+1)$ the dot decorates exactly those propositions encoding the contents of the k-th tape cell in each configuration in the computation history.

In addition to φ_{COPY}, $\varphi_{M,X}$ has another component φ_{CHECK}. For a given unit-length time interval $[k, k+1)$, φ_{CHECK} uses the dots as pointers to check the correctness of the k-th tape cell in each configuration. Thus, in $2^{|X|}$ time units the whole computation is checked.

4.3 Flatness

MITL and BMTL represent two different approaches to obtaining decidable metric temporal logics, and they have incomparable expressive power. In particular, BMTL is not capable of expressing invariance—one of the most basic safety specifications. To repair this deficiency [10] consider the syntactic property of *flatness* as a generalisation of boundedness. The term flatness here is motivated by by similarities with logics introduced in [12, 13]. Intuitively an MTL formula is flat if no punctual subformula appears within the scope of a connective that involves unbounded universal quantification over the time domain. In fact the most natural way to state the results of [10] is in terms of the dual notion to flatness, called *coflatness*.

The condition of coflatness applies to formulas in negation normal form. Recall that such formulas feature the constrained dual until operator $\widetilde{U}_I$ in addition to the constrained until operator. Formally we say an MTL formula in negation normal form is *coflat* if *(i)* in any subformula of the form $\varphi_1 \, U_I \, \varphi_2$, either I is bounded or φ_2 is in MITL, and *(ii)* in any subformula of the form $\varphi_1 \, \widetilde{U}_I \, \varphi_2$, either I is bounded or φ_1 is in MITL. If we write CFMTL for the sublogic of coflat formulas then CFMTL includes both BMTL and MITL, is closed under $\Box_I$ for arbitrary I (invariance), and is closed under U_I for bounded I (bounded liveness). Thus, for specifications, coflatness is a very natural and mild restriction.

The formula $\Box(\mathit{req} \to \Diamond_{(0,1)}(\mathit{acq} \wedge \Diamond_{\{1\}}\mathit{rel}))$ says that every time a lock is requested, it is acquired within one time unit, and released after exactly one further time unit. This formula is in CFMTL, but is not in BMTL (due to the unconstrained $\Box$) nor is it in MITL (due to the punctual $\Diamond_{\{1\}}$).

The main result of [10] is that the model-checking problem for CFMTL is EXPSPACE-complete; moreover this result holds irrespective of whether the constants in the timing constraints are encoded in unary or binary. In the case that constants are encoded in unary, the proof of EXPSPACE-hardness follows the same idea as the EXPSPACE-hardness proof for BMTL satisfiability. The matching upper bound is via a translation to LTL that incurs an exponential blow-up. Thus for the most commonly occurring specification patterns, such as invariance and bounded response, punctuality can be accommodated while model checking remains in EXPSPACE.

We emphasise that the above refers to model checking and not satisfiability. In fact the satisfiability problem for CFMTL is undecidable for the simple reason that all the formulas used in the proof of undecidability of satisfiability for MTL are coflat. (Note that CFMTL is not closed under negation.)

5 Decidable Sublogics: Pointwise Semantics

From another point of view, the source of undecidability in MTL is the richness of the semantic model. A natural restriction on the semantics is to interpret the logic on timed words in which all timestamps are integers. Indeed it is often argued that integer time suffices for most applications [16]. With respect to integer-valued timed words, satisfiability and model checking for MTL are easily seen to be EXPSPACE-complete, matching the complexity of MITL over the continuous semantics. The exponential blow-up over LTL arises from the possibility to write timing constraints succinctly in binary. We note that such succinct timing constraints are also the cause of the exponential blow-up in the complexity of MITL over MTL: if timing constraints are written in unary then model checking and satisfiability for MITL in the continuous semantics are both PSPACE-complete.

Keeping with timed words, but now allowing timestamps to be arbitrary real numbers, the situation becomes more delicate. Over finite timed words both model checking and satisfiability for MTL are decidable but not primitive recursive [27]. Over infinite timed words both problems are undecidable. Thus in the pointwise semantics the situation between decidability and undecidability is finely balanced.

The decidability of satisfiability and model checking for MTL over finite timed words was proved in [27] by giving a procedure for deciding language emptiness for alternating timed automata over finite words. That procedure used forward reachability analysis to search for an accepting computation tree on a given automaton. The well-quasi-order on configurations identified in Section 3 was used to prove the termination of such a search.

At this point it is instructive to see why the undecidability proof for MTL over the continuous semantics fails over the pointwise semantics. Consider the formula $\Box(a \leftrightarrow \Diamond_{\{1\}} a)$. For a timed word to satisfy this formula every a-event should be followed by another a-event after exactly one time unit. However the formula does not force every a-event to be preceded by an a-event one time unit earlier (for the reason that the former might not be preceded by *any* event exactly one time unit earlier). Thus if we try to encode computations of a Turing machine as timed words in MTL, we find that we can only encode the computations of a machine with *insertion errors*.

In fact, when considering such erroneous computation devices it is more convenient to talk about a class of computing devices called *insertion channel machines with emptiness-testing*, or *ICMET* [25]. Such devices consist of a finite control together with a fixed number of unbounded channels (or queues). Transitions between control states can write messages to the tail of a channel, read messages from the head of a channel, or perform an emptiness test on a channel.[2] There is a formal duality between such ICMETs and *lossy channel machines* [30].

The *control-state reachability problem* for ICMETs asks whether a given ICMET has a finite computation starting from the initial state and ending in an accepting control state. The *recurrent-state problem* for ICMETs asks whether a given ICMET has an infinite computation that visits an accepting state infinitely often. The control-state reachability problem is decidable, but not primitive recursive, while the recurrent-state problem is undecidable [25, 27, 30].

As suggested above, one can encode finite computations of ICMETs using MTL formulas over finite timed words; thus one shows that satisfiability and model-checking for MTL over finite timed words are not primitive recursive. Similarly one can reduce the recurrent-state problem for ICMETs to the satisfiability and model checking problems for MTL over infinite words, showing that the latter two problems are undecidable. In particular the formula $\Box\Diamond p$ is used to encode the fact that a computation visits an accepting state infinitely often.

[2] In contrast to models of asynchronous communication, we assume that the same finite control automaton writes to and reads from the channels.

5.1 Safety

The last remark above suggests that one might recover decidability over infinite timed words by restricting to safety properties. This approach was taken in [26, 27] which defined a syntactic fragment of MTL called *Safety MTL (SMTL)*. An MTL formula in negation normal form is said to be in SMTL if in any subformula of the form $\varphi_1 \ U_I \ \varphi_2$ the interval I is bounded. No restrictions are placed on the dual until connective $\widetilde{U}$. For example, if $\varphi \in \mathsf{SMTL}$ then $\Box\varphi$ and $\Diamond_{(0,5)}\varphi$ are both in SMTL. Informally the requirement for a formula to be in SMTL is that all eventualities be bounded.

Following the classical semantic definition of safety property in the untimed setting, a set Π of infinite timed words is said to be a safety property if any timed word $\rho \notin \Pi$ has a finite prefix ρ', such that no extension of ρ' lies in Π. Due to the assumption that timed words are non-Zeno, Henzinger [14] calls such properties 'safety relative to the divergence of time'. (In a dense-time model a bounded-response property, such as $\Diamond_{(0,5)}p$, can only be considered a safety property thanks to the assumption of non-Zenoness.) All SMTL formulas define semantic safety properties.

Continuing the thread of ideas from above, the undecidability proof for MTL over infinite timed words does not apply to SMTL, since the latter cannot encode a *recurrent* computation of an ICMET. One can reduce the *termination problem* for ICMETs [11] to the satisfiability problem for SMTL, but the former is decidable though non-elementary. In fact one can again use alternating timed automata and the well-quasi-order from Section 3 to show that both satisfiability and model checking for SMTL are decidable [26, 27].

Similarly to CFMTL, SMTL is suitable for defining invariance and time-bounded response properties. Comparing the two logics, we note that SMTL is more permissive in its use of $\widetilde{U}$, but less permissive in its use of U (cf. Section 4). Notwithstanding this superficial similarity, there is a chasm between the respective complexities of the model checking problem. Model checking for SMTL is non-primitive recursive [27] while it is EXPSPACE-complete for CFMTL.

6 Summary

We summarise the relationships between the various logics introduced in Sections 4 and 5 in the following diagram (where $\hookrightarrow$ indicates a syntactic inclusion):

$$\begin{tikzcd}
& & \mathsf{BMTL} \arrow[r, hook] \arrow[dr, hook] & \mathsf{SMTL} \arrow[dr, hook] & \\
\mathsf{LTL} \arrow[r, hook] & \mathsf{MTL}_{0,\infty} \arrow[r, hook] & \mathsf{MITL} \arrow[r, hook] & \mathsf{CFMTL} \arrow[r, hook] & \mathsf{MTL}
\end{tikzcd}$$

We also summarise complexity results for model-checking and satisfiability for different fragments of MTL in Table 1. In this table results that refer to the pointwise semantics are shaded in grey; all other results refer to the continuous semantics. The legend 'MTL (fin.)' in the last row stands for MTL over finite timed words.

	Model Checking	**Satisfiability**
LTL	PSPACE-c.	PSPACE-c.
$\mathsf{MTL}_{0,\infty}$	PSPACE-c.	PSPACE-c.
MITL	EXPSPACE-c.	EXPSPACE-c.
BMTL	EXPSPACE-c.	EXPSPACE-c.
SMTL	Non-Prim.-Rec.	Non-Elem.
CFMTL	EXPSPACE-c.	Undec.
MTL	Undec.	Undec.
MTL (fin.)	Non-Prim.-Rec	Non-Prim.-Rec

Table 1. Complexity of fragments of MTL

References

1. R. Alur, C. Courcoubetis and D. L. Dill. Model-checking for real-time systems. In *Proceeding of LICS 90*, IEEE Comp. Society Press, 1990.
2. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science* 126:183–235, 1994.
3. R. Alur, T. Feder and T. A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM* 43:116–146, 1996.
4. R. Alur and T. A. Henzinger. Logics and models of real time: A survey. In *Proceedings of Real Time: Theory in Practice*, LNCS 600, Springer, 1992.
5. R. Alur and T. A. Henzinger. Real-time logics: complexity and expressiveness. *Information and Computation* 104:35–77, 1993.
6. R. Alur and T. A. Henzinger. A really temporal logic. *Journal of the ACM* 41:181–204, 1994.
7. R. Alur and P. Madhusudan. Decision Problems for Timed Automata: A Survey. In *Proceedings of SFM-RT 04*, LNCS 3185, Springer, 2004.
8. P. Bouyer, F. Chevalier and N. Markey. On the Expressiveness of TPTL and MTL. In *Proceedings of FSTTCS 05*, LNCS 3821, Springer, 2005.
9. P. Bouyer, N. Markey, J. Ouaknine and J. Worrell. The Cost of Punctuality. In *Proceedings of LICS 07*, IEEE Computer Society Press, 2007.
10. P. Bouyer, N. Markey, J. Ouaknine and J. Worrell. On Expressiveness and Complexity in Real-time Model Checking. In *Proceedings of ICALP 08*, LNCS, to appear.
11. P. Bouyer, N. Markey, J. Ouaknine, Ph. Schnoebelen and J. Worrell. On Termination for Faulty Channel Machines. In *Proceedings of STACS 08*, 2008.
12. H. Comon and V. Cortier. Flatness is not a Weakness. In *Proceedings of CSL 00*, LNCS 1862, Springer, 2000.
13. S. Demri, R. Lazić, and D. Nowak. On the Freeze Quantifier in Constraint LTL: Decidability and Complexity. In *Information and Computation*: 205(1):2–24, 2007.

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, Springer, 1998.
16. T. A. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In *Proceedings of ICALP 92*, LNCS 623, Springer, 1992.
17. Y. Hirshfeld and A. M. Rabinovich: Logics for Real Time: Decidability and Complexity. *Fundam. Inform.* 62(1):1–28, 2004.
18. T. A. Henzinger, J.-F. Raskin, and P.-Y. Shobbens. The regular real-time languages. In *Proceedings of ICALP 98*, LNCS 1443, Springer, 1998.
19. G. Higman. Ordering by divisibility in abstract algebras. *Proc. of the London Mathematical Society* 2:236–366, 1952.
20. J. A. W. Kamp. Tense logic and the theory of linear order. Ph.D. Thesis, UCLA, 1968.
21. R. Koymans. Specifying real-time properties with metric temporal logic. *Real-time Systems* 2(4):255–299, 1990.
22. S. Lasota and I. Walukiewicz. Alternating timed automata. In *Proceedings of FOSSACS 05*, LNCS 3441, Springer, 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. Ouaknine and J. Worrell. Metric temporal logic and faulty Turing machines. In *Proceedings of FOSSACS 06*, LNCS 3921, Springer, 2006.
26. J. Ouaknine and J. Worrell. Safety metric temporal logic is fully decidable. In *Proceedings of TACAS 06*, LNCS 3920, Springer, 2006.
27. J. Ouaknine and J. Worrell. On the Decidability and Complexity of Metric Temporal Logic over Finite Words. In *Logicical Methods in Computer Science*, 3(1), 2007.
28. A. Pnueli. The temporal logic of programs. In *Proceedings of FOCS 77*, IEEE Computer Society Press, 1977.
29. J.-F. Raskin and P.-Y. Schobbens. State-clock logic: a decidable real-time logic. In *Proceedings of HART 97*, LNCS 1201, Springer, 1997.
30. P. Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity. *Information Processing Letters* 83(5):251–261, 2002.
31. T. Wilke. Specifying timed state sequences in powerful decidable logics and timed automata. *Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 863, Springer, 1994.