Timed temporal logics

Patricia Bouyer[1], François Laroussinie[2], Nicolas Markey[3], Joël Ouaknine[4,5], and James Worrell[5]

[1] LSV, CNRS & ENS Paris-Saclay, France
[2] IRIF, CNRS & Université Paris Diderot, France
[3] IRISA, CNRS & INRIA & Université Rennes 1, France
[4] Max Planck Institute for Software Systems, Germany
[5] Department of Computer Science, Oxford University, UK

Abstract. Since the early 1990's, classical temporal logics have been extended with timing constraints. While temporal logics only express contraints on the order of events, their timed extensions can add quantitative constraints on delays between those events. We survey expressiveness and algorithmic results on those logics, and discuss semantic choices that may look unimportant but do have an impact on the questions we consider.

1 Introduction

Timed automata [6] are a well-established model for real-time systems. One of their most fundamental properties is that reachability properties can be decided. This has given rise to multiple works, both on theoretical aspects and on more algorithmic and practical aspects. Several tools have even been developed for automatically verifying timed automata, for instance HyTech [28], Kronos [21] or Uppaal [41,12]. Among the success stories of that model, one can cite the verification and the correction of the Bang & Olufsen audio/video protocol [27] made using the tool Uppaal.

Timed automata are adequate to represent systems, but not that much for representing properties of systems. If $\mathcal{A}$ is a timed automaton representing the system, and $\mathcal{P}$ a timed automaton representing the property, verifying that $\mathcal{A}$ satisfies the property $\mathcal{P}$ corresponds to checking that all behaviours of $\mathcal{A}$ are also behaviours of $\mathcal{P}$. This is a language-inclusion question, which turns out to be undecidable for timed automata [6].

In order to circumvent this difficulty, following the development of temporal logics in model-checking [20,50], timed temporal logics have been proposed, which extend classical untimed temporal logics with timing constraints. There are several ways of expressing such constraints, a standard one consists in constraining temporal modalities. For instance, one can write a formula such as

$$\mathbf{G}\,(\mathsf{problem} \to \mathbf{F}_{\leq 5\,\mathrm{min}}\,\mathsf{repair})$$

to express the *quantitative* property that any problem must be followed by a repair action within 5 minutes. This kind of properties cannot be expressed using

standard temporal logics, as those logics can only refer to the relative order of events, not to their relative distance (in time).

Several timed extensions of CTL [20] and LTL [50] have been proposed. In this paper, we focus on some of those extensions that have been studied for the purpose of model-checking real-time systems. We start with the definition of timed automata, and we discuss several possible semantics for this model (Section 2). While the choice of semantics is harmless for many issues, it is crucial here in the context of timed temporal logics. We then turn to branching-time logics, and present TCTL as well as timed extensions of modal logics (Section 3). We end with linear-time logics, which are strongly related to first-order logics over the reals (Section 4). We end up with some conclusions and with further research directions (Section 5).

2 Continuous vs. pointwise semantics

Timed automata [6] are extensions of standard finite automata with finitely many clock variables. These variables, which take their values in a time domain, aim at constraining delays between events. The choice of the time domain has been discussed from the early definition of the model (see e.g. [4]); there has been a clear partition between papers considering dense-time domains such as the set $\mathbb{Q}_{\geq 0}$ of nonnegative rationals, or the set $\mathbb{R}_{\geq 0}$ of nonnegative reals, and papers considering a discrete-time domain like the set $\mathbb{Z}_{\geq 0}$ of nonnegative integers. In this paper, we assume that the time domain is $\mathbb{R}_{\geq 0}$.

In the setting of dense time, there is another distinction, which has been less clearly identified in the framework of timed automata: it is related to the nature of runs in a timed automaton. Indeed, the observation of the system can be considered continuous (executions are then viewed as *signals*), or it can be discrete (executions are then viewed as *timed words*) [4,52]. This distinction will be important in the context of logics, as we will see in this article. We begin with discussing this issue.

2.1 Timed automata

Timed automata extend finite-state automata with a finite set C of clocks, which measure delays between events that occur in the automaton. A configuration of a timed automaton is thus given by a pair (s, v) where s is a state of the automaton and $v\colon \mathsf{C} \to \mathbb{R}_{\geq 0}$ is a clock valuation. For $d \in \mathbb{R}_{\geq 0}$, we let $v' = v + d$ be the clock valuation such that $v'(c) = v(c) + d$ for each clock, corresponding to letting d time units elapse. For a subset $R \subseteq \mathsf{C}$, we let $v' = v[R]$ be the valuation such that $v'(c) = 0$ when $c \in R$, and $v'(c) = v(c)$ when $c \in \mathsf{C} \setminus R$. This corresponds to resetting clocks in R.

A clock constraint is a conjunction of atomic constraints of the form $c \in J$, where $c \in \mathsf{C}$ and J is an interval of $\mathbb{R}_{\geq 0}$ with bounds in $\mathbb{Z}_{\geq 0} \cup \{+\infty\}$. Whether a clock valuation satisfies a clock constraint is defined in the natural way. We write $\mathcal{G}(\mathsf{C})$ for the set of clock constraints on C, and $\mathcal{G}_M(\mathsf{C})$ for the set of all clock constraints on C using integer constants less than or equal to integer M.

Definition 1. *Let AP be a finite set of atomic propositions. A* timed automaton $\mathcal{A} = \langle S, C, E, \ell \rangle$ *over AP is made of a finite set S of states, a finite set C of clocks, a finite set of edges $E \subseteq S \times \mathcal{G}(C) \times 2^C \times S$, and a labelling function $\ell \colon S \to 2^{AP}$.*

The operational semantics of a timed automaton is defined through an infinite-state transition system, whose states are all the configurations $(s, v) \in \mathsf{S} \times \mathbb{R}_{\geq 0}^{\mathsf{C}}$, with transitions from configuration (s, v) to configuration (s', v') when one of the following two conditions is fulfilled:

- $s = s'$ and there exists a delay $d \in \mathbb{R}_{>0}$ such that $v' = v + d$;[1]
- there exists an edge $e = (s, g, R, s') \in E$ such that $v \models g$ and $v' = v[R]$.

This transition system mixes discrete changes (given by the second rule) with continuous changes due to time elapsing (given by the first rule). In particular, since delays are taken in $\mathbb{R}_{>0}$, the underlying graph has infinite branching.

2.2 Semantics for temporal logics over timed automata

We assume the reader is reasonably familiar with standard untimed temporal logics like LTL [50] and CTL [19,51]. While these untimed logics can well be interpreted over timed automata, extensions with quantitative constraints over delays are very much relevant in this setting. To define such constraints, one can either decorate the modalities with intervals specifying time delays that are allowed to satisfy the properties, or explicitely use clock variables in the formulas, in pretty much the same way as they are used in automata. These considerations will be discussed specifically in the sections over branching-time logics and linear-time logics.

There is a second important issue with interpreting temporal logics over timed automata, which is semantical. We need indeed to make precise which part of the behaviour of the timed automaton $\mathcal{A} = \langle \mathsf{S}, \mathsf{C}, \mathsf{E}, \ell \rangle$ is observed. We illustrate the possible choices using the *constrained until* formula. Intuitively, $\phi_1 \, \mathbf{U}_J \, \phi_2$ (where J is an interval of $\mathbb{R}_{\geq 0}$ with bounds in $\mathbb{Q}_{\geq 0} \cup \{+\infty\}$) holds along an execution of $\mathcal{A}$ if it is the case that ϕ_2 eventually holds, within a delay that belongs to interval J, and that ϕ_1 holds at all intermediary points in time. We will see that the choice of the semantics (more precisely, which *intermediary points in time* we consider) is crucial.

Discrete-observation semantics. A natural way to observe the system is to see paths in the transition system of the timed automaton as sequences of configurations reached when the automaton performs discrete transitions.

Formally, a path is a (finite or infinite) sequence $\pi = (s_i, v_i)_{i<L}$ of configurations, such that there is a delay transition between (s_i, v_i) and $(s_i, v_i + d_i)$, and

[1] Zero-delay transitions are not allowed here, but could be included without affecting the presented results.

a discrete transition between $(s_i, v_i + d_i)$ and (s_{i+1}, v_{i+1}). Notice that we do not require time divergence here, even for paths of infinite length.

For convenience, we assume that our timed automata include a special clock, named t hereafter, that is never reset and never used in any timing constraint.

The discrete-observation semantics (also called the pointwise semantics in the literature) of the constrained-until modality along a path $\pi = (s_i, v_i)_{i<L}$ in $\mathcal{A}$ can be defined as follows:

$$\mathcal{A}, \pi \models_{\mathsf{disc}} \phi_1 \,\mathbf{U}_J\, \phi_2 \quad \Leftrightarrow \quad \exists n > 0.\ \mathcal{A}, \pi_{\geq n} \models_{\mathsf{disc}} \phi_2 \text{ and } v_n(t) - v_0(t) \in J \text{ and } \forall 0 < m < n.\ \mathcal{A}, \pi_{\geq m} \models_{\mathsf{disc}} \phi_1 \quad (1)$$

where $\pi_{\geq k}$ is the path $(s_i, v_i)_{k \leq i < L}$. We see here that satisfaction of subformulas is checked *only* at discrete time points, precisely when there is a transition taken in the timed automaton, and not while delaying in the timed automaton. Notice that we consider the strict version of the until modality, imposing no constraint in the present time point. This is an arbitrary choice, which makes the logic slightly more expressive.

Continuous-observation semantics. It is also natural to consider continuous observations of the evolution of the automaton: let $\pi = (s_i, v_i)_{<L}$ be a path as formerly defined, with an additional global clock t. We associate with π a *signal* ϖ which maps every nonnegative real number to the configuration of the system at that time: for every $r \in \mathbb{R}_{\geq 0}$, $\varpi(r) = (s_i, v)$ where i is the largest index such that $v_i(t) \leq r$, and $v = v_i + r - v_i(t)$. This can be interpreted intuitively as follows: the system is observed continuously, hence when time elapses, increasing values of clocks are observed. So, at time $v_i(t)$, state s_i is entered, and then, while delaying, all clocks increase. We made the arbitrary choice to assume that at time $v_i(t)$, the system is already in state s_i. In order to avoid arbitrary switches between states, it is often required that ϖ has finite variability, that is, its set of discontinuities has no limit points.

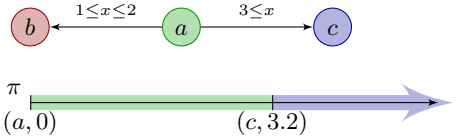
The continuous-observation semantics of the constrained-until modality along a path π (or equivalently, along its associated signal ϖ) in $\mathcal{A}$ can then be defined as follows:

$$\mathcal{A}, \varpi \models_{\mathsf{cont}} \phi_1 \,\mathbf{U}_J\, \phi_2 \quad \Leftrightarrow \quad \exists r > 0.\ \mathcal{A}, \varpi_{\geq r} \models_{\mathsf{cont}} \phi_2 \text{ and } r \in J \text{ and } \forall 0 < r' < r.\ \mathcal{A}, \varpi_{\geq r'} \models_{\mathsf{cont}} \phi_1 \quad (2)$$

where $\varpi_{\geq r}$ is the signal which associates to $r' \in \mathbb{R}_{\geq 0}$ the value $\varpi(r' + r)$. We also write $\mathcal{A}, \pi \models_{\mathsf{cont}} \phi$ when $\mathcal{A}, \varpi \models_{\mathsf{cont}} \phi$.

Example 1. Consider the timed automaton depicted on Fig. 1. A path in that timed automaton is $\pi = (a, 0)(c, 3.2)$. The corresponding signal is ϖ which associates to every $r < 3.2$ the configuration (a, r) and to every $r \geq 3.2$ the configuration (c, r).

Interestingly, we get different satisfaction relations, depending on the particular choice of the semantics. Classically, the modality $\mathbf{F}_J\, \phi$ stands for $\mathtt{true}\, \mathbf{U}_J\, \phi$.



	Discrete observation	Continuous observation
$\varphi_1 = a \,\mathbf{U}\, c$	$\pi \models \varphi_1$	$\varpi \models \varphi_1$
$\varphi_2 = \mathbf{F}_{\leq 2}\, \mathbf{F}\, c$	$\pi \not\models \varphi_2$	$\varpi \models \varphi_2$
$\varphi_3 = \neg a \,\mathbf{U}\, c$	$\pi \models \varphi_3$	$\varpi \not\models \varphi_3$

Fig. 1. A run π of a timed automaton, and its value against some formulas

Formula φ_2 in Fig. 1 requires the existence of an intermediary point along the execution where the subformula $\mathbf{F}\, c$ holds. This is the case in the continuous-observation setting, but not in the discrete-observation setting. On the other hand, φ_3 holds on π since there is no point in time where $\neg a$ has to be tested, whereas φ_3 does not hold on ϖ.

3 Branching-time temporal logics with timing constraints

In this section, we present some of the main results about the branching-time framework.

3.1 Timed CTL

Let us begin with the simpler case of plain CTL, with no constraints on *until* modalities. The main ingredient for model checking CTL, which already gives its lower-bound, is the original algorithm for reachability in timed automata:

Theorem 1 ([6]). *Reachability in timed automata is* PSPACE*-complete.*

Let $\mathcal{A} = \langle \mathsf{S}, \mathsf{C}, \mathsf{E}, \ell \rangle$ be a timed automaton, and let M be the maximal constant appearing in a clock constraint of $\mathcal{A}$. The above result is proved by quotienting the infinite state space of timed automata into finitely many *regions*: a region for a timed automaton $\mathcal{A}$ is made of a state of $\mathcal{A}$ and of sets of valuations defined by equivalence classes of the *region equivalence* $\equiv_M$. This relation is defined by $v \equiv_M v'$ whenever: for every $x \in \mathsf{C}$, (i) $v(x) > M$ iff $v'(x) > M$, (ii) if $v(x) \leq M$, then the integral parts of $v(x)$ and $v'(x)$ coincide; and for every $x, y \in \mathsf{C}$ such that $v(x), v(y) \leq M$, $\{v(x)\} \leq \{v(y)\}$ iff $\{v'(x)\} \leq \{v'(y)\}$ ($\{\cdot\}$ denotes the fractional part). The main property of region equivalence is that it defines a time-abstract bisimulation, and a finite automaton called the *region automaton* can be constructed based on this equivalence, which represents in an abstract manner the behaviour of $\mathcal{A}$.

A second ingredient for CTL model checking is that any two configurations whose clock valuations belong to the same region satisfy the same CTL formulas. As a consequence, standard CTL model checking can be performed on the region automaton by labelling all regions with the subformulas they satisfy. Such an algorithm would take exponential time in the worst case, since the number of regions is exponential. Several techniques can be used to circumvent this blowup,

e.g. by using tree automata, or space-efficient techniques recomputing the information on-demand while evaluating the truth value of a formula. In the end:

Theorem 2. *CTL model checking is* PSPACE*-complete over timed automata.*

Remark 1. It must be noted that there are CTL formulas that take different truth values in a given region, depending on the (discrete-observation or continuous-observation) semantics. Consider formula $\mathbf{E}(\neg\mathbf{EF}\, c)\,\mathbf{U}\, b$. This formula expresses the existence of a path eventually reaching a b-state, without visiting intermediary states from which c would be reachable. In the automaton depicted at Fig. 1, this formula holds true in the discrete-observation semantics, as witnessed by the path $(a,0)(b,3)$: along this path, the latter condition holds vacuously. Obviously, in the continuous-observation semantics, the formula fails to hold.

This can be reflected in the algorithm by considering different constructions for the region automaton: for the discrete-observation semantics, we would merge a delay and an action transition into a single transition of the region automaton (as performed e.g. in [6]). For the continuous-observation semantics we would have delay transitions to the immediate time-successor region only, and action transitions directly translated in the region automaton (as e.g. in [5]).

We now focus on TCTL. Two versions of this logic have been considered in the literature, either using decorated modalities or with formula clocks. We only consider the latter logic here, as it has more expressive power while having very similar algorithmic properties. Syntactically, the logic is defined as

$$\mathsf{TCTL} \ni \phi ::= \top \mid p \mid x \in J \mid \neg\phi \mid \phi \wedge \phi \mid \mathbf{E}\phi\,\mathbf{U}\,\phi \mid \mathbf{A}\phi\,\mathbf{U}\,\phi \mid x \cdot \phi$$

where p ranges over the set of atomic propositions AP, x ranges over a finite set of formula clocks C_F (these are *not* the clocks appearing in the automaton), J ranges over the set of intervals of $\mathbb{R}_{\geq 0}$ with integral bounds[2].

The two semantics discussed in Section 2.2 for the Until modality can be applied to TCTL. The semantics of $\top$ (always true) and of Boolean operators is omitted. Given a configuration (s,v) of $\mathcal{A}$, and a valuation u for the formula clocks, the satisfaction relation is defined as:

$$\begin{array}{lcl}
\mathcal{A},(s,v,u) \models p & \Leftrightarrow & p \in \ell(s) \\
\mathcal{A},(s,v,u) \models x \in J & \Leftrightarrow & u(x) \in J \\
\mathcal{A},(s,v,u) \models x \cdot \phi & \Leftrightarrow & \mathcal{A},(s,v,u[\{x\}]) \models \phi \\
\mathcal{A},(s,v,u) \models \mathbf{E}\phi_1\,\mathbf{U}\,\phi_2 & \Leftrightarrow & \text{there is a path } \pi \text{ (resp. signal } \varpi\text{) from } (s,v,u) \text{ s.t.} \\
 & & \mathcal{A},\pi \models \phi_1\,\mathbf{U}\,\phi_2 \text{ (resp. } \mathcal{A},\varpi \models \phi_1\,\mathbf{U}\,\phi_2\text{)} \\
\mathcal{A},(s,v,u) \models \mathbf{A}\phi_1\,\mathbf{U}\,\phi_2 & \Leftrightarrow & \text{for every path } \pi \text{ (resp. signal } \varpi\text{) from } (s,v,u), \\
 & & \mathcal{A},\pi \models \phi_1\,\mathbf{U}\,\phi_2 \text{ (resp. } \mathcal{A},\varpi \models \phi_1\,\mathbf{U}\,\phi_2\text{)}
\end{array}$$

[2] Rational bounds could be considered at the expense of scaling all constants by an appropriate factor.

Formula clocks are integrated to the paths and signals (without being reset by the timed automaton).

Example 2. The constrained-until formula $\mathbf{E}\phi_1 \,\mathbf{U}_J\, \phi_2$ can be written as:

$$x \cdot \mathbf{E}\phi_1 \,\mathbf{U}\, (\phi_2 \wedge x \in J)$$

It is not difficult to extend the CTL model-checking algorithm above to TCTL: one easily shows that again two valuations in the same region satisfy the same TCTL formulas. This can be shown by induction on the structure of the formula, taking formula clocks into account in the definition of region equivalence. Then the algorithm is similar to the algorithm for CTL, again taking care of the considered semantics.

Theorem 3 ([5]). *TCTL model checking is* PSPACE*-complete over timed automata (regardless of the semantics choice).*

Note that the syntax of the specification language used in Uppaal is inspired from TCTL, but basically all properties can be reduced to some kind of reachability properties. See Remark 2 later for more details.

As is the case for CTL, TCTL cannot express fairness properties. In particular, it cannot rule out Zeno runs, which are infinite runs along which time converges. Following the untimed approach, one may consider TCTL^*, in which Until modalities can be freely nested (without inserting path quantifications). This logic then embeds MTL, the extension of LTL with timing constraints, for which model checking is undecidable (see Section 4). An intermediary fragment is defined in [17], with the following syntax:

$$\begin{aligned} \mathsf{TCTL}_{\mathsf{LTL}} \ni \phi_s &::= \top \mid p \mid x \in J \mid \neg\phi_s \mid \phi_s \wedge \phi_s \mid \mathbf{E}\phi_p \mid x \cdot \phi_s \\ \phi_p &::= \phi_s \mid \neg\phi_p \mid \phi_p \wedge \phi_p \mid \phi_p \,\mathbf{U}\, \phi_p. \end{aligned}$$

Notice that in this fragment, formula clocks may only be reset at the level of state formulas. This allows us to recover decidability of model checking (in exponential space) [17], while being able to express fairness properties.

We conclude this section with a few words on the satisfiability problem for TCTL. We only deal here with *finite satisfiability* [5], asking whether there exists a finite-state timed automaton in which a given TCTL formula holds true. This problem is undecidable, which can be derived from the undecidability of satisfiability for MTL (see Section 4) by pairing each temporal modality with a universal path quantifier: such a formula is satisfiable if, and only if, the original MTL formula is. It is proven in [37] that forbidding equality constraints in TCTL makes finite satisfiability decidable; this is to be compared with what happens for MITL in the setting of linear-time logics (see Theorem 9).

3.2 Timed modal logics

In this section, we consider timed modal logics. The development of these logics is related to the attempt to extend different frameworks to the timed setting,

such as Milner's work on process algebra CCS, the HML logic [44], and the framework of modal specifications [40]. Here we only consider the logic part, and we interpret formulas over timed automata (see [54] for a contribution on timed CCS, [18] for the timed modal specifications and [26] for a presentation of the tool Epsilon for timed modal specifications).

Let Σ be a finite alphabet of *actions*. We assume that every edge of a timed automaton is labelled with an action $a \in \Sigma$, in addition to the guard and the set of clocks to be reset; thus we now assume $\mathsf{E} \subseteq \mathsf{S} \times \mathcal{G}(\mathsf{C}) \times \Sigma \times 2^{\mathsf{C}} \times \mathsf{S}$. Since modal logics are appropriate for compositional analysis, we also consider parallel compositions of timed automata $(\mathcal{A}_1 \mid \ldots \mid \mathcal{A}_n)_f$, where f is an n-ary synchronization function over Σ with renaming. We refer to [3, Section 4] for a formal definition, but the intuition is that f specifies how the various automata should synchronize on labels over transitions; for instance, f can force processes $\mathcal{A}_1$ and $\mathcal{A}_2$ to synchronize on action a while producing a b, by defining $f(a, a, \bullet, \ldots, \bullet) = b$; here labels $\bullet$ indicate that the corresponding processes do not take part in the synchronization. Such a parallel composition does not add expressive power (i.e., the parallel composition of several automata is equivalent to a single automaton) but it is a convenient way to describe complex systems. We will see that the modal logics we consider enjoy interesting expressiveness and compositionality properties over such parallel compositions.

HML is a modal logic interpreted over labelled transition systems: in addition to Boolean operators, there are two modalities: the existential and the universal quantification over actions (which we denote $\langle a \rangle$ and $[a]$, respectively). For example, formula $[a]\langle b \rangle \top$ specifies that after any a-transition, a b-transition is enabled.

Timed extensions of HML use the same syntax and, moreover, allow one to quantify over delay transitions: for delay transitions, instead of using explicit values (representing the delays) as labels, we consider a symbolic label δ to represent any delay; $\langle \delta \rangle$ (resp. $[\delta]$) stands for the existential (resp. universal) quantification over delay transitions. The formula $[a][\delta]\langle b \rangle \top$ specifies that after any a-transition and any delay, a b-transition is enabled, while the formula $[a]\langle \delta \rangle \langle b \rangle \top$ specifies that after any a-transition, a b-transition will be enabled after some delay. To complete these modalities, we use formula clocks (as in TCTL): a formula clock x can be reset before evaluating φ (written $x \cdot \varphi$), and it can be used in constraints of the form $x \in J$, where J is an interval of $\mathbb{R}_{\geq 0}$ with bounds in $\mathbb{Q}_{\geq 0} \cup \{+\infty\}$. We use C_F to denote the set of formula clocks. Note that this logic has been mostly studied using the discrete-observation paradigm, even though one could extend it to a continuous-observation setting. In this section, we focus on the former semantics.

As for HML, we can add maximal or minimal fixpoint operators to specify properties over executions based on unbounded sequences of actions: for example $\mathtt{min}(X, \varphi \vee \bigvee_{a \in \Sigma} \langle a \rangle X \vee \langle \delta \rangle X)$ holds for a state when it is possible to reach a state satisfying φ. The dual formula $\mathtt{max}(X, \varphi \wedge \bigwedge_{a \in \Sigma} [a] X \wedge [\delta] X)$ specifies that φ holds for every reachable state. We use *Var* to denote the set of variables.

We can define several logics depending on which of the above operators are allowed. Here we just introduce the logic L_ν [39] whose syntax is given by the following grammar:

$$\mathsf{L}_\nu \ni \varphi, \psi ::= \top \mid \bot \mid x \sim c \mid x \cdot \varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \langle \ell \rangle \varphi \mid [\ell] \varphi \mid \mathtt{max}(X, \varphi) \mid X$$

where $\ell \in \Sigma \cup \{\delta\}$, $x \in \mathsf{C}_F$, $\sim \in \{<, >\}$, $c \in \mathbb{N}$, and $X \in \mathit{Var}$. An L_ν formula φ is interpreted over a configuration (s, v) of a timed automaton $\mathcal{A}$ (or over a configuration $(\bar{s}, \bar{v})$ of a parallel composition $\big(\mathcal{A}_1 \mid \ldots \mid \mathcal{A}_n\big)_f$) with a valuation u for the formula clocks. We omit the formal semantics, which can be derived from the previous discussion.

L_ν benefits from the same decidability properties as TCTL: two (extended) states in the same region satisfy the same L_ν formulas.

Theorem 4 ([3]). *L_ν model checking is* EXPTIME*-complete over (parallel compositions of) timed automata.*

The EXPTIME membership can be obtained by applying standard model-checking algorithms over the region automaton corresponding to the system (note that adding minimal fixpoints would not change the complexity). The EXPTIME-hardness proof uses the same encoding of linear-bounded Turing machines we use to show PSPACE-hardness of reachability in timed automata, extended to simulate *alternating* Turing machine with the existential and universal modalities in L_ν.

Remark 2. The tool Uppaal mostly analyzes reachability-like properties. It was therefore natural, early in the process of development of the tool, to properly understand which properties can be expressed and verified using the tool. To that aim, a fragment of L_ν has been investigated [1], which fully characterizes properties that can be expressed through a reachability query via *test automata*. A test automaton for a property $\varphi \in \mathsf{L}_\nu$ is a timed automaton $\mathcal{A}_\varphi$ such that for every timed automaton $\mathcal{A}$, it holds $\mathcal{A} \models \varphi$ if, and only if, some designated target set of states in the composition $\big(\mathcal{A} \mid \mathcal{A}_\varphi\big)_{f_s}$ where f_s enforces the synchronization of actions of the automata, is not reachable. The resulting fragment of L_ν has a PSPACE-complete model-checking problem.

L_ν is very expressive as a specification language. For example, it is easy to observe that timed bisimilarity can be expressed in L_ν: two timed automata over the same alphabet Σ $\mathcal{A}_1$ and $\mathcal{A}_2$ are *strongly timed bisimilar* (denoted $\mathcal{A}_1 \sim \mathcal{A}_2$) if, and only if, their parallel composition $\big(\mathcal{A}_1 \mid \mathcal{A}_2\big)_{f_{\text{inter}}}$, where f_{inter} is an interleaving synchronization with a renaming[3] of every action $a \in \Sigma$ of $\mathcal{A}_i$ by action a_i, satisfies the following L_ν formula:

$$\Psi_{\text{bisim}} = \mathtt{max}\Big(Z, \bigwedge_{a \in \Sigma} \big([a_1] \langle a_2 \rangle Z \wedge [a_2] \langle a_1 \rangle Z\big) \wedge [\delta] Z\Big).$$

[3] That is, for every $a \in \Sigma$, $f_{\text{inter}}(a, \bullet) = a_1$ and $f_{\text{inter}}(\bullet, a) = a_2$.

This ability to deal with single action transitions of an automaton is very useful and allows a *compositional algorithm* for model-checking (as for the classical modal μ-calculus [11]). Given a specification φ, an automaton $\mathcal{A}$ and a synchronization function f describing its interaction with another component $\mathcal{B}$, one can build a *quotient formula* $\varphi/_f\mathcal{A}$ such that $\big(\mathcal{A} \mid \mathcal{B}\big)_f \models \varphi$ if, and only if, $\mathcal{B} \models (\varphi/_f\mathcal{A})$. Note that the clocks of the quotiented automaton $\mathcal{A}$ become formula clocks in $\varphi/_f\mathcal{A}$: any behaviour of $\mathcal{A}$ that is relevant w.r.t. φ is encoded in the formula and this includes all timing informations.

By iterating this quotienting, one can reduce a model-checking instance $\big(\mathcal{A}_1 \mid \ldots \mid \mathcal{A}_n\big)_f \models \varphi$ to some question $\mathtt{nil} \models \varphi'$ where φ' is the quotient formula $\varphi/\mathcal{A}_1/\mathcal{A}_2/\ldots/\mathcal{A}_n$, and $\mathtt{nil}$ is a process letting time elapse without performing any action. Of course, in this approach, the size of the formula grows exponentially with quotienting (the state-space explosion problem is translated from the model to the formula), but this approach still provides an alternative way of performing model-checking [38], and it gives also many interesting results for such logics.

From the previous properties, it is easy to deduce the construction of *characteristic formulas* for timed automata: the quotient formula $\Psi_{\text{bisim}}/_{f_{\text{inter}}}\mathcal{A}_1$ holds true for some automaton $\mathcal{A}_2$ if, and only if, $\mathcal{A}_1 \sim \mathcal{A}_2$. The formula $\Psi_{\text{bisim}}/_{f_{\text{inter}}}\mathcal{A}_1$ is the characteristic formula of $\mathcal{A}_1$, it describes the precise behaviour of $\mathcal{A}_1$ up to timed bisimulation. See [2] for more results on characteristic formulas for timed automata.

Finally, quotienting is also useful for the control synthesis problem. The problem is defined as follows: given a system $\mathcal{S}$ to be controlled and a global specification Φ that has to be satisfied by the complete system, one aims at synthesizing a controller $\mathcal{C}$ such that $(\mathcal{S} \mid \mathcal{C})_f \models \Phi$. The quotient construction allows us to build a specification for the controller with $\Phi/_f\mathcal{S}$. Notice however that the satisfiability for L_ν is undecidable (actually even for its non-recursive fragment) [33], and only a strong bounded-resources version of the problem has been shown decidable [39].

4 Linear-time temporal logics with timing constraints

In this section we survey some of the main results concerning expressiveness and decidability of linear-time temporal logics in the metric setting. In general, a linear-time specification determines a set of runs of a given system: a collection of signals in the continuous semantics and a collection of timed words in the pointwise semantics. In this section we will mostly focus on the continuous semantics when talking about expressiveness (because the theory is cleaner), but we consider decidability issues with respect to both semantics.

The results surveyed in this section should be read in the context of two classical theorems about linear temporal logic in the non-metric setting. The first, a celebrated result of Kamp [34], is that the linear-time temporal logic (LTL) is

expressively complete for monadic first-order logic over both the ordered integers $(\mathbb{Z}, <)$ and ordered reals $(\mathbb{R}, <)$. The second result, due to Wolper, Vardi, and Sistla [53], is that the model checking problem for LTL formulas on Kripke structures is PSPACE-complete (Note that, notwithstanding the equivalent expressiveness of LTL and monadic first-order logic, the model checking problem for monadic first-order logic is non-elementary.)

4.1 Monadic First-Order Logic

A natural approach to specifying properties of signals is to use first-order logic. Consider a first-order language $\mathsf{L}_{\mathrm{MET}}$ over a signature with a binary relation symbol $<$, an infinite collection of unary predicate symbols $\mathsf{AP} = \{P_1, P_2, \ldots\}$, and an infinite family of unary function symbols $+q$, $q \in \mathbb{Q}$.

Formulas of $\mathsf{L}_{\mathrm{MET}}$ can naturally be interpreted over signals $\varpi \colon \mathbb{R} \to 2^{\mathsf{AP}}$. Such a signal determines a first-order structure in which the universe is $\mathbb{R}$, the relation symbol $<$ and function symbols $+q$, $q \in \mathbb{Q}$, are interpreted by the standard order relation and addition function on $\mathbb{R}$, and where each unary predicate symbol P_i is interpreted as $\{r \in \mathbb{R} \mid P_i \in \varpi(r)\}$. For example, the formula

$$\varphi(x) := \exists y \, \exists z \, ((x < y < z < x + 1) \wedge P(y) \wedge P(z)) \tag{3}$$

holds at a point $r \in \mathbb{R}$ in a signal if P is true at least twice in the open interval $(r, r + 1)$.

The satisfiability problem for $\mathsf{L}_{\mathrm{MET}}$ asks whether a given sentence is satisfied by some signal. The model-checking problem asks whether a given sentence is satisfied by all signals in the language of a given timed automaton.

Theorem 5. *The satisfiability and model checking problems for* $\mathsf{L}_{\mathrm{MET}}$ *are undecidable.*

Proof (Sketch). Let P be a monadic predicate symbol and consider the following two properties of a signal:

- for all $r \in \mathbb{R}$, P is true at r if and only if it is true at $r + 1$;
- the set of $r \in \mathbb{R}$ at which P holds has no accumulation point.

These two properties can easily be expressed in $\mathsf{L}_{\mathrm{MET}}$, using only the order relation $<$ and $+1$ function. Moreover any signal satisfying these properties embeds a grid of dimensions $\mathbb{Z} \times \{0, \ldots, N\}$, for some $N \in \mathbb{N}$, where the (i, j)-th cell in the grid maps to the j-th P-position within the open interval $(i, i+1)$. We can use the relation $<$ and function $+1$ to navigate horizontally and vertically through such a grid and thereby reduce the halting problem for Turing machines to the satisfiability problem for $\mathsf{L}_{\mathrm{MET}}$.

Undecidability of model checking follows immediately from undecidability of satisfiability. □

4.2 Metric Temporal Logic

The above-mentioned result of Kamp [34] on the expressiveness of LTL motivates the search for an expressively complete temporal logic for $\mathsf{L}_{\mathrm{MET}}$. A natural candidate is *Metric Temporal Logic (MTL)* [35], a popular and widely studied temporal logic that augments LTL with time-constrained versions of the *Until* and *Since* modalities (*Since* is symmetric to *Until*: $\varphi_1 \,\mathbf{S}\, \varphi_2$ requires that φ_2 holds at some position in the past, and that φ_1 holds in all intermediary positions).

Given a set AP of atomic propositions, the formulas of MTL are given by the following grammar

$$\mathsf{MTL} \ni \varphi ::= \top \mid p \mid \varphi \wedge \varphi \mid \neg\varphi \mid \varphi \,\mathbf{U}_I\, \varphi \mid \varphi \,\mathbf{S}_I\, \varphi\,,$$

where $p \in \mathsf{AP}$ and $I \subseteq (0,\infty)$ is an interval with endpoints in $\mathbb{Q}_{\geq 0} \cup \{\infty\}$. We also use derived boolean operators such as $\varphi_1 \to \varphi_2 = \neg\varphi_1 \vee \varphi_2$ and $\varphi_1 \leftrightarrow \varphi_2 = (\varphi_1 \to \varphi_2) \wedge (\varphi_2 \to \varphi_1)$, and derived temporal connectives like $\mathbf{F}_I\, \varphi = \top \,\mathbf{U}_I\, \varphi$ and $\mathbf{P}_I\, \varphi = \top \,\mathbf{S}_I\, \varphi$.

Note that we consider signals whose domain is the set $\mathbb{R}$ of all real numbers. Below we will also consider the future fragment of MTL on signals over the non-negative real numbers $\mathbb{R}_{\geq 0}$.

4.3 Expressive Completeness

At first glance MTL seems to have weak expressive power. For example, consider the formula (3) expressing that there will be at least two p-states in the next time unit. MTL cannot naturally express both the consecution of two events and a timing constraint on the *second* event. This led to the conjecture that such constraints cannot be expressed in MTL (try to express (3) before reading further!); cf. [9,10]. However, as shown in [14], this formula can indeed be expressed in MTL:

Example 3. We give an MTL formula $\varphi^\dagger$ that is equivalent to the $\mathsf{L}_{\mathrm{MET}}$ formula $\varphi(x)$ in (3) in the sense that for every signal ϖ and $r \in \mathbb{R}$, $\varpi \models \varphi[r]$ if and only if $\varpi, r \models \varphi^\dagger$. The key is to use fractional constants in the definition of $\varphi^\dagger$. We define the formula as a disjunction of three overlapping cases according to position of the two times at which p holds that witness the truth of φ.

$$\varphi^\dagger := \mathbf{F}_{(0,\frac{1}{2})}(p \wedge \mathbf{F}_{(0,\frac{1}{2})} p) \vee \mathbf{F}_{(0,\frac{1}{2})}(\mathbf{F}_{(0,\frac{1}{2})} p \wedge \mathbf{F}_{\{\frac{1}{2}\}} p) \vee (\mathbf{F}_{(0,\frac{1}{2})} p \wedge \mathbf{F}_{(\frac{1}{2},1)} p)\,.$$

The "trick" in the previous example of using fractional constants to boost the expressiveness of MTL turns out to be very powerful:

Theorem 6 ([32]). *For every* $\mathsf{L}_{\mathrm{MET}}$ *formula* $\varphi(x)$ *there is an equivalent MTL formula* $\varphi^\dagger$.

Let us briefly discuss two key ideas underlying the proof of Theorem 6: namely *boundedness* and *separation*. Given $N \in \mathbb{N}$, an $\mathsf{L}_{\mathrm{MET}}$ formula $\varphi(x)$ is N-bounded if all quantifiers are relativised to the interval $(x-N, x+N)$. Exploiting a normal

form for $\mathsf{FO}(<)$ [25], we show how to translate bounded $\mathsf{L}_{\mathrm{MET}}$ formulas into MTL. Extending this translation to arbitrary $\mathsf{L}_{\mathrm{MET}}$ formulas requires an appropriate analog of Gabbay's notion of *separation* [24].

Gabbay [24] shows that every LTL formula can be equivalently rewritten as a Boolean combination of formulas, each of which depends only on the past, present, or future. This property underlies an inductive translation from first-order logic over $(\mathbb{R}, <)$ to LTL. The proof of Theorem 6 relies on an analogous result for MTL:

Lemma 1 (Separation Lemma). *Every MTL formula can be equivalently rewritten as a Boolean combination of MTL formulas, each of which has one of the following three forms:*

- Bounded*: the interval I in every temporal operator $\mathbf{U}_I$ and $\mathbf{S}_I$ is bounded;*
- Distant Future*: has the form $\mathbf{F}_{(1,\infty)}\varphi$, for φ a formula with no past connectives;*
- Distant Past*: has the form $\mathbf{P}_{(1,\infty)}\varphi$, for φ a formula with no future connectives.*

Gabbay's separation result for LTL is an ingredient of the proof of the Separation Lemma for MTL. As we have said, the latter result can be used to give an inductive translation from $\mathsf{L}_{\mathrm{MET}}$ to MTL. A key difference to the purely order-theoretic case is that in the metric setting the different types of formulas in the Separation Lemma may talk about overlapping parts of the signal. For this reason it is crucial that we already have a separate translation of bounded $\mathsf{L}_{\mathrm{MET}}$ formulas to MTL.

Integer Constants. Having rational constants plays a crucial role in the proof of Theorem 6. Indeed, as illustrated in Example 3, the translation from $\mathsf{L}_{\mathrm{MET}}$ to MTL does not preserve the granularity of timing constraints. Pursuing this issue, define $\mathsf{L}^{(1)}_{\mathrm{MET}}$ to be the fragment of $\mathsf{L}_{\mathrm{MET}}$ in which the family of unary addition function symbols $+q$, $q \in \mathbb{Q}$, is replaced by a single unary function symbol $+1$. It was shown by Hirshfeld and Rabinovich [29] that MTL with integer constants is not expressively complete for $\mathsf{L}^{(1)}_{\mathrm{MET}}$. Indeed [29] proves a much stronger impossibility result: no temporal logic whose modalities are definable by a set of formulas of $\mathsf{L}^{(1)}_{\mathrm{MET}}$ of bounded quantifier depth can be expressively complete for $\mathsf{L}^{(1)}_{\mathrm{MET}}$. Later, and again based on the Separation Lemma, Hunter [31] gave an expressively complete temporal logic for $\mathsf{L}^{(1)}_{\mathrm{MET}}$ by taking the fragment of MTL with integer constants and augmenting it with with an infinite family of unary *counting modalities* $\mathbf{C}_n$ (first considered in [29]).

Given a positive integer n, the semantics of the counting modality $\mathbf{C}_n$ is defined as follows:

- $\varpi, r \models \mathbf{C}_n(\varphi)$ if there exist $r < r_1 < \ldots < r_n < r + 1$ such that $\varpi, r_i \models \varphi$ for $i = 1, \ldots, n$.

Notice that the $\mathsf{L}^{(1)}_{\mathrm{MET}}$-formula in (3) is equivalent to $\mathbf{C}_2(p)$. Notice also that the the natural way to render $\mathbf{C}_n(p)$ as an $\mathsf{L}_{\mathrm{MET}}$ formula requires quantifier depth n, consistent with the above-referenced "impossibility result" of [29].

Theorem 7 ([31]). *For every $\mathsf{L}^{(1)}_{\mathrm{MET}}$ formula $\varphi(x)$ there is an equivalent formula $\varphi^\dagger$ in MTL augmented with the unary counting modalities $\mathbf{C}_n$, $n \in \mathbb{N}$, such that $\varphi^\dagger$ mentions only integer constants.*

Future Modalities. Another crucial feature of MTL for obtaining expressive completeness is the presence of past connectives. Recall in this regard that for any sentence φ of monadic first-order logic over the structure $(\mathbb{R}_{\geq 0}, <)$, there is an equivalent LTL formula $\varphi^\dagger$ that uses only future connectives. Here equivalence is considered with respect to the initial semantics and over finitely variable signals. More precisely we have that for any finitely variable signal $\varpi \colon \mathbb{R}_{\geq 0} \to 2^{\mathsf{AP}}$, (i.e., one with finitely many discontinuities in any bounded interval) one has $\varpi \models \varphi$ if, and only if, $\varpi, 0 \models \varphi^\dagger$. [4] The following result, which follows immediately from [14, Proposition 4], shows that the analogous expressive completeness fails for MTL.

Theorem 8. *Over the initial semantics the $\mathsf{L}_{\mathrm{MET}}$ sentence*

$$\varphi = \exists x \exists y \forall z \, (x < y < x + 1 \wedge p(y) \wedge (y < z < x + 1 \rightarrow q(z)))$$

cannot be expressed in MTL using only $\mathbf{U}_I$.

4.4 Satisfiability and Model checking

The satisfiability and model checking problems for MTL are formulated in a similar manner to the corresponding problems for $\mathsf{L}_{\mathrm{MET}}$.

Since the translation from $\mathsf{L}_{\mathrm{MET}}$ to MTL in Theorem 6 is effective, it follows that satisfiability and model checking for MTL are undecidable. Alternatively, one can give a direct proof along the same lines of Theorem 5 (see, e.g., [10]). However a number of expressive and decidable fragments of MTL have been identified. The best-known such fragment, called *Metric Interval Temporal Logic* (MITL), arises by restricting the interval I in the modalities $\mathbf{U}_I$ and $\mathbf{S}_I$ to be non-singular. In particular, the formula

$$\mathbf{G}_{(0,\infty)}(p \leftrightarrow \mathbf{F}_{\{1\}} p) \, ,$$

which features in the undecidability proof of MTL cannot be expressed in MITL.

Both the satisfiability and model checking problems for MITL were shown to be decidable in [7] via an exponential translation of MITL formulas to equivalent timed automata. Combined with the fact that language emptiness for timed automata is in PSPACE one obtains:

[4] As shown in [30] this property fails without the assumption of finite variability.

Theorem 9 ([7]). *The model checking problem for MITL is* EXPSPACE*-complete.*

Another decidable fragment of MTL, called Bounded MTL, arises by restricting the interval I in the modalities $\mathbf{U}_I$ and $\mathbf{S}_I$ to be bounded. While Bounded MTL can express punctual properties, it obviously can only express time-bounded properties. A common extension of MITL and Bounded MTL with an EXPSPACE-complete model checking problem is identified in [15].

The proof that satisfiability and model checking for MTL are undecidable works similarly in the pointwise semantics as in the continuous semantics. However, if one restricts to the future fragment of MTL (that is, keeping $\mathbf{U}_I$ but omitting $\mathbf{S}_I$) then the situation becomes more delicate. While both problems are again undecidable, the proof becomes substantially different.

Consider the formula $\mathbf{G}_{(0,\infty)}(p \leftrightarrow \mathbf{F}_{\{1\}}p)$, which is instrumental in the proof of undecidability of MTL. A timed word satisfies this formula if every p-event is followed by a p-event exactly one time unit later. However, the formula does not require that every p-event be preceded by a p-event one time unit earlier (indeed, one cannot enforce that there be *any* event one time unit earlier). For this reason, a direct encoding of the computations of a Turing machine or 2-counter machine into a language of timed words (as in the undecidability proofs in [6] and [10]) fails for MTL. However one can encode computations of channel machines (finite automata, equipped with an unbounded FIFO memory) with *insertion errors*, that is, channel machines under a semantics in which extra letters may nondeterministically be inserted anywhere in the channel during each transition. Using this idea, [47] shows undecidability of satisfiability for the future fragment of MTL in the pointwise semantics by reduction from the recurrence problem for channel machines with insertion errors, that is, the problem of whether a given channel machine has a computation that visits an accepting control state infinitely often. Naturally, the ability of MTL to express the recurrence property $\mathbf{GF}p$ plays a key role in this proof.

The undecidability result of [47] only works over infinite words. Indeed, it was shown in [46] that both satisfiability and model checking are decidable for the future fragment of MTL over finite timed words. The decision procedure in [46] involves translating an MTL formula into an equivalent alternating timed automaton. Crucially such an automaton requires only a single clock. The main technical result of [46] was to show that the language emptiness problem for one-clock alternating timed automata is decidable. This was done by a method analogous to the region-automaton construction for ordinary timed automata. However in the case of alternating automata this construction does not yield a finite quotient, and [46] relies on the existence of a well-quasi-order (established using Higman's Lemma) on the set of configurations of a given one-clock alternating timed automaton to prove termination of the algorithm for deciding language emptiness.

Theorem 10 ([46]). *The satisfiability and model checking problems for (the future fragment of) MTL over finite timed words are non-primitive recursive.*

Over infinite words, using similar methods, one can identify a safety fragment of MTL for which model checking is decidable [48].

4.5 Timed Propositional Temporal Logic

The logic TPTL [8] is another extension of LTL to the metric setting, this time using so-called *formula clocks*. The formulas of TPTL are given by the following grammar:

$$\varphi ::= p \mid x \sim c \mid \neg\varphi \mid \varphi \wedge \varphi \mid x \cdot \varphi \mid \varphi \,\mathbf{U}\, \varphi \mid \varphi \,\mathbf{S}\, \varphi$$

where $p \in \mathsf{AP}$, x is a formula clock, $\sim \in \{<, >\}$ and $c \in \mathbb{Q}$.

Example 4. As for the case of branching-time, one easily expresses decorated modalities using formula clocks: formula $p \,\mathbf{U}_I\, q$ translates as $x \cdot p \,\mathbf{U}\, (q \wedge x \in I)$, which is in TPTL since I is required to have rational endpoints.

It is easy to see that for every MTL formula there is an equivalent TPTL formula, and for every TPTL formula there is an equivalent $\mathsf{L}_{\mathrm{MET}}$ formula. It immediately follows from Theorem 6 that TPTL with rational constants is expressively complete for $\mathsf{L}_{\mathrm{MET}}$. Similarly, it follows from Theorem 7 that TPTL with integer constants is expressively complete for $\mathsf{L}^{(1)}_{\mathrm{MET}}$. Finally, if we disallow the past operator $\mathbf{S}_I$ in both MTL and TPTL, then the latter is strictly more expressive, since it can express the property of Theorem 8.

5 Conclusion

Timed temporal logics have been defined to express quantitative constraints over delays between events. For instance, one can express the property that any request is answered within some fixed delay. We have first discussed semantic choices: formulas of (linear-time) timed temporal logics can either be interpreted using a discrete-observation setting (only actions are observed), or using a continuous-observation setting (time elapsing in states and changes of states are both observed). While this may seem harmless (though one can easily exhibit examples distinguishing the two semantics), it actually impacts the complexity of model-checking.

In a second part, we have focused on branching-time temporal logics. We have both discussed extensions of CTL and of modal logics. We have explained that the model-checking problem of TCTL over timed automata can be done using a simple extension of standard technics for reachability analysis. We have then turned to timed extensions of HML and have discussed the model-checking problem as well as other properties like compositionality.

In the last part of the paper, we have focused on linear-time, and we have explained the expressive completeness of MTL with respect to the natural metric extension of first-order logic over the reals. We have then discussed the model-checking and the satisfiability problems for (fragments of) MTL, and finished the section with a short discussion on a timed extension of LTL with explicit clock variables.

A short survey cannot be exhaustive on such a wide topic, and there are a number of related results that we could not mention in this paper. We refer e.g.

to [52,22,23,45,49,13,36, to cite only a few] for more results on the very topic developed in this paper. (Linear-time) timed temporal logics have also been used in other domains, e.g. in the prolific domains of monitoring and run-time verification for real-time systems [43]. We refer to [42] for a recent discussion on this problematic.

While timed temporal logics are rather well-understood now, several important questions are still to be investigated. In particular, the satisfiability (or synthesis) problem for timed logics is not fully (or satisfactorily) understood yet. For instance, the synthesis problem for TCTL and L_ν is undecidable, and only a strong assumption on the resources leads to decidability [39]. Similar resource restrictions have to be made [16] to be able to solve the so-called reactive-synthesis problem for MITL (while without restrictions it is shown to be undecidable). Therefore, designing (efficient) algorithms for the synthesis of real-time systems is a real challenge!

References

1. Luca Aceto, Patricia Bouyer, Augusto Burgueño, and Kim Guldstrand Larsen. The power of reachability testing for timed automata. *Theor. Computer Science*, 300(1-3):411–475, 2003.
2. Luca Aceto, Anna Ingólfsdóttir, Mikkel Lykke Pedersen, and Jan Poulsen. Characteristic formulae for timed automata. *RAIRO – Theoretical Informatics and Applications*, 34(6):565–584, 2000.
3. Luca Aceto and François Laroussinie. Is your model checker on time? *Journal of Logic and Algebraic Programming*, 52-53:3–51, 2002.
4. Rajeev Alur. *Techniques for Automatic Verification of Real-Time Systems*. PhD thesis, Stanford University, Palo Alto, California, USA, 1991.
5. Rajeev Alur, Costas Courcoubetis, and David L. Dill. Model-checking in dense real-time. *Inf. & Comp.*, 104(1):2–34, 1993.
6. Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Computer Science*, 126(2):183–235, 1994.
7. Rajeev Alur, Tómas Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, 1996.
8. Rajeev Alur and Thomas A. Henzinger. A really temporal logic. In FOCS'89, p. 164–169. IEEE Comp. Soc. Press, 1989.
9. Rajeev Alur and Thomas A. Henzinger. Logics and models of real time: A survey. In REX'91, LNCS 600, p. 74–106. Springer, 1992.
10. Rajeev Alur and Thomas A. Henzinger. Real-time logics: Complexity and expressiveness. *Inf. & Comp.*, 104(1):35–77, 1993.
11. Henrik Reif Andersen. Partial model-checking (extended abstract). In LICS'95, p. 398–407. IEEE Comp. Soc. Press, 1995.
12. Gerd Behrmann, Alexandre David, Kim Guldstrand Larsen, John Håkansson, Paul Pettersson, Wang Yi, and Martijn Hendriks. Uppaal 4.0. In QEST'06, p. 125–126. IEEE Comp. Soc. Press, 2006.
13. Marcello Maria Bersani, Matteo Rossi, and Pierluigi San Pietro. Deciding the satisfiability of MITL specifications. In GandALF'13, EPTCS 119, p. 64–78, 2013.
14. Patricia Bouyer, Fabrice Chevalier, and Nicolas Markey. On the expressiveness of TPTL and MTL. In FSTTCS'05, LNCS 3821, p. 432–443. Springer, 2005.

15. Patricia Bouyer, Nicolas Markey, Joël Ouaknine, and James Worrell. The cost of punctuality. In LICS'07, p. 109–118. IEEE Comp. Soc. Press, 2007.
16. Thomas Brihaye, Morgane Estiévenart, Gilles Geeraerts, Hsi-Ming Ho, Benjamin Monmege, and Nathalie Sznajder. Real-time synthesis is hard! In *Proceedings of the 14th International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS'16)*, LNCS 9884, p. 105–120. Springer, 2016.
17. Thomas Brihaye, François Laroussinie, Nicolas Markey, and Ghassan Oreiby. Timed concurrent game structures. In CONCUR'07, LNCS 4703, p. 445–459. Springer, 2007.
18. Kārlis Čerāns, Jens Christian Godskesen, and Kim Guldstrand Larsen. Timed modal specification - theory and tools. In CAV'93, LNCS 697, p. 253–267. Springer, 1993.
19. Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In LOP'81, LNCS 131, p. 52–71. Springer, 1982.
20. Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.
21. Conrado Daws, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. The tool KRONOS. In HSCC'95, LNCS 1066, p. 208–219. Springer, 1996.
22. Deepak D'Souza and Pavithra Prabhakar. On the expressiveness of MTL in the pointwise and continuous semantics. *International Journal on Software Tools for Technology Transfer*, 9(1):1–4, 2007.
23. Carlo A. Furia and Matteo Rossi. On the expressiveness of MTL variants over dense time. In FORMATS'07, LNCS 4763, p. 163–178. Springer, 2007.
24. Dov M. Gabbay. Expressive functional completeness in tense logic. In *Aspects of Philosophical Logic*, Synthese Library 147, p. 91–117. Springer, 1981.
25. Dov M. Gabbay, Amir Pnueli, Saharon Shelah, and Jonathan Stavi. On the temporal analysis of fairness. In POPL'80, p. 163–173. ACM Press, 1980.
26. Jens Christian Godskesen, Kim Guldstrand Larsen, and Arne Skou. Automatic verification of real-time systems using epsilon. In PSTV'94, IFIP Conference Proceedings 1, p. 323–330. Chapman & Hall, 1995.
27. Klaus Havelund, Arne Skou, Kim Guldstrand Larsen, and Kristian Lund. Formal modelling and analysis of an audio/video protocol: An industrial case study using Uppaal. In RTSS'97, p. 2–13. IEEE Comp. Soc. Press, 1997.
28. Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. HyTech: A model-checker for hybrid systems. *International Journal on Software Tools for Technology Transfer*, 1(1-2):110–122, 1997.
29. Yoram Hirshfeld and Alexander Rabinovich. Expressiveness of metric modalities for continuous time. *Logical Methods in Computer Science*, 3(1), 2007.
30. Yoram Hirshfeld and Alexander Moshe Rabinovich. Future temporal logic needs infinitely many modalities. *Inf. & Comp.*, 187(2):196–208, 2003.
31. Paul Hunter. When is metric temporal logic expressively complete? In CSL'13, LIPIcs 23, p. 380–394. Leibniz-Zentrum für Informatik, 2013.
32. Paul Hunter, Joël Ouaknine, and James Worrell. Expressive completeness for metric temporal logic. In LICS'13, p. 349–357. IEEE Comp. Soc. Press, 2013.
33. Samy Jaziri, Kim G. Larsen, Radu Mardare, and Bingtian Xue. Adequacy and complete axiomatization for timed modal logic. In *Proceedings of the 30th Conference on Mathematical Foundations of Programming Semantics (MFPS'14)*, ENTCS 308, p. 183–210. Elsevier Science Publishers, 2014.

34. Johan Anthony Willem Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, Computer Science Department, University of California at Los Angeles, USA, 1968.
35. Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.
36. Shankara Narayanan Krishna, Khushraj Madnani, and Paritosh K. Pandya. Metric temporal logic with counting. In FoSSaCS'16, LNCS 9634, p. 335–352. Springer, 2016.
37. Salvatore La Torre and Margherita Napoli. A decidable dense branching-time temporal logic. In FSTTCS'00, LNCS 1974, p. 139–150. Springer, 2000.
38. François Laroussinie and Kim Guldstrand Larsen. CMC: A tool for compositional model-checking of real-time systems. In FORTE/PSTV'98, IFIP Conference Proceedings 135, p. 439–456. Kluwer Academic, 1998.
39. François Laroussinie, Kim Guldstrand Larsen, and Carsten Weise. From timed automata to logic – and back. In MFCS'95, LNCS 969, p. 529–539. Springer, 1995.
40. Kim Guldstrand Larsen. Modal specifications. In AVMFSS'89, LNCS 407, p. 232–246. Springer, 1990.
41. Kim Guldstrand Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1-2):134–152, 1997.
42. Oded Maler. Some thoughts on runtime verification. In *Proceedings of the 16th International Conference on Runtime verification (RV'16)*, LNCS 10012, p. 3–14. Springer, 2016.
43. Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *Proceedings of the Joint Conference on Formal Modelling and Analysis of Timed Systems and Formal Techniques in Real-Time and Fault Tolerant System (FORMATS+FTRTFT'04)*, Lectute Notes in Computer Science 3253, p. 152–166. Springer, 2004.
44. Robin Milner. *Communication and concurrency*, Prentice Hall International Series in Computer Science. Prentice Hall Int., 1989.
45. Joël Ouaknine, Alexander Rabinovich, and James Worrell. Time-bounded verification. In CONCUR'09, LNCS 5710, p. 496–510. Springer, 2009.
46. Joël Ouaknine and James Worrell. On the decidability of metric temporal logic. In LICS'05, p. 188–197. IEEE Comp. Soc. Press, 2005.
47. Joël Ouaknine and James Worrell. On metric temporal logic and faulty Turing machines. In FoSSaCS'06, LNCS 3921, p. 217–230. Springer, 2006.
48. Joël Ouaknine and James Worrell. Safety metric temporal logic is fully decidable. In TACAS'06, LNCS 3920, p. 411–425. Springer, 2006.
49. Paritosh K. Pandya and Simoni S. Shah. On expressive powers of timed logics: Comparing boundedness, non-punctuality, and deterministic freezing. In CONCUR'11, LNCS 6901, p. 60–75. Springer, 2011.
50. Amir Pnueli. The temporal logic of programs. In FOCS'77, p. 46–57. IEEE Comp. Soc. Press, 1977.
51. Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in CESAR. In SOP'82, LNCS 137, p. 337–351. Springer, 1982.
52. Jean-François Raskin. *Logics, Automata and Classical theories for Deciding Real Time*. Thèse de doctorat, FUNDP, Namur, Belgium, 1999.
53. Pierre Wolper, Moshe Y. Vardi, and A. Prasad Sistla. Reasoning about infinite computation paths. In FOCS'83, p. 185–194. IEEE Comp. Soc. Press, 1983.
54. Wang Yi. CCS + time = an interleaving model for real time systems. In ICALP'91, LNCS 510, p. 217–228. Springer, 1991.