Timed Modal Logics for Specifying and Verifying Real-Time Systems

François Laroussinie

Lab. Spécification & Vérification
ENS de Cachan & CNRS UMR 8643
61, av. Pdt. Wilson, 94235 Cachan Cedex France
email: `fl@lsv.ens-cachan.fr`

Abstract. The timed modal logic L_ν has been proposed in order to express timed properties over real-time systems modeled as (compositions of) timed automata. In this paper, we present a short survey of results about L_ν: complexity of model checking, expressivity, compositional methods, relationship with strong timed bisimulation etc. We also show how L_ν can be extended in order to express new properties.

1 Introduction

Model checking is widely used for the design and debugging of critical reactive systems [Eme90,CGP99]. During the last fifteen years, it has been extended to *real-time systems*, where quantitative information about time is required.

Timed models. Real-time model checking has been mostly studied and developed in the framework of Alur and Dill's *Timed Automata* (TA) [ACD93,AD94], *i.e.* automata extended with *clocks* that progress synchronously with time. The behavior of a real-time system can often be modeled as a parallel composition of TA. There now exists a large body of theoretical knowledge and practical experience for this class of systems.

Timed specifications. Temporal and modal logics provide a fundamental framework for formally specifying systems and reasoning about them [Eme90,MP92]. For example, a property like "any problem is followed by an alarm" can be easily expressed with most of the temporal logics (*CTL*, *LTL*, μ-calculus, etc.).

When specifying real-time systems, it is necessary to express *timed property* like:

"Any problem is followed by an alarm <u>in at most 10 time units</u>." (1)

In order to express timing aspects of computations, we can extend the classical temporal or modal logics. A first possibility is to use timing constraints tagging temporal modalities [AH92]. For example, with the timed version of *CTL*, namely *TCTL*, one can specify the property (1) as follows:

$$\mathsf{AG}\Big(\mathsf{problem} \;\Rightarrow\; \mathsf{AF}_{\leq 10}\,\mathsf{alarm}\Big)$$

A more expressive method consists in adding clocks – a.k.a. freeze variables – in the specification language [AH94,BCM05]. In this framework, a formula clock (with the $\underline{\texttt{in}}$ operator) can be reset and compared with some constant later on. For example, the previous property can be written as follows:

$$\mathsf{AG}\Big(\mathsf{problem} \;\Rightarrow\; x \;\underline{\texttt{in}}\; \mathsf{AF}(x \leq 10 \,\wedge\, \mathsf{alarm})\Big)$$

The same extensions can be done for modal logics with fixpoint operators.

Outline. In this paper, we consider the timed modal logic L_ν that has been introduced in [LLW95]. It is a natural extension of classical modal logics for finite labeled transition systems with (1) two modalities $\langle\delta\rangle$ and $[\delta]$ in order to deal with delay transitions of the system to be specified (in addition to the standard modalities for action transitions), (2) with formula clocks and (3) maximal fixpoint operators.

We address complexity questions for model checking and present several properties about the expressive power of L_ν. This paper contains results from the following papers [AL02,LL95,LLW95,LL98,BCL05a].

2 Definitions

We first define the *timed automata* proposed by Alur and Dill [ACH94] and then we introduce the timed modal logic L_ν.

2.1 Timed Automata

Notations. Let Act be a finite set of *actions*, and let $\mathbb{N}$ and $\mathbb{R}_+$ denote the sets of natural and non-negative real numbers, respectively. Let X be a set of clocks. A *clock valuation* for X is a function from X to $\mathbb{R}_+$, the set of valuations is denoted $\mathbb{R}_+^X$. Given a valuation $v \in \mathbb{R}_+^X$ and $t \in \mathbb{R}_+$, $v+t$ is a valuation assigning $v(x)+t$ for any $x \in X$. Let $r \subseteq X$, $v[r \leftarrow 0]$ be the valuation assigning 0 (resp. $v(x)$) for any $x \in r$ (resp. $x \in X \setminus r$).

We use $\mathcal{C}(X)$ to denote the set of *clock constraints* defined as the boolean combinations of atomic constraints of the form $x \sim p$ or $x - y \sim p$, with $x, y \in X$, $p \in \mathbb{N}$, and $\sim \in \{<, \leq, >, \geq, =\}$. Given $g \in \mathcal{C}(X)$ and $v \in \mathbb{R}_+^X$, we write $v \models g$ when v satisfies g. We write $\mathcal{C}_{\lessdot}(X)$ for the restriction of $\mathcal{C}(X)$ to the positive combinations of constraints $x \leq p$ or $x < p$, with $p \in \mathbb{N}$.

Definition 2.1. *A* timed automaton (TA) *is a tuple $\mathcal{A} = (L, \ell_0, \mathsf{Act}, X, \mathsf{Inv}, T)$ where L is a finite set of locations, $\ell_0 \in L$ is the initial location, X is a finite set of clocks, $\mathsf{Inv} : L \rightarrow \mathcal{C}_{\lessdot}(X)$ is a function that assigns an invariant to each location, and $T \subseteq L \times \mathcal{C}(X) \times \mathsf{Act} \times 2^X \times L$ is a finite set of edges. A quintuple*

$(\ell, g, a, r, \ell') \in T$ — denoted by $\ell \xrightarrow{g,a,r} \ell'$ — represents an edge from location ℓ to location ℓ' with action a, g is the guard *and r is a set of clocks to be reset to* 0.

A guard is used to specify when a transition can be performed. An invariant is used to avoid excessive time delays in a location and thus it may enforce action transitions to be performed.

The semantics of TA is defined as a *Timed Transition System* (TTS), that is a tuple $S = (Q, q_0, \mathsf{Act}, \rightarrow_S)$ where Q is a set of states, $q_0 \in Q$ is the initial state, and $\rightarrow_S \subseteq Q \times (\mathsf{Act} \cup \mathbb{R}_+) \times Q$ is a set of transitions (we write $q \xrightarrow{e}_S q'$ when $(q, e, q') \in \rightarrow_S$). The transitions labeled by $a \in \mathsf{Act}$ (resp. $t \in \mathbb{R}_+$) are called *action* (resp. *delay*) transitions. We make the following common assumptions about delay transitions in TTSs:

- 0-delay: $q \xrightarrow{0}_S q'$ if and only if $q = q'$,
- Time-Additivity: if $q \xrightarrow{t}_S q'$ and $q' \xrightarrow{t'}_S q''$ with $t, t' \in \mathbb{R}_+$, then $q \xrightarrow{t+t'}_S q''$,
- Time-Continuity: if $q \xrightarrow{t}_S q'$, then $\forall t', t'' \in \mathbb{R}_+$ with $t = t' + t''$, there exists q'' such that $q \xrightarrow{t'}_S q'' \xrightarrow{t''}_S q'$,
- Time-determinism: if $q \xrightarrow{t}_S q'$ and $q \xrightarrow{t}_S q''$ with $t \in \mathbb{R}_+$, then $q' = q''$.

Standard notions of finite or infinite runs apply to TTS.

Given a TA $\mathcal{A} = (L, \ell_0, \mathsf{Act}, X, \mathsf{Inv}, T)$, we define its semantics as the TTS $S_{\mathcal{A}} = (L \times \mathbb{R}_+^X, (\ell_0, v_0), \mathsf{Act}, \rightarrow_{S_{\mathcal{A}}})$ where $v_0(x) = 0$ for all $x \in X$ and $\rightarrow_{S_{\mathcal{A}}}$ consists of:

1. action transition: $(\ell, v) \xrightarrow{a}_{S_{\mathcal{A}}} (\ell', v')$ if there exists an edge $\ell \xrightarrow{g,a,r} \ell'$ in T s.t. $v \models g$, $v' = v[r \leftarrow 0]$ and $v' \models \mathsf{Inv}(\ell')$;
2. delay transitions: $(\ell, v) \xrightarrow{t}_{S_{\mathcal{A}}} (\ell, v')$ if $t \in \mathbb{R}_+$, $v' = v + t$ and[1] $v' \models \mathsf{Inv}(\ell)$.

Then a *state* (or *configuration*) of a timed automaton A is a pair (ℓ, v), where ℓ is a location – a control state – and v is a clock valuation for X. A key point (for decidability) is the synchronous time elapsing: all clocks have the same speed.

The size of a TA is $|L| + |X| + \sum_{(\ell,g,a,r,\ell') \in T} |g| + \sum_\ell |\mathsf{Inv}(\ell)|$ where the size of a constraint is its length (constants are encoded in binary).

Networks of timed automata We model real time systems as parallel compositions of timed automata with n-ary synchronization functions. Let $\mathcal{A}_1, \ldots, \mathcal{A}_n$ be n timed automata. A *synchronization function* f is a partial function $(\mathsf{Act} \cup \{\bullet\}) \times \ldots \times (\mathsf{Act} \cup \{\bullet\}) \rightarrow \mathsf{Act}$, where $\bullet$ denotes a distinguished no–action symbol. Note that f is a synchronization function with renaming. We write $(\mathcal{A}_1 | \ldots | \mathcal{A}_n)_f$ for the parallel composition of $\mathcal{A}_1, \ldots, \mathcal{A}_n$ w.r.t. function f. A network configuration is a pair $\langle \overline{\ell}, v \rangle$ where $\overline{\ell} = (\ell_1, \ldots, \ell_n)$ is a vector of locations and v is a valuation for $X = \cup_i X_i$, i.e. the clocks of the network (we denote by v_i the restriction $v_{|X_i}$).

[1] due to the definition of the invariants, this entails: $v + t' \models \mathsf{Inv}(\ell)$ for any $0 \leq t' \leq t$.

The semantics of $(\mathcal{A}_1|\dots|\mathcal{A}_n)_f$ can be defined as a TTS whose states are the configurations of the network and the transitions are given by the two following rules:

$$\begin{array}{lll} \langle \overline{\ell}, v\rangle \xrightarrow{t} \langle \overline{\ell}, v+t\rangle & \text{iff} & \forall i \in \{1,\dots,n\}, \langle \ell_i, v_i\rangle \xrightarrow{t} \langle \ell_i, v_i+t\rangle \\ \langle \overline{\ell}, v\rangle \xrightarrow{b} \langle \overline{\ell'}, v'\rangle & \text{iff} & \forall i \in \{1,\dots,n\}, a_i \in \mathsf{Act} \text{ implies } \langle \ell_i, v_i\rangle \xrightarrow{a_i} \langle \ell'_i, v'_i\rangle \text{ and} \\ & & a_i = \bullet \text{ implies } (\ell'_i = \ell_i \ \wedge\ v'_i = v_i) \text{ and} \\ & & v' = v'_1 \dots v'_n \text{ and } f(a_1,\dots,a_n) = b \end{array}$$

Note that the parallel composition does not add expressive power: from any parallel composition, one can build an *equivalent* (i.e. strongly bisimilar, see Section 4.1) timed automaton.

2.2 The Timed Modal Logic L_ν

We now define L_ν, a timed modal μ-calculus [LLW95]:

Definition 2.2. *Let K be a finite set of clocks (disjoint from X), and* Id *be a countably infinite set of identifiers (ranged over by X, Y). The set L_ν of formulae over Act, K and* Id *is generated by the following grammar:*

$$\begin{array}{l} \varphi ::= \mathtt{tt} \ \mid\ \mathtt{ff} \ \mid\ \varphi \wedge \varphi \ \mid\ \varphi \vee \varphi \ \mid\ r \ \underline{\mathtt{in}}\ \varphi \ \mid\ g \mid\ [a]\,\varphi \ \mid\ \langle a\rangle\,\varphi \ \mid \\ \qquad [\delta]\ \varphi \ \mid\ \langle\delta\rangle\,\varphi \ \mid\ Z \end{array}$$

where $a \in$ Act, $g \in \mathcal{C}(K)$, $r \subseteq K$ and $Z \in$ Id.

The meaning of the identifiers is specified by a declaration $\mathcal{D}$ assigning an L_ν formula to each identifier in order to define properties with maximal fixpoints. Let $\mathcal{A}$ be a TA or a parallel composition of TA. We interpret L_ν formulae over extended states (ℓ, v, u) where (ℓ, v) is a configuration of A and u is a clock valuation for K. This satisfaction relation $\models_{\mathcal{A},\mathcal{D}}$ is defined as the largest relation satisfying the implications in Table 1 (for $\models_{\mathcal{A},\mathcal{D}}$ the implications are bi-implications). The modalities $\langle e\rangle$ with $e \in \mathsf{Act} \cup \{\delta\}$ correspond to existential quantification over action or delay transitions, and $[e]$ is the counterpart for universal quantification. An extended state satisfies an identifier Z if it belongs to the maximal fixpoint of the equation $Z =_\nu \mathcal{D}(Z)$. Finally the formula clocks are used to measure time elapsing in properties. We write $A \models \varphi$ for $(\ell_0, v_0, u_0) \models \varphi$ where $u_0(x) = 0$ for all $x \in K$.

We can use L_ν formula to express classical temporal (and timed) properties:

- $z\ \underline{\mathtt{in}}\ \langle\delta\rangle\,(z < 10 \wedge (\langle a\rangle\,\mathtt{tt} \vee \langle b\rangle\,\mathtt{tt}))$ specifies that an a-action or a b-action is enabled before 10 time units (without performing any action transition).
- We can express that some formula φ holds for any reachable state ("$\mathsf{ALWAYS}_(\varphi)$", corresponding to the AG operator from *CTL*). This can be defined by the following equation: $Z \overset{\text{def}}{=} \varphi \wedge \bigwedge_{a \in \mathsf{Act}} [a]\,Z \wedge [\delta]\,Z$.

$$
\begin{array}{l}
(\ell,v,u) \models \mathtt{tt} \implies \mathtt{tt} \\
(\ell,v,u) \models \mathtt{ff} \implies \mathtt{ff} \\
(\ell,v,u) \models g \implies u \models g \\
(\ell,v,u) \models \varphi_1 \vee \varphi_2, \implies (\ell,v,u) \models \varphi_1 \quad \textit{or} \quad (\ell,v,u) \models \varphi_2 \\
(\ell,v,u) \models \varphi_1 \wedge \varphi_2, \implies (\ell,v,u) \models \varphi_1 \quad \textit{and} \quad (\ell,v,u) \models \varphi_2 \\
(\ell,v,u) \models r \ \underline{\mathtt{in}}\ \varphi \implies (\ell,v,u[r \leftarrow 0]) \models \varphi \\
(\ell,v,u) \models [a]\,\varphi \implies \text{for all } (\ell,v) \xrightarrow{a}_{S_{\mathcal{A}}} (\ell',v'),\ (\ell',v',u) \models \varphi \\
(\ell,v,u) \models \langle a\rangle\,\varphi \implies \text{there is some } (\ell,v) \xrightarrow{a}_{S_{\mathcal{A}}} (\ell',v'),\ (\ell',v',u) \models \varphi \\
(\ell,v,u) \models [\delta]\,\varphi \implies \text{for all } t \in \mathbb{R}_+ \text{ s.t. } (\ell,v) \xrightarrow{t}_{S_{\mathcal{A}}} (\ell,v+t),\ (\ell,v+t,u+t) \models \varphi \\
(\ell,v,u) \models \langle\delta\rangle\,\varphi \implies \text{there is some } t \in \mathbb{R}_+ \text{ s.t. } (\ell,v) \xrightarrow{t}_{S_{\mathcal{A}}} (\ell,v+t),\ (\ell,v+t,u+t) \models \varphi \\
(\ell,v,u) \models Z \implies (\ell,v,u) \models \mathcal{D}(Z)
\end{array}
$$

Table 1. Satisfaction implications for L_ν

- Given two subsets of events Act_1 and Act_2, we can state that any event in Act_1 is followed by an event in Act_2 in less than $\varDelta$ time units (assume $\varDelta \in \mathbb{N}$):

$$
\begin{array}{l}
Z_1 \stackrel{\text{def}}{=} \bigwedge_{a\in\mathsf{Act}_1}[a]\,(y\ \underline{\mathtt{in}}\ Z_2) \wedge \bigwedge_{a\in\mathsf{Act}\setminus\mathsf{Act}_1}[a]\,Z_1 \wedge [\delta]\,Z_1 \\
Z_1 \stackrel{\text{def}}{=} (y < \varDelta) \wedge \bigwedge_{a\in\mathsf{Act}_2}[a]\,Z_1 \wedge \bigwedge_{a\in\mathsf{Act}\setminus\mathsf{Act}_2}[a]\,Z_2 \wedge [\delta]\,Z_2
\end{array}
$$

- To express that some property φ will hold during at least $\varDelta$ time units, whatever the transitions performed ($\varphi\mathsf{UpTo}\ \varDelta$), we can use the L_ν formula $z\ \underline{\mathtt{in}}\ Z_1$ with:

$$
Z_1 \stackrel{\text{def}}{=} (z > \varDelta) \vee \Big(\varphi \wedge \bigwedge_{a\in\mathsf{Act}} [a]\,Z_1 \wedge [\delta]\,Z_1\Big)
$$

3 Model checking

Region graph technique. Automatic verification of timed systems modelled as (networks of) timed automata is possible despite the uncountably infinite number of states associated with the semantics of a timed automaton. The decision procedure for the problem $A \models \varphi$ is based on the well known *region technique* (see [ACD93] for a description for *TCTL*). Given A and φ, it is possible to partition the uncountably infinite set of time assignments over $X \cup K$ into a finite number of *regions*, s.t. two extended configurations (ℓ,v,u) and (ℓ,v',u'), where uv and $u'v'$ are in the same region, satisfy the same subformulae of φ. Let c_{max} be the maximal constant occurring in the guards and invariants of A and in the formula φ. The regions can be defined as the equivalence classes induced by the equivalence relation over valuations defined as follows: two valuations $w, w' \in \mathbb{R}_+^{X\cup K}$ are in the same region iff they satisfy the same clock constraints from the set $\mathcal{C}_{c_{max}}(X \cup K)$ containing $\mathcal{C}(X \cup K)$ expressions whose integer constants belong to $\{0, \ldots, c_{max}\}$. Due to this bound over the constants, the number of equivalence classes is clearly finite.

Then we can define a symbolic semantics [LLW95] for A over a finite transition system, called the *region graph*, whose states are pairs (ℓ, γ): a location and a region. And formulae in L_ν can be interpreted over the states of the region graph.

Thus the region graph technique provides decidability results for many verification problems over timed systems because it allows us to reduce a question over an infinite transition system to a problem over a finite one. But note that the number of regions is in $O(|X \cup K|! \cdot c_{max}^{|X \cup K|})$ and the size of the region graph is in $O((|L| + |T|) \cdot |X \cup K|! \cdot c_{max}^{|X \cup K|})$. Then in practice, the region graph is not built in timed model checkers. and in the tools like Uppaal[LPY97] or Kronos[Yov97], the algorithms are based on an efficient data-structure, the DBMs – Differences Bounded Matrices – that allow to handle convex sets of valuations. Moreover several heuristics have been developed to improve the efficiency of the algorithms (on-the-fly algorithms, reduction of the number of clocks, . . .).

Verification of L_ν properties. Model checking L_ν over networks of timed automata is then decidable:

Theorem 3.1 ([AL02]). *Model checking problem for L_ν is EXPTIME-complete. The specification and program complexities are also EXPTIME-complete.*

The EXPTIME membership comes from the fact that applying standard verification algorithms for modal logics over the region graph can be done in time linear in the size of φ and the size of the region graph, this provides an algorithm which is exponential in $|\mathcal{A}|$ and $|\varphi|$.

The EXPTIME-hardness is proved by reducing the acceptance of a word w by a linear (space) bounded alternating Turing machine $\mathcal{M}$ to a model checking problem for L_ν. We assume (w.l.o.g.) that there is a strict alternation of existential and universal states in $\mathcal{M}$ and that the initial state is existential. First one can build a timed automaton $\mathcal{A}_{\mathcal{M},w}$ that represents the behavior of the non-alternating version of $\mathcal{M}$ over the word w (clocks are used to encode the contents of the $|w|$ cells of the tape): any action transition corresponds to a step of $\mathcal{M}$ and between two actions one require a strictly positive delay. We distinguish three labels for actions of $\mathcal{M}$: Init corresponds to the writing of w on the tape at the beginning of the computation, a labels any step of $\mathcal{M}$ and Accepting labels accepting states of $\mathcal{M}$. Secondly one can use the L_ν formula $\Phi_{na} \overset{\text{def}}{=} [\delta]\, [\mathsf{Init}]\, Y$ with :

$$Y \overset{\text{def}}{=} [\mathsf{Accepting}]\, \mathtt{ff} \wedge [\delta]\, [\mathsf{a}]\, (\langle\delta\rangle\, \langle\mathsf{a}\rangle\, Y)$$

Then the formula Φ_{na} holds for the initial state of $\mathcal{A}_{\mathcal{M},w}$ iff $\mathcal{M}$ does not accept w. See [AL02] for the full proof.

The main difference with reachability or *TCTL* model checking (which are PSPACE-complete problems for TA) is the ability for L_ν to simulate the alternating behavior of $\mathcal{M}$.

From the previous construction, one can easily deduce that the program complexity is also EXPTIME-hard since the formula used in the reduction does not depend on $\mathcal{M}$ and w.

Finally the specification complexity comes from the ability of L_ν to encode the behavior of timed automata. The previous reduction could have been done for a simple automaton with no clock and no edge and with a more complex formula. This can be obtained as a direct consequence of the properties of L_ν presented in the next section.

Satisfiability. The decidability of the satisfiability problem — given an L_ν formula Φ, does there exists a timed automaton $\mathcal{A}$ s.t. $\mathcal{A} \models \Phi$ — is still an opened problem [LLW95]. Note that for most of the timed branching-time temporal logics ($TCTL$, T_μ) the satisfiability problem is undecidable [ACD93,AFH96,HNSY94]. For the timed linear-time temporal logics like MTL, satisfiability is undecidable for the (standard) interval-based semantics [HNSY94], but is decidable for pointwise semantics [OW05].
Finally note that if we bound the *resources* of $\mathcal{A}$ — its number of clocks and the maximal constant occurring in its guards — the problem is decidable for L_ν [LLW95].

4 Expressivity

4.1 Strongly timed bisimilar

The standard notion of bisimulation [Mil89,Par81] can be naturally extended to timed systems : let S_A and S_B two TTSs and and Q_A and Q_B be their set of states, the *strong timed bisimulation* $\sim$ is defined as the largest symmetric relation over $Q_A \times Q_B$ such that whenever $q_A \sim q_B$, we have (1) for any $q_A \xrightarrow{a} q'_A$ with $a \in \mathsf{Act}$, there exists $q_B \xrightarrow{a} q'_B$ and $q'_A \sim q'_B$, and (2) for any $q_A \xrightarrow{t} q'_A$ with $t \in \mathbb{R}_+$, there exists $q_B \xrightarrow{t} q'_B$ and $q'_A \sim q'_B$.

We say that two TA $\mathcal{A}_1$ and $\mathcal{A}_2$ are *strong timed bisimilar* if their corresponding TTSs are strong bisimilar. In the following we show that some L_ν can express strong timed bisimilarity of timed automata.

Let $\mathcal{A}_1 = (L^1, \ell_0^1, \mathsf{Act}, X_1, \mathsf{Inv}_1, T_1)$ and $\mathcal{A}_2 = (L^2, \ell_0^2, \mathsf{Act}, X_2, \mathsf{Inv}_2, T_2)$ be two timed automata. Let σ be the binary synchronization function defined as follows : $\forall a \in \mathsf{Act}$, we have $\sigma(a, \bullet) = a_1$ and $\sigma(\bullet, a) = a_2$. With σ, the action transitions of the TA are not synchronized and they are tagged by the number of the active TA. But the delay transitions are synchronized : every clock of the composition progresses synchronously with time.

Now assume $\mathcal{A}_1$ and $\mathcal{A}_2$ have no invariant (for any location ℓ, we have $\mathsf{Inv}_i(\ell) = \mathtt{tt}$). Then we have: $\mathcal{A}_1 \sim \mathcal{A}_2$ iff $(\mathcal{A}_1|\mathcal{A}_2)_\sigma \models Z$ with:

$$Z \overset{\text{def}}{=} \bigwedge_{a \in \mathsf{Act}} \Big([a_1] \, \langle a_2 \rangle \, Z \wedge [a_2] \, \langle a_1 \rangle \, Z \Big) \wedge [\delta] \, Z$$

Indeed the definition of Z is precisely the definition of the strong timed bisimilarity.

When the automata contain invariants, the parallel composition $(\mathcal{A}_1|\mathcal{A}_2)_\sigma$ does not contain all the behaviors of $\mathcal{A}_1$ and $\mathcal{A}_2$: from a configuration $(\ell_1, \ell_2, v_1 v_2)$ of $(\mathcal{A}_1|\mathcal{A}_2)_\sigma$, a delay transition $\xrightarrow{t}$ is enabled iff it is enabled for both automata. If ℓ_1 has an invariant, the parallel composition does not contain delays that violate this invariant even if such delays exist in $\mathcal{A}_2$. The solution consists in defining new TA $\mathcal{A}'_i$'s with the same locations L^i, the same clocks X_i, the set of actions $\mathsf{Act} \cup \{\mathsf{InvFail}\}$, but without invariant and with another set of edges T'_i defined as follows:

- For any location $\ell \in L^i$, we add a transition $(\ell, \neg\mathsf{Inv}(\ell), \mathsf{InvFail}, \emptyset, \ell)$ to T'_i.
- For any transition (ℓ, g, a, r, ℓ') in T_i, we add a transition $(\ell, g \wedge \mathsf{Inv}(\ell) \wedge (\mathsf{Inv}(\ell')[r \leftarrow 0]), r, \ell')$ to T'_i. Where $g[r \leftarrow 0]$ denotes the constraint g where every occurrence of the clocks in r are replaced by the constant 0.

The first kind of transitions is used to mark states which are not reachable in $\mathcal{A}_i$ because the current invariant is violated. The second kind of transitions corresponds to original action steps but we require in the guard that the current invariant has to be satisfied and also that the invariant of the target location has to be satisfied by the valuation after the reset: indeed we want to deal only with transitions that are actually enabled.

Finally we have: $\mathcal{A}_1 \sim \mathcal{A}_2$ iff $(\mathcal{A}'_1|\mathcal{A}'_2)_\sigma \models Z$ where Z is defined as above.

4.2 Compositionality

Compositional model checking [And95] is possible for L_ν: given a parallel composition $(\mathcal{A}_1| \dots |\mathcal{A}_n)_f$ and some L_ν formula Φ, one can build a *quotient* formula $\Phi/\mathcal{A}_1$ s.t. $(\mathcal{A}_1| \dots |\mathcal{A}_n)_f \models \Phi$ iff $(\mathcal{A}_2| \dots |\mathcal{A}_n)_f \models \Phi/\mathcal{A}_1$. The formula $\Phi/\mathcal{A}_1$ integrates the initial property and the pertinent part (w.r.t. Φ) of the behavior of $\mathcal{A}_1$. By repeatedly quotienting components from the network into the formula, we will finally be faced with the verification problem: $\mathit{nil} \models \Phi/\mathcal{A}_1/\dots/\mathcal{A}_n$ where nil is an untimed automaton unable to perform action transitions.

Table 2 presents the definition of quotienting for L_ν [LL95,LL98]. Note that the quotient is defined for a formula, a location and a synchronization function f. Moreover the clocks of the quotiented automaton become formula clocks in the quotient formula. Finally any fixpoint variable of Φ may give rise to $|L|$ variables in $\Phi/\mathcal{A}$.

We have the following theorem:

Theorem 4.1 ([LL98]). *Given two timed automata $\mathcal{A}_1$ and $\mathcal{A}_2$, a synchronization function f and an L_ν formula φ, we have the following property for any configuration $(\ell_1, \ell_2, v_1 v_2)$ of $(\mathcal{A}_1|\mathcal{A}_2)_f$ and $u \in \mathbb{R}_+^K$:*

$(\varphi_1 \wedge \varphi_2)/\ell = (\varphi_1/\ell) \wedge (\varphi_2/\ell)$	$Z/\ell = Z^{\ell}$
$(\varphi_1 \vee \varphi_2)/\ell = (\varphi_1/\ell) \vee (\varphi_2/\ell)$	
$\Big(\langle a\rangle\, \varphi\Big)/\ell = \bigvee_{\ell \xrightarrow{g,b,r} \ell' \text{s.t.} f(b,c)=a} g \wedge \langle c\rangle \Big(r \;\underline{\mathtt{in}}\; (\mathsf{Inv}(\ell') \wedge \varphi/\ell')\Big)$	
$\Big([a]\, \varphi\Big)/\ell = \bigwedge_{\ell \xrightarrow{g,b,r} \ell' \text{s.t.} f(b,c)=a} g \Rightarrow [c] \Big(r \;\underline{\mathtt{in}}\; (\mathsf{Inv}(\ell') \Rightarrow \varphi/\ell')\Big)$	
$\langle\delta\rangle\, \varphi/\ell = \langle\delta\rangle \Big(\mathsf{Inv}(\ell) \wedge \varphi/\ell\Big)$	$[\delta]\, \varphi/\ell = [\delta] \Big(\mathsf{Inv}(\ell) \Rightarrow \varphi/\ell\Big)$
$(x + c \bowtie y + d)/\ell = (x + c \bowtie y + d)$	$(x \;\underline{\mathtt{in}}\; \varphi)/\ell = x \;\underline{\mathtt{in}}\; \Big(\varphi/\ell\Big)$

Table 2. Quotient construction.

$$\Big(((\ell_1, \ell_2), v_1 v_2, u) \models \varphi\Big) \quad \textit{iff} \quad (\ell_2, v_2, u v_1) \models \varphi/\ell_1$$

This technique allows us to avoid the construction of the (exponential) construction of the product corresponding to the parallel composition $(\mathcal{A}_1|\ldots|\mathcal{A}_n)_f$, but of course this complexity is reported in the formula: the size of the quotient formula $\Phi/\mathcal{A}_1$ is in $O(|\Phi|\cdot|\mathcal{A}_1|)$. In order to be applied, the compositional method needs to be completed with reductions: after every quotienting operation, we apply reduction laws in order to keep the size of the formula as small as possible (see [LL98] for a description of these reductions).

The tool CMC (Compositional Model Checker)[2] implements this method for the verification of L_ν properties for TA.

This technique has also been extended to linear hybrid systems [CL00].

4.3 Characteristic properties of timed automata

Given a timed automaton $\mathcal{A} = (L, \ell_0, \mathsf{Act}, X, \mathsf{Inv}, T)$, it is possible to build an L_ν formula $\Phi_{\mathcal{A}}$ that precisely characterizes the behavior of $\mathcal{A}$: a timed automaton is strongly timed-bisimilar to $\mathcal{A}$ iff it satisfies $\Phi_{\mathcal{A}}$. This construction can be done directly from the definition of $\mathcal{A}$ [LLW95] or it can be seen as a consequence of the ability to express timed bisimilarity and the compositionality. Indeed consider the variable Z defined as above to express strong bisimilarity, then the formula Z^{ℓ_0} defined as the quotient[3] Z/ℓ_0 verifies the following property for any TA $\mathcal{B}$:

$$\mathcal{A} \sim \mathcal{B} \quad \text{iff} \quad \mathcal{B}' \models Z^{\ell_0}$$

where $\mathcal{B}'$ denotes the TA without invariant described in Section 4.1.

Then the timed modal logic L_ν is expressive enough to describe the behavior of timed automata.

[2] http://www.lsv.ens-cachan.fr/ fl/cmcweb.html

[3] with a declaration assigning a definition to every new identifier Z^{ℓ}.

5 An extension of L_ν

Recently a new operator has been added to L_ν in order to express controllability properties [BCL05a]. More generally the expressivity of L_ν is limited in the way of dealing with delay transitions: the modalities $\langle\delta\rangle$ and $[\delta]$ are respectively too weak or too strong for specifying some natural properties.

For example, consider the classical $\mathsf{E}a\mathsf{W}b$ formula: “there exists a path satisfying $a\mathsf{W}b$” where W stands for the *weak* until operator: either the path satisfies $a\mathsf{U}b$ (there is eventually a b and the previous states satisfy a), or it satisfies $\mathsf{G}a$ (“always a”). In classical Kripke structures, this property can be expressed as the greatest fixpoint of $Y \overset{\text{def}}{=} b \vee (a \wedge \mathsf{EX}\,Y)$ where EX denotes the Next operator. In our framework, the property $\mathsf{E}\varphi\mathsf{W}\psi$ cannot be expressed in L_ν due to the lacks of expressivity of $\langle\delta\rangle$ and $[\delta]$. Indeed, from a current state, we need to state that there is a way of letting time elapse s.t. all visited states satisfy φ, until a state where an action can be performed and so on. But the formula $\langle\delta\rangle\,\xi$ allows us to specify that after some delay, the property ξ holds, but there is no requirement on the intermediary states. And the formula $[\delta]\,\xi$ requires that *any* state reachable via a delay transition has to satisfy ξ. This does not allow us to express $\mathsf{E}\varphi\mathsf{W}\psi$ and which has motivated the introduction of the operator $[\delta\rangle$ defined as follows [BCL05a]:

$$\begin{aligned}(\ell,v,u) \models \varphi\,[\delta\rangle\,\psi \Leftrightarrow\ & \text{either } \forall t \in \mathbb{R}_+,\ (\ell,v) \xrightarrow{t} (\ell,v+t) \Rightarrow (\ell,v+t,u+t) \models \varphi \\ & \text{or } \exists t \in \mathbb{R}_+ \text{ s.t. } (\ell,v) \xrightarrow{t} (\ell,v+t) \text{ and } (\ell,v+t,u+t) \models \psi \text{ and} \\ & \forall 0 \le t' < t,\ \text{we have } (\ell,,v+t',u+t') \models \varphi\end{aligned}$$

Note that in [HNSY94] a timed μ-calculus (T_μ) has been defined with a modality $\rhd$ whose semantics is close to the one of $[\delta\rangle$ [4].

The new operator $[\delta\rangle$ is a kind of weak Until operator over delay transitions. We could have defined the operator $[\delta\rangle\rangle$ corresponding to a (strong) Until requiring that a position satisfying ψ exists. But we can easily define this modality with $[\delta\rangle$ and vice versa:

$$\begin{aligned}\varphi\,[\delta\rangle\rangle\,\psi &\equiv \varphi\,[\delta\rangle\,\psi \wedge \langle\delta\rangle\,\psi \\ \varphi\,[\delta\rangle\,\psi &\equiv \varphi\,[\delta\rangle\rangle\,\psi \vee [\delta]\,\varphi\end{aligned}$$

Let L_ν^+ be the extension of L_ν with the modality $[\delta\rangle$. This new logic is more expressive than L_ν:

Lemma 5.1 ([BCL05a]). *The L_ν^+ formula $([a]\,\mathtt{ff})\ [\delta\rangle\ (\langle b\rangle\,\mathtt{tt})$ has no equivalent in L_ν.*

The full proof can be found in [BCL05b]. Let Φ be the formula $([a]\,\mathtt{ff})\ [\delta\rangle\ (\langle b\rangle\,\mathtt{tt})$. The difficult point is that it is not possible to find two TA $\mathcal{A}$

[4] The main difference between $\rhd$ and $[\delta\rangle$ is that $\rhd$ may include an action transition after the delay

and $\mathcal{B}$ such that $\mathcal{A} \models \Phi$, $\mathcal{B} \not\models \Phi$ and $\mathcal{A} \models \psi \Leftrightarrow \mathcal{B} \models \psi$ for any $\psi \in L_\nu$. Indeed as we have seen in Section 4.3, L_ν formulas allow us to distinguish between two TA that are not bisimilar and if $\mathcal{A} \models \Phi$ and $\mathcal{B} \not\models \Phi$, then $\mathcal{A} \not\approx \mathcal{B}$. This is a classical problem in temporal logic [EMSS91] where one shows that two temporal logics may have different *expressive powers* even if they have the same *distinguishing power* and this makes the proof about expressivity more difficult.

Adding the modality $[\delta\rangle$ does not modify the interesting properties of L_ν:

- The logic L_ν^+ allows compositional model checking. It is sufficient to add the following rule to the quotient definition:

$$\Big(\varphi_1\ [\delta\rangle\ \varphi_2\Big)/\ell \stackrel{\text{def}}{=} \Big(\mathsf{Inv}(\ell) \Rightarrow (\varphi_1/\ell)\Big)\ [\delta\rangle\ \Big(\mathsf{Inv}(\ell) \wedge (\varphi_2/\ell)\Big)$$

- Moreover there is no additional complexity for the verification: Model checking L_ν^+ is EXPTIME-complete.

Control and controllability. The main motivation for adding $[\delta\rangle$ to L_ν was related to *control problems*. In this framework, we address a more general problem than classical model checking: given a plant P and a control objective Φ, one aims at synthesizing a controller C such that $C(P) \models \Phi$ where $C(P)$ denotes the plant supervised by C. Usually P is a (timed) automaton describing a system and its environment and we often distinguish between controllable and uncontrollable events in P: the controller can only act over controllable event in order to satisfy the property Φ. Moreover the notion of *supervision* $C(P)$ can be seen as a simple synchronization function between the two automata. Finally note that we can consider different kinds of controller: the sampled controllers (performing an action every Δ time units — Δ being fixed), or the more general case of the dense-time controllers (we just require that at least Δ time units elapse between two controllable actions).

Note that synthesizing C — when it exists — is closely related to the satisfiability problem and it is an open problem when P, E and C are TA and $\Phi \in L_\nu$.

Nevertheless it has been shown in [BCL05a] that when a control objective belongs to a *deterministic* fragment of L_ν, it is possible to reduce the controllability problem — the existence of a controller — to a model checking problem. More precisely, given P and Φ, the existence of a controller for P and Φ can be reduced to a model checking instance $(P|A_\Delta) \models \Phi'$ where:

- Φ' is an L_ν^+ formula that can be constructed automatically from Φ,
- A_Δ is a simple timed automaton describing the type of the controller (sampled, or dense-time).

If Φ' holds for $(P|A_\Delta)$, then we know that there exists a strategy for choosing controllable actions in P in order to satisfy Φ. This strategy — i.e. the controller — is a timed transition system but it may be non definable as a timed automaton: a TTS is much more general.

Acknowledgments. Special thanks to Luca Aceto, Béatrice Bérard, Patricia Bouyer, Franck Cassez and Kim Guldstrand Larsen for all the discussions about timed automata and timed modal logics.

References

[ACD93] R. Alur, C. Courcoubetis, and D. L. Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, 1993.

[ACH94] R. Alur, C. Courcoubetis, and T. A. Henzinger. The observational power of clocks. In *Proc. 5th Int. Conf. Theory of Concurrency (CONCUR '94), Uppsala, Sweden, Aug. 1994*, volume 836 of *Lecture Notes in Computer Science*, pages 162–177. Springer, 1994.

[AD94] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[AFH96] R. Alur, T. Feder, and T. A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, 1996.

[AH92] R. Alur and T. A. Henzinger. Logics and models of real time: A survey. In *Real-Time: Theory in Practice, Proc. REX Workshop, Mook, NL, June 1991*, volume 600 of *Lecture Notes in Computer Science*, pages 74–106. Springer, 1992.

[AH94] R. Alur and T. A. Henzinger. A really temporal logic. *Journal of the ACM*, 41(1):181–203, 1994.

[AL02] L. Aceto and F. Laroussinie. Is your model checker on time? On the complexity of model checking for timed modal logics. *Journal of Logic and Algebraic Programming*, 52-53:7–51, August 2002.

[And95] H. R. Andersen. Partial model checking (extended abstract). In *Proc. 10th IEEE Symp. Logic in Computer Science (LICS '95), San Diego, CA, USA, June 1995*, pages 398–407. IEEE Comp. Soc. Press, 1995.

[BCL05a] P. Bouyer, F. Cassez, and F. Laroussinie. Modal logics for timed control. In M. Abadi and L. de Alfaro, editors, *Proceedings of the 16th International Conference on Concurrency Theory (CONCUR'05)*, Lecture Notes in Computer Science, San Francisco, CA, USA, August 2005. Springer. To appear.

[BCL05b] P. Bouyer, F. Cassez, and F. Laroussinie. Modal logics for timed control. Research Report LSV-05-04, Laboratoire Spécification et Vérification, ENS Cachan, France, April 2005. 23 pages.

[BCM05] P. Bouyer, F. Chevalier, and N. Markey. On the expressiveness of tptl and mtl. Technical Report 05, Laboratoire Specification et Vérification, May 2005.

[CGP99] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking.* MIT Press, 1999.

[CL00] F. Cassez and F. Laroussinie. Model-checking for hybrid systems by quotienting and constraints solving. In E. A. Emerson and A. P. Sistla, editors, *Proceedings of the 12th International Conference on Computer Aided Verification (CAV 2000)*, volume 1855 of *Lecture Notes in Computer Science*, pages 373–388, Chicago, Illinois, USA, July 2000. Springer.

[Eme90] E. A. Emerson. Temporal and modal logic. In J. v. Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 16, pages 995–1072. Elsevier Science, 1990.

[EMSS91] E. A. Emerson, A. K. Mok, A. P. Sistla, and J. Srinivasan. Quantitative temporal reasoning. In *Proc. 2nd Int. Workshop Computer-Aided Verification (CAV '90), New Brunswick, NJ, USA, June 1990*, volume 531 of *Lecture Notes in Computer Science*, pages 136–145. Springer, 1991.

[HNSY94] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.

[LL95] F. Laroussinie and K. G. Larsen. Compositional model-checking of real time systems. In I. Lee and S. A. Smolka, editors, *Proceedings of the 6th International Conference on Concurrency Theory (CONCUR'95)*, volume 962 of *Lecture Notes in Computer Science*, pages 529–539, Philadelphia, Pennsylvania, USA, August 1995. Springer.

[LL98] F. Laroussinie and K. G. Larsen. CMC: A tool for compositional model-checking of real-time systems. In S. Budkowski, A. R. Cavalli, and E. Najm, editors, *Proceedings of IFIP TC6 WG6.1 Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE'XI) and Protocol Specification, Testing and Verification (PSTV'XVIII)*, volume 135 of *IFIP Conference Proceedings*, pages 439–456, Paris, France, November 1998. Kluwer Academic Publishers.

[LLW95] F. Laroussinie, K. G. Larsen, and C. Weise. From timed automata to logic – and back. In J. Wiedermann and P. Hájek, editors, *Proceedings of the 20th International Symposium on Mathematical Fundations of Computer Science (MFCS'95)*, volume 969 of *Lecture Notes in Computer Science*, pages 27–41, Prague, Czech Republic, August 1995. Springer.

[LPY97] K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a nutshell. *Journal of Software Tools for Technology Transfer*, 1(1–2):134–152, 1997.

[Mil89] R. Milner. A complete axiomatisation for observational congruence of finite-state behaviours. *Information and Computation*, 81(2):227–247, 1989.

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification.* Springer, 1992.

[OW05] J. Ouaknine and J. Worrell. On the decidability of metric temporal logic. In *Proc. 20th IEEE Symp. Logic in Computer Science (LICS 2005), Chicago, IL, USA, June 2005*, 2005.

[Par81] D. Park. Concurrency and automata on infinite sequences. In *Proc. 5th GI Conf. on Theor. Comp. Sci., Karlsruhe, FRG, Mar. 1981*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer, 1981.

[Yov97] S. Yovine. Kronos: A verification tool for real-time systems. *Journal of Software Tools for Technology Transfer*, 1(1–2):123–133, 1997.