Modal Logics for Timed Control*

Patricia Bouyer[1], Franck Cassez[2], François Laroussinie[1]

[1] LSV, UMR 8643, CNRS & ENS de Cachan, France
Email: {bouyer,fl}@lsv.ens-cachan.fr

[2] IRCCyN, UMR 6597, CNRS, France
Email: cassez@irccyn.ec-nantes.fr

Abstract. In this paper we use the timed modal logic L_ν to specify control objectives for timed plants. We show that the control problem for a large class of objectives can be reduced to a model-checking problem for an extension (L_ν^{cont}) of the logic L_ν with a new modality.
More precisely we define a fragment of L_ν, namely L_ν^{det}, such that any control objective of L_ν^{det} can be translated into a L_ν^{cont} formula that holds for the plant if and only if there is a controller that can enforce the control objective.
We also show that the new modality of L_ν^{cont} strictly increases the expressive power of L_ν while model-checking of L_ν^{cont} remains EXPTIME-complete.

1 Introduction

Control problem. The *control problem* (CP) for discrete event systems was first studied by Ramadge & Wonham in [RW89]. The CP is the following: "Given a finite-state model of a *plant* P (*open system)* with controllable and uncontrollable discrete actions, a *control objective* Φ, does there exist a *controller* f such that the plant supervised by f (*closed system*) satisfies Φ?" The *dense-time* version of the CP with an *untimed* control objective has been investigated and solved in [MPS95]. In this seminal paper, Maler *et al.* consider a plant P given by a *timed game automaton* which is a standard timed automaton [AD94] with its set of discrete actions partitioned into controllable and uncontrollable actions. They give an algorithm to decide whether a controller exists or not, and show that if one such controller exists a witness can be effectively computed. In [WT97] a semi-algorithm has been proposed to solve the CP when the plant is defined by a hybrid (game) automaton.

Specification of control properties. In the aforementioned papers the control objective is either a *safety* or *reachability* property (or some simple Büchi conditions). In [dAHM01] the authors give an algorithm to deal with general ω-regular control objectives. It is to be noticed that those control objectives are often called *internal* in the sense that they refer to the state properties (and

* Work supported by ACI Cortos, a program of the French government.

clocks) of the system to be controlled. In the case of timed systems they only refer to the untimed sequences of states of the system and thus have a restrictive expressiveness: it is possible to specify a property like "after p has been reached q will be reached" but nothing like "after p has been reached, q will be reached within less than d time units" (*bounded liveness*). Moreover, in the verification methodology for closed systems, one usually models (and thinks of) the plant P and the controller f as a closed system $f(P)$, and specifies a property φ with a suitable timed temporal logic and check whether the closed system $f(P)$ satisfies φ. It is then very natural to have similar logics in the game framework to specify timed control objectives for *open* systems.

Our contribution. The logic L_ν [LL95] is a subset of the timed μ-calculus that can be used for specifying timed safety properties of closed timed systems. Modalities of L_ν seem to be appropriate to specify timed control objectives as well because we can use existential and universal quantifications over *discrete* actions (as it is used in the untimed framework of [AVW03,RP03]), and also over time *delays*. The control problem CP for a plant (specified as a timed automaton) and a control objective in L_ν expresses as folloes:

> *Given a timed automaton P, the* plant*, and a L_ν formula φ, the* safety control objective*, is there a* controller *f s.t. $f(P) \models \varphi$?* (CP)

So far there is no constraint neither on the structure nor on the power of the controller f we are looking for: it may even require unbounded memory or arbitrary small delays between two consecutive controllable actions. In this paper we focus on controllability (CP) and not on the controller synthesis problem (*i.e.* exhibit a witness controller).

The main result of the paper is that we can reduce CP for a plant P and an L_ν control objective φ, to a standard *model-checking* problem on the plant P and a formula φ_c of a more expressive logic L_ν^{cont}, that extends L_ν with a new modality. More precisely we exhibit a *deterministic* fragment of L_ν, namely L_ν^{det}, s.t. for all $\varphi \in L_\nu^{\text{det}}$, the following reduction (RED) holds:

$$\left(\textit{There exists a controller } f \textit{ s.t. } f(P) \models \varphi\right) \iff P \models \varphi_c \qquad \text{(RED)}$$

where φ_c is a formula of L_ν^{cont}. We also give an effective procedure to obtain φ_c from φ.

Further on we study the logic L_ν^{cont} and prove that it is strictly more expressive than L_ν, which is a technically involved result on its own. We also show that the new modality of L_ν^{cont} is not necessary when we restrict our attention to *sampling* control (the controller can do an action every Δ time units) or to *Known Switch Conditions Dense-Time* control (where time elapsing is uncontrollable [CHR02]). A natural question following equation (RED) above is to study the model-checking problem for timed automata against L_ν^{cont} specifications. In the paper we prove that *i*) the model-checking of L_ν^{cont} over timed automata is EXPTIME-complete; *ii*) L_ν^{cont} inherits the *compositionality* property of L_ν.

Related work. In the discrete (untimed) case many logics used to specify *correctness* properties of closed systems have been extended to specify *control objectives* of open systems. ATL [AHK02] (resp. ATL*) is the *control* version of CTL (resp. CTL*). More recently [AVW03,RP03] have considered a more general framework in which properties of the controlled system are specified in various extensions of the μ-calculus: *loop* μ-calculus for [AVW03] and *quantified* μ-calculus for [RP03]. In both cases the control problem is reduced to a model-checking (or satisfiability) problem as in equation (RED). In the timed framework, external specifications have been studied in [DM02]: properties of the controlled system are specified with timed automata, and in [FLTM02], the control objective is given as a formula of the logic TCTL.

Outline of the paper. In section 2 we define basic notions used in the paper: timed systems, logic L_ν and variants and the control problem. In section 3 we prove that (RED) holds and also that φ_c is in L_ν for two simpler control problems. Section 4 is devoted to the study of the logic L_ν^{cont} (expressiveness, decidability, and compositionality).

The proofs are omitted and can be found in [BCL05].

2 Timed Automata and the Timed Modal Logic L_ν

We consider as time domain the set $\mathbb{R}_{\geq 0}$ of non-negative reals. Act is a finite set of *actions*.[3] We consider a finite set X of variables, called *clocks*. A *clock valuation* over X is a mapping $v : X \rightarrow \mathbb{R}_{\geq 0}$ that assigns to each clock a time value. The set of all clock valuations over X is denoted $\mathbb{R}_{\geq 0}^X$. Let $t \in \mathbb{R}_{\geq 0}$, the valuation $v + t$ is defined by $(v + t)(x) = v(x) + t$ for all $x \in X$. For $Y \subseteq X$, we denote by $v[Y \leftarrow 0]$ the valuation assigning 0 (resp. $v(x)$) for any $x \in Y$ (resp. $x \in X \setminus Y$).
We denote $\mathcal{C}(X)$ the set of *clock constraints* defined as the conjunctions of atomic constraints of the form $x \bowtie c$ with $x \in X$, $c \in \mathbb{Q}_{\geq 0}$ and $\bowtie \in \{<, \leq, =, \geq, >\}$. For $g \in \mathcal{C}(X)$ and $v \in \mathbb{R}_{\geq 0}^X$, we write $v \models g$ if v satisfies g and $[\![g]\!]$ denotes the set $\{v \in \mathbb{R}_{\geq 0}^X \mid v \models g\}$.

2.1 Timed Transition Systems & Timed Automata

Timed transition systems. A *timed transition system* (TTS) is a tuple $S = (Q, q_0, \mathsf{Act}, \longrightarrow_S)$ where Q is a set of states, $q_0 \in Q$ is the initial state, and $\longrightarrow_S \subseteq Q \times (\mathsf{Act} \cup \mathbb{R}_{\geq 0}) \times Q$ is a set of transitions. If $(q, e, q') \in \longrightarrow_S$, we also write $q \xrightarrow{e}_S q'$. The transitions labeled by $a \in \mathsf{Act}$ (resp. $t \in \mathbb{R}_{\geq 0}$) are called *action* (resp. *delay*) transitions. We make the following common assumptions about TTSs [Yi90]:

- 0-delay: $q \xrightarrow{0}_S q'$ if and only if $q = q'$,

[3] We assume that Act and $\mathbb{R}_{\geq 0}$ are disjoint.

– Additivity: if $q \xrightarrow{d}_S q'$ and $q' \xrightarrow{d'}_S q''$ with $d, d' \in \mathbb{R}_{\geq 0}$, then $q \xrightarrow{d+d'}_S q''$,
– Continuity: if $q \xrightarrow{d}_S q'$, then for every d' and d'' in $\mathbb{R}_{\geq 0}$ such that $d = d' + d''$, there exists q'' such that $q \xrightarrow{d'}_S q'' \xrightarrow{d''}_S q'$,
– Time-determinism: if $q \xrightarrow{e}_S q'$ and $q \xrightarrow{e}_S q''$ with $e \in \mathbb{R}_{\geq 0}$, then $q' = q''$.

A *run* is a finite or infinite sequence $\rho = s_0 \xrightarrow{e_1}_S s_1 \xrightarrow{e_2}_S \cdots \xrightarrow{e_n} s_n \cdots$ We denote by $first(\rho) = s_0$. If ρ is finite, $last(\rho)$ denotes the last state of ρ. $\mathsf{Runs}(q, S)$ is the set of runs in S starting from q and $\mathsf{Runs}(S) = \mathsf{Runs}(q_0, S)$. We use $q \xrightarrow{e}_S$ as a shorthand for "$\exists q'$ s.t. $q \xrightarrow{e}_S q'$" and extend this notation to finite runs $\rho \xrightarrow{e}_S$ whenever $last(\rho) \xrightarrow{e}_S$.

Timed automata. A *timed automaton (TA)* [AD94] 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, $\mathsf{inv} : L \to \mathcal{C}(X)$ is a mapping 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 transitions[4]. The semantics of a TA $\mathcal{A} = (L, \ell_0, \mathsf{Act}, X, \mathsf{inv}, T)$ is a TTS $S_\mathcal{A} = (L \times \mathbb{R}^X_{\geq 0}, (\ell_0, v_0), \mathsf{Act}, \longrightarrow_{S_\mathcal{A}})$ where $v_0(x) = 0$ for all $x \in X$ and $\longrightarrow_{S_\mathcal{A}}$ consists of: *i)* action transition: $(\ell, v) \xrightarrow{a}_{S_\mathcal{A}} (\ell', v')$ if there exists a transition $\ell \xrightarrow{g,a,Y} \ell'$ in T s.t. $v \models g$, $v' = v[Y \leftarrow 0]$ and $v' \models \mathsf{inv}(\ell')$; *ii)* delay transitions: $(\ell, v) \xrightarrow{t}_{S_\mathcal{A}} (\ell, v')$ if $t \in \mathbb{R}_{\geq 0}$, $v' = v + t$ and $v, v' \in \mathsf{inv}(\ell)$.
A TA is *deterministic* w.r.t. $\Sigma \subseteq \mathsf{Act}$ if for all $a \in \Sigma$, if $(\ell, g_1, a, Y_1, \ell_1) \in T$ and $(\ell, g_2, a, Y_2, \ell_2) \in T$ then $[\![g_1]\!] \cap [\![g_2]\!] = \emptyset$.

2.2 The Modal Logics L_ν, L_ν^{det} and L_ν^{cont}

The modal logic L_ν ***[LL95,LL98].*** The logic L_ν over the finite set of clocks K, the set of identifiers Id, and the set of actions Act is defined as the set of formulae generated by the following grammar:

$$\varphi ::= \mathtt{tt} \mid \mathtt{ff} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid x \ \underline{\mathsf{in}} \ \varphi \mid x \bowtie c \mid [a]\,\varphi \mid \langle a \rangle\, \varphi \mid [\delta] \ \varphi \mid \langle \delta \rangle\, \varphi \mid Z$$

where $a \in \mathsf{Act}$, $x \in K$, $\bowtie \in \{<, \leq, =, \geq, >\}$, $c \in \mathbb{Q}_{\geq 0}$, $Z \in \mathsf{Id}$.
The meaning of the identifiers is specified by a declaration $\mathcal{D}$ assigning a L_ν formula to each identifier. When $\mathcal{D}$ is understood we write $Z =_\nu \Psi_Z$ if $\mathcal{D}(Z) = \Psi_Z$. We define the following shorthands in L_ν: $r \ \underline{\mathsf{in}} \ \varphi \stackrel{def}{=} x_1 \ \underline{\mathsf{in}} \ x_2 \ \underline{\mathsf{in}} \ \cdots \ \underline{\mathsf{in}} \ x_n \ \underline{\mathsf{in}} \ \varphi$ if $r = \{x_1, \cdots, x_n\} \subseteq K$.
Let $S = (Q, q_0, \mathsf{Act}, \longrightarrow_S)$ be a TTS. L_ν formulae are interpreted over extended states (q, v) where for $q \in Q$ and $v \in \mathbb{R}^K_{\geq 0}$. We write "$S, (q, v) \models \varphi$" when an extended state (q, v) satisfies φ in the TTS S. This satisfaction relation is defined as the largest relation satisfying the implications in Table 1. The modalities $\langle e \rangle$ with $e \in \mathsf{Act} \cup \{\delta\}$ correspond to existential quantification over action or delay

[4] We often write $\ell \xrightarrow{g,a,Y} \ell'$ instead of simply the tuple (ℓ, g, a, Y, ℓ').

transitions, and $[e]$ is the counterpart for universal quantification. An extended state satisfies an identifier Z (denoted $S,(q,v) \models Z$) if it belongs to the maximal fixedpoint of the equation $Z =_\nu \Psi_Z$. Finally the formula clocks are used to measure time elapsing in properties. We define $[\![\varphi]\!]_S = \{(q,v) \mid S,(q,v) \models \varphi\}$. We write $S \models \varphi$ for $S,(q_0,v_0) \models \varphi$ where $v_0(x) = 0$ for all $x \in K$. The logic L_ν allows us to express many behavioural properties of timed systems [LL98]. For example the formula Z defined by $\Psi_Z = (\bigwedge_{a \in \mathsf{Act}} [a]\, Z \wedge [\delta]\, Z \wedge \varphi)$ holds when all reachable states satisfy φ. Other examples of formulae will be given later on in the paper.

$$
\begin{array}{l}
S,(q,v) \models \alpha \implies \alpha \text{ with } \alpha \in \{\mathtt{tt}, \mathtt{ff}\} \\
S,(q,v) \models x \bowtie c \implies v(x) \bowtie c \\
S,(q,v) \models Z \implies S,(q,v) \models \Psi_Z \\
S,(q,v) \models \varphi_1 \ op\ \varphi_2, \implies S,(q,v) \models \varphi_1 \quad op \quad S,(q,v) \models \varphi_2 \text{ with } \ op \in \{\wedge, \vee\} \\
S,(q,v) \models x \ \underline{\mathsf{in}}\ \varphi \implies S,(q,v[x \leftarrow 0]) \models \varphi \\
S,(q,v) \models [a]\, \varphi \implies \text{for all } q \xrightarrow{a}_S q',\ S,(q',v) \models \varphi \\
S,(q,v) \models \langle a \rangle\, \varphi \implies \text{there is some } q \xrightarrow{a}_S q',\ S,(q',v) \models \varphi \\
S,(q,v) \models [\delta]\, \varphi \implies \text{for all } t \in \mathbb{R}_{\geq 0} \text{ s.t. } q \xrightarrow{t}_S q',\ S,(q',v+t) \models \varphi \\
S,(q,v) \models \langle \delta \rangle\, \varphi \implies \text{there is some } t \in \mathbb{R}_{\geq 0} \text{ s.t. } q \xrightarrow{t}_S q',\ S,(q',v+t) \models \varphi
\end{array}
$$

Table 1. Satisfaction implications for L_ν

The modal logic L_ν^{cont}. As we will see later in the paper, the modal operators of L_ν are not sufficient to express dense-time control. Indeed we need to express the persistence (w.r.t. time elapsing) of a property **until** a controllable action is performed: we thus need to express that some property is true only for a subset of the states of the plant which are reachable by time elapsing **before** a controllable action leading to good states is possible. This kind of property cannot be expressed using the $[\delta]$ and $\langle \delta \rangle$ operators. This is why we define the new modality $[\delta\rangle$, the semantics of which is defined over an extended configuration (q,v) of a TTS S as follows:

$$
\begin{aligned}
S,(q,v) \models \varphi\, [\delta\rangle\, \psi \Leftrightarrow\ & \text{either } \forall t \in \mathbb{R}_{\geq 0},\ q \xrightarrow{t}_S q' \Rightarrow S,(q',v+t) \models \varphi \\
& \text{or } \exists t \in \mathbb{R}_{\geq 0} \text{ s.t. } q \xrightarrow{t}_S q' \text{ and } S,(q',v+t) \models \psi \text{ and} \\
& \forall 0 \leq t' < t,\ q \xrightarrow{t'}_S q'' \text{ we have } S,(q'',v+t') \models \varphi
\end{aligned}
\tag{1}
$$

Let L_ν^{cont} be the timed modal logic which extends L_ν by adding the modality $[\delta\rangle$. This operator is some kind of "Until" modality over delays. In [HNSY94] the timed μ-calculus which is studied contains a modality $\rhd$ the semantics of which is close to the semantics of $[\delta\rangle$ (the main difference between $\rhd$ and $[\delta\rangle$ is that $\rhd$ may include an action transition after the delay).

A deterministic fragment of L_ν***,*** L_ν^{det}***.*** In the following we will restrict the possible control objectives to properties expressed in a subset L_ν^{det} of L_ν. Indeed, we want to define a transformation such that equation (RED) given in the introduction holds, the restriction is then motivated by the following remark:

Remark 1. A control objective of L_ν like $\varphi_1 \wedge \varphi_2$ intuitively requires to find a controller that both ensures φ_1 and φ_2. In an inductive construction, this amounts to build a controller that ensures $\varphi_1 \wedge \varphi_2$ from two controllers: one that ensures φ_1 and an other that ensures φ_2. This means that we must be able to merge controllers in a suitable manner. The definition of L_ν^{det} will syntactically ensure that the conjunctions of L_ν^{det} formulae can be merged safely, *i.e.* that they are in some sense *deterministic*.
Indeed, any (first-level) subformula of a conjunction in L_ν^{det} will be prefixed by a modal operator with a particular action, and then the existence of a controller for φ_1 and another one for φ_2 entails the existence of a controller for $\varphi_1 \wedge \varphi_2$.
In the untimed case, some kind of "deterministic" form is also used (the so-called *disjunctive normal form*), but this is not a restriction as all formulae of the μ-calculus can be rewritten in a disjunctive normal form [JW95]. One hope could be to be able to transform any formula of L_ν into an equivalent formula of L_ν^{det}, but we do not know yet if this is possible. Note that in the untimed framework, transforming formulae of the μ-calculus into formulae in disjunctive normal form is strongly related to the satisfiability problem, and in the timed case, the satisfiability problem for L_ν is still an open problem [LLW95].

We first define *basic* terms B_ν by the following grammar:

$$\alpha ::= \mathtt{tt} \;\mid\; \mathtt{ff} \;\mid\; x \bowtie c \;\mid\; r \;\underline{\mathsf{in}}\; \langle a\rangle\, \varphi \;\mid\; r \;\underline{\mathsf{in}}\; [a]\, \varphi$$

with $x \in K$, $r \subseteq K$, $c \in \mathbb{Q}$ and $a \in \mathsf{Act} \cup \{\delta\}$ and $\varphi \in L_\nu^{\mathrm{det}}$ (L_ν^{det} is defined hereafter). A set of basic terms $A = \{\alpha_1, \alpha_2, \cdots, \alpha_n\}$ is *deterministic* if for all $\sigma \in \mathsf{Act} \cup \{\delta\}$ there is at most one i s.t. $\alpha_i = r \;\underline{\mathsf{in}}\; \langle\sigma\rangle\, \varphi$ or $\alpha_i = r \;\underline{\mathsf{in}}\; [\sigma]\, \varphi$. We then define L_ν^{det} as the deterministic fragment of L_ν inductively defined as follows:

$$L_\nu^{\mathrm{det}} \ni \varphi, \psi ::= X \;\mid\; \varphi \vee \psi \;\mid\; \bigwedge_{\alpha \in A} \alpha$$

with $X \in \mathsf{Id}$ and A a (finite) deterministic set of basic terms. With this restriction on the conjunctions, if there are controllers f_α for all $\alpha \in A$, we can merge them to obtain a controller for $\bigwedge_{\alpha \in A} \alpha$ (see remark 1 above).

Note that already many properties can be expressed in the fragment L_ν^{det}, for example safety and bounded liveness properties:

$$X_1 \;=\; [\mathsf{Bad}]\, \mathtt{ff} \;\wedge\; \bigwedge_{\mathsf{a} \neq \mathsf{Problem}, \mathsf{Bad}} [\mathsf{a}]\, \mathsf{X_1} \;\wedge\; [\mathsf{Problem}]\, (\mathsf{z} \;\underline{\mathsf{in}}\; \mathsf{X_2}) \;\wedge\; [\delta]\, \mathsf{X_1}$$

$$X_2 \;=\; z < d_{\max} \;\wedge [\mathsf{Bad}]\, \mathtt{ff} \wedge\; [\mathsf{Alarm}]\, \mathsf{X_1} \;\wedge\; \bigwedge_{\mathsf{a} \neq \mathsf{Alarm}, \mathsf{Bad}} [\mathsf{a}]\, \mathsf{X_2} \;\wedge\; [\delta]\, \mathsf{X_2}$$

The above formula expresses that the system is always safe (represented by property $[\mathsf{Bad}]\,\mathtt{ff}$), and that every $\mathsf{Problem}$ is followed in less than $d_{\max}$ time units by the Alarm. The previous formula can also be specified using simpler formalism (*e.g. test automaton* [ABBL03]) but this is not the case for every L_ν^{det} formula. The formula $X = [\delta]\,X \wedge \bigwedge_{a\in\mathsf{Act}}[a]\,X \wedge \langle\delta\rangle\,\langle b\rangle\,\mathtt{tt}$ for some $b \in \mathsf{Act}$, which means that there is always some delay s.t. b is enabled cannot be expressed with test automata.

2.3 The Control Problem

Definition 1 (Live Plant). *A* live plant *(plant in the sequel) P is a TA where* Act *is partitionned into* Act_u *and* Act_c *and s.t. 1) it is deterministic w.r.t. every $a \in \mathsf{Act}_c$; 2) in every state (ℓ, v) the TA P can let time elapse or do an uncontrollable action.*

A *controller* [MPS95] for a plant, is a function that during the evolution of the system constantly gives information as to what should be done in order to ensure a control objective Φ. In a given state the controller can either *i*) "enable some particular *controllable* action" or *ii*) "do nothing at this point in time, just wait" which will be denoted by the special symbol λ. Of course a controller cannot prevent uncontrollable actions from occurring. Nevertheless, we assume that the controller can **disable** a controllable action at any time, and this will not block the plant because the plant is *live*.

Definition 2 (Controller). *Let $P = (L, \ell_0, \mathsf{Act}, X, \mathsf{inv}, T)$ be a plant. A controller*[5] *f over P is a partial function from $\mathsf{Runs}(S_P)$ to $\mathsf{Act}_c \cup \{\lambda\}$ s.t. for any finite run $\rho \in \mathsf{Runs}(S_P)$, if $f(\rho)$ is defined*[6] *then $f(\rho) \in \{e \mid \rho \xrightarrow{e}_{S_P}\}$.*

The purpose of a controller f for a plant P is to restrict the set of behaviours in S_P in order to ensure that some property holds. Closing the plant P with f produces a TTS (set of runs) corresponding to the controlled plant:

Definition 3 (Controlled plant). *Let $P = (L, \ell_0, \mathsf{Act}, X, \mathsf{inv}, T)$ be a plant, $q \in S_P$ and f a controller over P. The* controlled plant *$f(S_P, q)$ is the TTS $(Q, q, \mathsf{Act}, \longrightarrow_f)$ defined inductively by:*

- *$q \in Q$,*
- *if $\rho \in \mathsf{Runs}(f(S_P, q))$, then $last(\rho) \xrightarrow{e}_f q'$ and $q' \in Q$, if $last(\rho) \xrightarrow{e}_{S_P} q'$ and one of the following three conditions hold:*
  1. *$e \in \mathsf{Act}_u$,*
  2. *$e \in \mathsf{Act}_c$ and $e = f(\rho)$,*
  3. *$e \in \mathbb{R}_{\geq 0}$ and $\forall 0 \leq e' < e, \exists last(\rho) \xrightarrow{e'}_{S_P} q''$ s.t. $\lambda = f(\rho \xrightarrow{e'}_{S_P} q'')$.*

We note $f(P)$ the controlled plant P by controller f from initial state of P.

[5] The notation f comes from the fact that a controller is specified as a function, as strategies in game theory.

[6] $\rho \xrightarrow{\lambda}_{S_P}$ stands here for $\exists t > 0$ s.t. $last(\rho) \xrightarrow{t}_{S_P} s'$.

The Δ-dense-time control problem amounts to finding a controller for a system s.t. at least $\Delta \geq 0$ time units elapse between two consecutive control actions. Such a controller is called a Δ-*controller* and can prevent time elapsing and force a controllable action to happen at any point in time if the time elapsed since the last controllable move is more than Δ. If $\Delta = 0$ we admit controllers that can do two consecutive actions separated by arbitrary small delays (even 0-delay), *i.e.* controllers that have infinite speed. If $\Delta > 0$, the Δ-controllers are forced to be *strongly non-zeno*. We note $\mathsf{Contr}_\Delta(P)$ the set of Δ-controllers for plant P.

Definition 4 (Δ-Dense-Time Control Problem). *Let $P = (L, \ell_0, \mathsf{Act}, X, \mathsf{inv}, T)$ be a plant, $\varphi \in L_\nu^{det}$, a (deterministic)* safety control objective*, and $\Delta \in \mathbb{Q}_{\geq 0}$. The* Δ-Dense-Time Control Problem *(Δ-CP for short) asks the following:*

$$\textit{Is there a controller } f \in \mathsf{Contr}_\Delta(P) \textit{ such that } f(P) \models \varphi\textit{?} \tag{Δ-CP}$$

Remark 2. In the above Δ-CP, we look for controllers which can do a controllable action only if the time elapsed since the last controllable action is at least Δ. We could specify many other classes of controllers: for example we could impose the controller doing controllable actions exactly every Δ units of time (this is called *sampling control* — see later), or to alternate controllable actions. Notice that this fits very well in our framework as we will see in section 4 that L_ν^{det} is *compositional*: any reasonable constraint on the controller can be given as an extra (timed) automaton and taken into account simply by synchronizing it with the plant P. For example the Δ-controllers can be specified by an extra self-loop automaton where the loop is constrained by a guard $x \geq \Delta$, any controllable action can be done, and clock x is reset.
In the following we note P_Δ the synchronized product of P with this self-loop automaton (see [AD94] for the definition of the classical synchronisation product).

3 From Control to Model Checking

In this section, we prove that for any control objective defined as a L_ν^{det} formula φ, we can build an L_ν^{cont} formula $\overline{\varphi}$ that holds for P_Δ iff there exists a Δ-controller which supervises plant P in order to satisfy φ. This corresponds to equation (RED) we have settled in the introduction.

3.1 Dense-Time Control Problem

Let φ be a L_ν^{det} formula and $\sigma \in \mathsf{Act}_c \cup \{\lambda\}$, we define the formula $\overline{\varphi}^{\,\sigma}$ by the inductive translation of Fig. 1. Intuitively, formula $\overline{\varphi}^{\,a_c}$ will hold when there is a controller which ensures φ and which starts by enforcing controllable action a_c whereas formula $\overline{\varphi}^{\,\lambda}$ will hold when there is a controller which ensures φ and which starts by delaying. We use the shortcut $\overline{\varphi}$ to express that nothing is required for the strategy, which will correspond to $\bigvee_{\sigma \in \mathsf{Act}_c \cup \{\lambda\}} \overline{\varphi}^{\,\sigma}$.We also use $\langle\lambda\rangle\,\mathtt{tt}$ as a shortcut for $\bigwedge_{a_c \in \mathsf{Act}_c} [a_c]\,\mathtt{ff}$. Note that the new operator $[\delta\rangle$ is used in the formula $\overline{[\delta]\,\varphi}^{\sigma}$. This translation rule introduces the superscript a_c in the disjunctive right argument of $[\delta\rangle$. This just means that we can actually prevent time from elapsing at some point, if we perform a controllable action.

$\overline{\bigwedge_{\alpha\in A}\alpha}^{\sigma} \stackrel{def}{=} \bigwedge_{\alpha\in A}\overline{\alpha}^{\sigma}$	$\overline{\bigvee_{\alpha\in A}\alpha}^{\sigma} \stackrel{def}{=} \bigvee_{\alpha\in A}\overline{\alpha}^{\sigma}$
$\overline{\langle a\rangle\,\varphi}^{\sigma} \stackrel{def}{=} \begin{cases} \mathtt{ff} & \text{if } \sigma, a\in \mathsf{Act}_c \wedge \sigma\neq a \\ \langle a\rangle\,\overline{\varphi}\wedge\langle\sigma\rangle\,\mathtt{tt} & \text{if } a\in\mathsf{Act}_u \\ \langle a\rangle\,\overline{\varphi} & \text{otherwise}\end{cases}$	$\overline{x\sim c}^{\sigma} \stackrel{def}{=} x\sim c\wedge\langle\sigma\rangle\,\mathtt{tt}$
$\overline{\langle\delta\rangle\,\varphi}^{\sigma} \stackrel{def}{=} \begin{cases} \langle\delta\rangle\,\overline{\varphi} & \text{if } \sigma=\lambda \\ \overline{\varphi}^{\sigma} & \text{if } \sigma\in\mathsf{Act}_c\end{cases}$	$\overline{r\ \underline{\mathsf{in}}\ \varphi}^{\sigma} \stackrel{def}{=} r\ \underline{\mathsf{in}}\ \overline{\varphi}^{\sigma}$
$\overline{[a_c]\,\varphi}^{\sigma} \stackrel{def}{=} \begin{cases} \langle\sigma\rangle\,\mathtt{tt} & \text{if } a_c\neq\sigma \\ \langle a_c\rangle\,\overline{\varphi} & \text{if } a_c=\sigma\end{cases}$	$\overline{[a_u]\,\varphi}^{\sigma} \stackrel{def}{=} [a_u]\,\overline{\varphi}\wedge\langle\sigma\rangle\,\mathtt{tt}$
$\overline{[\delta]\,\varphi}^{\sigma} \stackrel{def}{=} \begin{cases} \overline{\varphi}^{\sigma} & \text{if } \sigma\in\mathsf{Act}_c \\ \overline{\varphi}^{\lambda}\,[\delta\rangle\left(\bigvee_{a_c\in\mathsf{Act}_c}\overline{\varphi}^{a_c}\right) & \text{otherwise}\end{cases}$	$\overline{X}^{\sigma} \stackrel{def}{=} X_\sigma\wedge\langle\sigma\rangle\,\mathtt{tt}$

Fig. 1. Definition of $\overline{\varphi}^{\sigma}$, $\varphi\in L_\nu^{\mathrm{det}}$ and $\sigma\in\mathsf{Act}_c\cup\{\lambda\}$

We can now state our main theorem about controllability:

Theorem 1. *Given P a plant, $\varphi\in L_\nu^{\mathrm{det}}$ a control objective, $\Delta\in\mathbb{Q}_{\geq 0}$, we then have:*

$$\Big(\exists f\in\mathsf{Contr}_\Delta(P)\ s.t.\ f(P)\models\varphi\Big)\iff P_\Delta\models\overline{\varphi} \tag{2}$$

The proof of Theorem 1 can be done by induction on the structure of the formula and is given in [BCL05].

This theorem reduces the *controllability* problem for properties expressed in L_ν^{det} to some *model-checking* problem for properties expressed in L_ν^{cont}. Note however that this theorem does not provide a method to synthesize controllers: indeed L_ν and L_ν^{cont} are compositional logics (see in the next section), controller synthesis is thus equivalent to model synthesis. But, as already said, the satisfiability problem (or model synthesis) for L_ν^{det} or L_ν is still open (see [LLW95] for partial results about this problem). Note also that as L_ν^{cont} is compositional (see next section), verifying $P_\Delta\models\overline{\varphi}$ reduces to checking $P\models\overline{\varphi}/S_\Delta$ where S_Δ is the self-loop automaton mentioned before.

3.2 Known-Switch Condition Dense-Time Control

Known-switch condition (KSC) dense-time control [CHR02] corresponds to the control of the time-abstract model of a game: intuitively this assumes that time elapsing is not controllable. A controller can thus choose to do a controllable action $a \in \mathsf{Act}_c$ or to do nothing (λ), but in the latter case the controller does not control the duration of the next continuous move.

To see that L_ν is sufficient to express KSC dense-time control, we just need to focus on formula of the type $[\delta]\,\varphi$ as this is the only formula that may need the use of the $[\delta\rangle$ operator when translated into a model-checking formula. More precisely we only need to focus on the translation of $\overline{[\delta]\,\varphi}^{\,\lambda}$ as this is the only case that can generate a $[\delta\rangle$ formula. It is then clear that if the controller chooses λ, and as it has no way of controlling time-elapsing in the time-abstract system, it must ensure φ in all possible future positions in S. Thus $\overline{[\delta]\,\varphi}^{\,\lambda}$ simply reduces to $[\delta]\,\overline{\varphi}^{\,\lambda}$. Thus L_ν is sufficient to express KSC dense-time control.

3.3 Sampling Control

The *sampling* control problem is a version of the control problem where the controller can perform a controllable action only at dates $k.\Delta$ for $k \in \mathbb{N}$ and $\Delta \in \mathbb{Q}$. Δ is the sampling *rate* of the controller. Let P be a plant. As emphasized earlier in this section for the Δ-dense-time control, we can build a plant P_Δ where all the controllable actions are required to happen at multiple values of the sampling rate Δ. This can be done by defining a timed automaton $\mathcal{B}_\Delta$ with one location ℓ_0, a fresh clock y, the invariant $\mathsf{inv}(\ell_0) \equiv y \leq \Delta$ and a number of loops on ℓ_0: for each $a_c \in \mathsf{Act}_c$ there is a loop $(\ell_0, y = \Delta, a_c, \{y\}, \ell_0)$. Moreover we want to leave the controller free to do nothing. To this end we add a new controllable action *reset* and a loop $(\ell_0, y = \Delta, \mathit{reset}, \{y\}, \ell_0)$. As this action is not in P, it is harmless to do it and when the controller does not want to do an action, it can always choose to do *reset*.

Thus we can design an equivalent version of the sampling control where the controller is bound to do a controllable action at each date $k.\Delta$ with $k \in \mathbb{N}$. As in the previous case of KSC dense-time control problem, we just modify the definition of $\overline{[\delta]\,\varphi}^{\,\lambda}$ with:

$$\overline{[\delta]\,\varphi}^{\,\lambda} \stackrel{def}{=} [\delta]\,\Big(([\mathit{reset}]\,\mathtt{ff} \wedge \overline{\varphi}^{\,\lambda}) \vee \bigvee_{a_c \in \mathsf{Act}_c} \overline{\varphi}^{\,a_c}\Big)$$

which is equivalent to $[\delta]\overline{\varphi}$. Indeed the formula $[\mathit{reset}]\,\mathtt{ff}$ holds precisely when no controllable action can be perfomed by the controller; and when $\langle \mathit{reset}\rangle\,\mathtt{tt}$ holds, a controllable move has to be performed.

4 The Timed Modal Logic L_ν^{cont}

In this section we focus on the logic L_ν^{cont} and prove several properties of this logic, namely its expressive power, its decidability and compositionality.

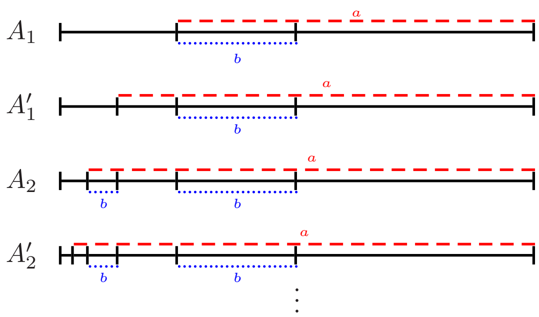
L_ν^{cont} is more expressive than L_ν. The modality "$[\delta\rangle$" has been introduced for expressing control properties of open systems. We now prove that this operator adds expressive power to L_ν, *i.e.* it can not be expressed with L_ν. As usual we say that two formulae φ and ψ are equivalent for a class of systems $\mathcal{S}$ (we then write $\varphi \equiv_{\mathcal{S}} \psi$) if for all $s \in \mathcal{S}$, $s \models \varphi$ iff $s \models \psi$. A logic L is said to be as expressive as L' over $\mathcal{S}$ (denoted $L \succeq_{\mathcal{S}} L'$) if for every $\varphi \in L'$, there exists $\psi \in L$ s.t. $\varphi \equiv_{\mathcal{S}} \psi$. And L is said to be strictly more expressive than L' if $L \succeq_{\mathcal{S}} L'$ and $L' \not\succeq_{\mathcal{S}} L$. We have the following result:

Theorem 2. *The logic L_ν^{cont} is strictly more expressive than L_ν over timed automata.*

The full proof is long and technical, we give it in [BCL05]. Here we just give the techniques which we have used. Let φ be the L_ν^{cont} formula $([a]\,\mathtt{ff})\ [\delta\rangle\ (\langle b\rangle\,\mathtt{tt})$ stating that no a-transition can be performed as long as (*via* delay transitions) no b has been enabled. The core of the proof is based on the fact that there is no L_ν formula equivalent to φ.

The difficult point is that it is not possible to find two TAs A and A' such that $A \models \varphi$, $A' \not\models \varphi$ and $A \models \psi \Leftrightarrow A' \models \psi$ for any $\psi \in L_\nu$. Indeed L_ν allows us to build a characteristic formula for a TA [LLW95] (*i.e.* a formula which describes the behaviour of A w.r.t. strong timed bisimulation) and clearly the two TAs A and A' wouldn't be bisimilar. This is a classical problem in temporal logic [Eme91] where one shows that two temporal logics may have different expressive power even if they have the same *distinguishing power*. This makes the proof more difficult. Such expressiveness problems are not much considered in the timed framework. Up to our knowledge this is one of the first proofs of that type for timed logics.

To prove the result, we build two families of TAs $(A_i)_{i\geq 1}$ and $(A'_i)_{i\geq 1}$ such that for every integer i, $A_i \models \varphi$ whereas $A'_i \not\models \varphi$. We then prove that if φ can be expressed equivalently as formula $\Phi \in L_\nu$ (over timed automata), then there must exist some integer $i \geq 1$ such that $A'_i \models \Phi$, which will be a contradiction. The behaviours of automata A_i and A'_i can be represented by (and infered from) the following picture.



Model-checking L_ν^{cont}. Model-checking of L_ν over TAs is an EXPTIME-complete problem [AL02]. Adding the modality $[\delta\rangle$ does not change this result, we have:

Theorem 3. *The model-checking of L_ν^{cont} over timed automata is EXPTIME-complete.*

Proof (Sketch). The EXPTIME-hardness comes from the EXPTIME-hardness of the model-checking of L_ν. For the EXPTIME-easyness, we just have to explain how to handle the $[\delta\rangle$ modality. Let A be a TA and $\Phi \in L_\nu^{\text{cont}}$. We consider the region graph [AD94] $R_{\mathcal{A}}$ associated with A and the set of formula clocks K. Clearly the classical notion of region can be used for $[\delta\rangle$: two states in a region r satisfy the same L_ν^{cont} formulae (the semantics of $[\delta\rangle$ can be defined in term of regions as well). Then we can define procedures to label R_A states with the Φ subformulae they satisfy. We can use the same algorithms as for L_ν to label $[\delta]\,\varphi$, $\langle\delta\rangle\,\varphi$, $\langle a\rangle\,\varphi, \ldots$ and define a new procedure for the $\varphi\,[\delta\rangle\,\psi$ subformulae. This can be done easily (as soon as φ and ψ have already been labeled) and it consists in a classical "Until" over the delay transitions (see below a way of computing $\varphi\,[\delta\rangle\,\psi$ with DBMs). The complexity of the algorithm will remain linear in the size of R_A and Φ, and finally exponential in the size of A and Φ [AL02]. $\square$

Instead of considering region techniques, classical algorithms for timed model-checking use *zones* (*i.e.* convex sets of valuations, defined as conjunctions of $x - y \bowtie c$ constraints and implemented with DBMs [Dil90,Bou04]). This makes verification more efficient in practice. In this approach $[\![\varphi]\!]$ is defined as sets of pairs (q, z) where z is a zone and q is a control state of the TA. This approach is also possible for L_ν^{cont}. Indeed we can define $[\![\varphi[\delta\rangle\psi]\!]$ when $[\![\varphi]\!]$ and $[\![\psi]\!]$ are already defined as sets of symbolic configurations (q, z). We use standard operations on zones: $\overleftarrow{z}$ (resp. $\overrightarrow{z}$, z^c) denotes the past (resp. future, complement) of z, and z^+ represents the set $z \cup \{v \mid \exists t > 0 \text{ s.t. } v - t \in z \text{ and } \forall 0 \le t' < t,\ v - t' \in z\}$ (if z is represented by a DBM in normal form, z^+ is computed by relaxing constraints $x < c$ to $x \le c$). It is then easy to prove that:

$$[\![\varphi\,[\delta\rangle\,\psi]\!] = \left(\overleftarrow{[\![\varphi]\!]^c}\right)^c \cup \left[\left(\overleftarrow{\left(\overrightarrow{[\![\psi]\!]} \cup [\![\varphi]\!]\right)^c}\right)^c \cap \left([\![\psi]\!] \cup \left([\![\varphi]\!] \cap \left(\overleftarrow{[\![\varphi]\!]^+ \cap [\![\psi]\!]}\right)\right)\right)\right]$$

L_ν^{cont} is compositional. An important property of L_ν is that it is *compositional* [LL95,LL98] for timed automata. This is also the case for L_ν^{cont}.
A logic L is said to be *compositional* for a class $\mathcal{S}$ of models if, given an instance $(s_1|\cdots|s_n) \models \varphi$ with $s_i \in \mathcal{S}$ and $\varphi \in L$, it is possible to build a formula φ/s_1 (called a *quotient* formula) s.t. $(s_1|\cdots|s_n) \models \varphi \Leftrightarrow (s_2|\cdots|s_n) \models \varphi/s_1$. This can be viewed as an encoding of the behaviour of s_1 into the formula. Of course this also depends on the synchronization function, but we will not enter into the details here.
For $\varphi \in L_\nu$, A a TA, it is possible to define inductively a *quotient formula* φ/A (we refer to [LL98] for a complete description of this technique). In order to prove that L_ν^{cont} is compositional it is sufficient to define the quotient formula

for the new modality $\varphi\,[\delta\rangle\,\psi$. We define the quotient of $\varphi_1\,[\delta\rangle\,\varphi_2$ for a location ℓ of a TA A in the following way:

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

With such a quotient construction we get the following proposition:

Proposition 1. *The logic L_ν^{cont} is compositional for the class of timed automata.*

We have discussed a little bit in previous sections why the property is very useful and important. In particular, the new modality of L_ν^{cont} has been added to the model-checker CMC [LL98] which implements a compositional model-checking algorithm: it first computes a quotient formula of the system and the property and then check for the satisfiability of the formula. We have added to CMC the quotient rule for the operator $[\delta\rangle$ and thus we can use CMC for checking controllability properties. We do not provide here our experimental results but better refer to the web page of the tool: `http://www.lsv.ens-cachan.fr/~fl/cmcweb.html`.

5 Conclusion

In this paper we have used the logic L_ν to specify control objectives on timed plants. We have proved that a deterministic fragment of L_ν allows us to reduce control problems to a model-checking problem for an extension of L_ν (denoted L_ν^{cont}) with a new modality. We have also studied the properties of the extended logic L_ν^{cont} and proved that *i*) L_ν^{cont} is strictly more expressive than L_ν; *ii*) the model-checking of L_ν^{cont} over timed automata is EXPTIME-complete; *iii*) L_ν^{cont} inherits the *compositionality* property of L_ν.

Our current and future work is many-fold:

- extend our work to the synthesis of controllers. Note that this problem is strongly related to the satisfiability problem for L_ν^{det} and L_ν which is still open [LLW95].
- use the features of the logic L_ν to express more general types of control objectives *e.g.* to take into account dynamic changes of the set of controllable events as in [AVW03].

References

[ABBL03] Luca Aceto, Patricia Bouyer, Augusto Burgueño, and Kim G. Larsen. The power of reachability testing for timed automata. *Theoretical Computer Science*, 300(1–3):411–475, 2003.

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

[AHK02] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49:672–713, 2002.

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

[AVW03] André Arnold, Aymeric Vincent, and Igor Walukiewicz. Games for synthesis of controllers with partial observation. *Theoretical Computer Science*, 1(303):7–34, 2003.

[BCL05] Patricia Bouyer, Franck Cassez, and François Laroussinie. Modal logics for timed control. Research Report LSV-05-04, Laboratoire Spécification & Vérification, ENS de Cachan, France, 2005.

[Bou04] Patricia Bouyer. Forward analysis of updatable timed automata. *Formal Methods in System Design*, 24(3):281–320, 2004.

[CHR02] Franck Cassez, Thomas A. Henzinger, and Jean-François Raskin. A comparison of control problems for timed and hybrid systems. In *Proc. 5th International Workshop on Hybrid Systems: Computation and Control (HSCC'02)*, volume 2289 of *LNCS*, pages 134–148. Springer, 2002.

[dAHM01] Luca de Alfaro, Thomas A. Henzinger, and Rupak Majumdar. Symbolic algorithms for infinite-state games. In *Proc. 12th International Conference on Concurrency Theory (CONCUR'01)*, volume 2154 of *Lecture Notes in Computer Science*, pages 536–550. Springer, 2001.

[Dil90] David Dill. Timing assumptions and verification of finite-state concurrent systems. In *Proc. of the Workshop on Automatic Verification Methods for Finite State Systems (1989)*, volume 407 of *Lecture Notes in Computer Science*, pages 197–212. Springer, 1990.

[DM02] Deepak D'Souza and P. Madhusudan. Timed control synthesis for external specifications. In *Proc. 19th International Symposium on Theoretical Aspects of Computer Science (STACS'02)*, volume 2285 of *Lecture Notes in Computer Science*, pages 571–582. Springer, 2002.

[Eme91] E. Allen Emerson. *Temporal and Modal Logic*, volume B (Formal Models and Semantics) of *Handbook of Theoretical Computer Science*, pages 995–1072. MIT Press Cambridge, 1991.

[FLTM02] Marco Faella, Salvatore La Torre, and Aniello Murano. Dense real-time games. In *Proc. 17th Annual Symposium on Logic in Computer Science (LICS'02)*, pages 167–176. IEEE Computer Society Press, 2002.

[HNSY94] Thomas A. Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. Symbolic model-checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.

[JW95] David Janin and Igor Walukiewicz. Automata for the modal mu-calculus and related results. In *Proc. 20th International Symposium on Mathematical Foundations of Computer Science (MFCS'95)*, volume 969 of *Lecture Notes in Computer Science*, pages 552–562. Springer, 1995.

[LL95] François Laroussinie and Kim G. Larsen. Compositional model-checking of real-time systems. In *Proc. 6th International Conference on Concurrency Theory (CONCUR'95)*, volume 962 of *Lecture Notes in Computer Science*, pages 27–41. Springer, 1995.

[LL98] François Laroussinie and Kim G. Larsen. CMC: A tool for compositional model-checking of real-time systems. In *Proc. IFIP Joint International Conference on Formal Description Techniques & Protocol Specification, Testing, and Verification (FORTE-PSTV'98)*, pages 439–456. Kluwer Academic, 1998.

[LLW95] François Laroussinie, Kim G. Larsen, and Carsten Weise. From timed automata to logic – and back. In *Proc. 20th International Symposium on Mathematical Foundations of Computer Science (MFCS'95)*, volume 969 of *Lecture Notes in Computer Science*, pages 529–539. Springer, 1995.

[MPS95] Oded Maler, Amir Pnueli, and Joseph Sifakis. On the synthesis of discrete controllers for timed systems. In *Proc. 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS'95)*, volume 900 of *Lecture Notes in Computer Science*, pages 229–242. Springer, 1995.

[RP03] Stéphane Riedweg and Sophie Pinchinat. Quantified mu-calculus for control synthesis. In *Proc. 28th International Symposium on Mathematical Foundations of Computer Science (MFCS'03)*, volume 2747 of *Lecture Notes in Computer Science*, pages 642–651. Springer, 2003.

[RW89] P.J.G. Ramadge and W.M. Wonham. The control of discrete event systems. *Proc. of the IEEE*, 77(1):81–98, 1989.

[WT97] Howard Wong-Toi. The synthesis of controllers for linear hybrid automata. In *Proc. 36th IEEE Conference on Decision and Control*, pages 4607–4612. IEEE Computer Society Press, 1997.

[Yi90] Wang Yi. Real-time behaviour of asynchronous agents. In *Proc. 1st International Conference on Theory of Concurrency (CONCUR'90)*, volume 458 of *Lecture Notes in Computer Science*, pages 502–520. Springer, 1990.