P. Bouyer, F. Cassez and
F. Laroussinie

# Modal Logics for Timed Control

Research Report LSV-05-04
April 2005

Laboratoire
Spécification
et
Vérification

CENTRE NATIONAL
DE LA RECHERCHE
SCIENTIFIQUE

Ecole Normale Supérieure de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex France

# 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 $L_\nu$ 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 ($L_\nu^{\text{cont}}$) of the logic $L_\nu$ with a new modality.
More precisely we define a fragment of $L_\nu$, namely $L_\nu^{\text{det}}$, such that any control objective of $L_\nu^{\text{det}}$ can be translated into a $L_\nu^{\text{cont}}$ 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 $L_\nu^{\text{cont}}$ strictly increases the expressive power of $L_\nu$ while model-checking of $L_\nu^{\text{cont}}$ 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* $\Phi$, does there exist a *controller* $f$ such that the plant supervised by $f$ (*closed system*) satisfies $\Phi$?"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 $\omega$-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 $\varphi$ with a suitable timed temporal logic and check whether the closed system $f(P)$ satisfies $\varphi$. 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 $L_\nu$ [LL95] is a subset of the timed $\mu$-calculus that can be used for specifying timed safety properties of closed timed systems. Modalities of $L_\nu$ 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 $L_\nu$ expresses as folloes:

*Given a timed automaton $P$, the* plant*, and a $L_\nu$ formula $\varphi$, 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 $L_\nu$ control objective $\varphi$, to a standard *model-checking* problem on the plant $P$ and a formula $\varphi_c$ of a more expressive logic $L_\nu^{\mathrm{cont}}$, that extends $L_\nu$ with a new modality. More precisely we exhibit a *deterministic* fragment of $L_\nu$, namely $L_\nu^{\mathrm{det}}$, s.t. for all $\varphi \in L_\nu^{\mathrm{det}}$, the following reduction (RED) holds:

$$\Big(\textit{There exists a controller } f \textit{ s.t. } f(P) \models \varphi\Big) \iff P \models \varphi_c \tag{RED}$$

where $\varphi_c$ is a formula of $L_\nu^{\mathrm{cont}}$. We also give an effective procedure to obtain $\varphi_c$ from $\varphi$.

Further on we study the logic $L_\nu^{\mathrm{cont}}$ and prove that it is strictly more expressive than $L_\nu$, which is a technically involved result on its own. We also show that the new modality of $L_\nu^{\mathrm{cont}}$ is not necessary when we restrict our attention to *sampling* control (the controller can do an action every $\Delta$ time units) or to *Known Switch Conditions Dense-Time* control (where time elapsing is uncontrollable [CHR02]).

A natural question following the reduction of equation (RED) above is to study the model-checking problem for timed automata against $L_\nu^{\mathrm{cont}}$ specifications. In the paper we prove that *i*) the model-checking of $L_\nu^{\mathrm{cont}}$ over timed automata is EXPTIME-complete; *ii*) $L_\nu^{\mathrm{cont}}$ inherits the *compositionality* property of $L_\nu$.

***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 $\mu$-calculus: *loop* $\mu$-calculus for [AVW03] and *quantified* $\mu$-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 $\mathsf{TCTL}$.

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

The proofs are given in appendices.

## 2 Timed Automata and the Timed Modal Logic $L_\nu$

We consider as time domain the set $\mathbb{R}_{\geq 0}$ of non-negative reals. $\mathsf{Act}$ 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]:

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

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

– $\mathsf{Additivity}$: if $q \xrightarrow{d}_S q'$ and $q' \xrightarrow{d'}_S q''$ with $d, d' \in \mathbb{R}_{\geq 0}$, then $q \xrightarrow{d+d'}_S q''$,
– $\mathsf{Continuity}$: 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 $q \xrightarrow{d'}_S q'' \xrightarrow{d''}_S q'$,
– $\mathsf{Time\text{-}determinism}$: 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 $\mathit{first}(\rho) = s_0$. If $\rho$ is finite, $\mathit{last}(\rho)$ denotes the last state of $\rho$. $\mathsf{Runs}(q, S)$ 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 extends this notation to finite runs $\rho \xrightarrow{e}_S$ whenever $\mathit{last}(\rho) \xrightarrow{e}_S$.

***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 $v, v' \in \mathsf{inv}(\ell)$.

### 2.2 The Modal Logics $L_\nu$, $L_\nu^{\mathrm{det}}$ and $L_\nu^{\mathrm{cont}}$

***The modal logic $L_\nu$ [LL95,LL98].*** The logic $L_\nu$ over the finite set of clocks $K$, the set of identifiers $\mathsf{Id}$, and the set of actions $\mathsf{Act}$ 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 $L_\nu$ 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 $L_\nu$: $r \ \underline{\mathsf{in}}\ \varphi \stackrel{def}{=} x_1 \ \underline{\mathsf{in}}\ x_2 \ \underline{\mathsf{in}}\ \cdots \ \underline{\mathsf{in}}\ x_n \ \underline{\mathsf{in}}\ \varphi$ if $r = \{x_1, \cdots, x_n\} \subseteq K$.
Let $S = (Q, q_0, \mathsf{Act}, \longrightarrow_S)$ be a TTS. $L_\nu$ 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 $\varphi$ 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 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

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

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 $L_\nu$ 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 $\varphi$. Other examples of formulae will be given later on in the paper.

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

**Table 1.** Satisfaction implications for $L_\nu$

***The modal logic $L_\nu^{cont}$.*** As we will see later in the paper, the modal operators of $L_\nu$ 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 $L_\nu^{\text{cont}}$ be the timed modal logic which extends $L_\nu$ by adding the modality $[\delta\rangle$. This operator is some kind of "Until" modality over delays. In [HNSY94] the timed $\mu$-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 $L_\nu$, $L_\nu^{det}$.*** In the following we will restrict the possible control objectives to properties expressed in a subset $L_\nu^{\text{det}}$ of $L_\nu$. 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: a control objective of $L_\nu$ like $\varphi_1 \wedge \varphi_2$ intuitively requires to find a controller that both ensures $\varphi_1$ and $\varphi_2$. In an inductive construction, this amounts to build a controller that ensures $\varphi_1 \wedge \varphi_2$ from two controllers: one that ensures $\varphi_1$ and an other that ensures $\varphi_2$. This means that we must be able to merge controllers in a suitable manner. The definition of $L_\nu^{\text{det}}$ will syntactically ensure that the conjunctions of $L_\nu^{\text{det}}$ formulae can be merged safely, *i.e.* that they are in some sense *deterministic*.

Indeed, any (first-level) subformula of a conjunction in $L_\nu^{\text{det}}$ will be prefixed by a modal operator with a particular action, and then the existence of a controller for $\varphi_1$ and another one for $\varphi_2$ 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 $\mu$-calculus can be rewritten in a disjunctive normal form [JW95]. One hope could be to be able to transform any formula of $L_\nu$ into an equivalent formula of $L_\nu^{\text{det}}$, but we do not know yet if this is possible. Note that in the untimed framework, transforming formulae of the $\mu$-calculus into formulae in disjunctive normal form is strongly related to the satisfiability problem, and in the timed case, the satisfiability problem for $L_\nu$ is still an open problem [LLW95].

We first define *basic* terms $B_\nu$ 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 $\varphi \in L_\nu^{\text{det}}$ ($L_\nu^{\text{det}}$ 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 $L_\nu^{\text{det}}$ as the deterministic fragment of $L_\nu$ inductively defined as follows:

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

with $X \in \mathsf{Id}$ and $A$ a (finite) deterministic set of basic terms.

Note that already many properties can be expressed in the fragment $L_\nu^{\text{det}}$, for example safety and bounded liveness properties:

$$\begin{aligned} X_1 \ &= [\mathsf{Bad}]\, \mathtt{ff} \ \wedge \bigwedge_{\mathsf{a} \neq \mathsf{Problem}, \mathsf{Bad}} [\mathsf{a}]\, \mathsf{X}_1 \ \wedge \ [\mathsf{Problem}]\, (\mathsf{z} \ \underline{\mathsf{in}} \ \mathsf{X}_2) \ \wedge \ [\delta]\, \mathsf{X}_1 \\ X_2 \ &= z < d_{\max} \ \wedge [\mathsf{Bad}]\, \mathtt{ff} \wedge \ [\mathsf{Alarm}]\, \mathsf{X}_1 \ \wedge \bigwedge_{\mathsf{a} \neq \mathsf{Alarm}, \mathsf{Bad}} [\mathsf{a}]\, \mathsf{X}_2 \ \wedge \ [\delta]\, \mathsf{X}_2 \end{aligned}$$

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 $\mathsf{Alarm}$.

### 2.3 The Control Problem

**Definition 1 (Fair Plant).** *A* fair plant *(plant in the sequel) $P$ is a TA where* $\mathsf{Act}$ *is partitionned into* $\mathsf{Act}_u$ *and* $\mathsf{Act}_c$ *and s.t. 1) it is deterministic w.r.t. every $a \in \mathsf{Act}_c$; 2) in every state $(\ell, v)$ 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 $\Phi$. 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 $\lambda$. 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 *fair*.

**Definition 2 (Controller).** *Let $P = (L, \ell_0, \mathsf{Act}, X, \mathsf{inv}, T)$ be a plant. A* controller *$f$[5] 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 $e \in f(\rho)$,*
  3. *$e \in \mathbb{R}_{\geq 0}$ and $\forall 0 \leq e' < e, \exists last(\rho) \xrightarrow{e'}_{S_P} q''$ s.t. $\lambda \in f(\rho \xrightarrow{e'}_{S_P} q'')$.*

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

The $\Delta$-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 $\Delta$*-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 $\Delta$. 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 $\Delta$-controllers are forced to be *strongly non-zeno*. We note $\mathsf{Contr}_\Delta(P)$ the set of $\Delta$-controllers for plant $P$.

[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. $last(\rho) \xrightarrow{t}_{S_P} s'$.

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

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

*Remark 1.* In the above $\Delta$-CP, we look for controllers which can do a controllable action only if the time elapsed since the last controllable action is at least $\Delta$. We could specify many other classes of controllers: for example we could impose the controller doing controllable actions exactly every $\Delta$ 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 $L_\nu^{\text{det}}$ 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 $\Delta$-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 $P_\Delta$ 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 $L_\nu^{\text{det}}$ formula $\varphi$, we can build an $L_\nu^{\text{cont}}$ formula $\varphi_c$ that holds for $P_\Delta$ iff there exists a $\Delta$-controller which supervises plant $P$ in order to satisfy $\varphi$. This corresponds to equation (RED) we have settled in the introduction.

### 3.1 Dense-Time Control Problem

Let $\varphi$ be a $L_\nu^{\text{det}}$ 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 $\varphi$ and which starts by enforcing controllable action $a_c$ whereas formula $\overline{\varphi}^{\,\lambda}$ will hold when there is a controller which ensures $\varphi$ 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.
We can now state our main theorem about controllability:

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

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

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

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

The proof of Theorem 1 can be done by induction on the structure of the formula and is given in appendix A.

This theorem reduces the *controllability* problem for properties expressed in $L_\nu^{\text{det}}$ to some *model-checking* problem for properties expressed in $L_\nu^{\text{cont}}$. Note however that this theorem does not provide a method to synthesize controllers: indeed $L_\nu$ and $L_\nu^{\text{cont}}$ 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 $L_\nu$ is still an open problem [LLW95]. Note also that as $L_\nu^{\text{cont}}$ is compositional (see next section), verifying $P_\Delta\models\overline{\varphi}$ reduces to checking $P\models\overline{\varphi}\,/S_\Delta$ where $S_\Delta$ 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 ($\lambda$), but in the latter case the controller does not control the duration of the next continuous move.
To see that $L_\nu$ 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 $\lambda$, and as it has no way of controlling time-elapsing in the time-abstract system, it must ensure $\varphi$ in all possible future positions in $S$. Thus $\overline{[\delta]\,\varphi}^{\,\lambda}$ simply reduces to $[\delta]\,\overline{\varphi}^{\,\lambda}$. Thus $L_\nu$ 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 $\Delta \in \mathbb{Q}$. $\Delta$ is the sampling *rate* of the controller. Let $P$ be a plant. As emphasized earlier in this section for the $\Delta$-dense-time control, we can build a plant $P_\Delta$ where all the controllable actions are required to happen at multiple values of the sampling rate $\Delta$. This can be done by defining a timed automaton $\mathcal{B}_\Delta$ with one location $\ell_0$, a fresh clock $y$, the invariant $\mathsf{inv}(\ell_0) \equiv y \leq \Delta$ and a number of loops on $\ell_0$: 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:

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

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 $L_\nu^{\mathrm{cont}}$

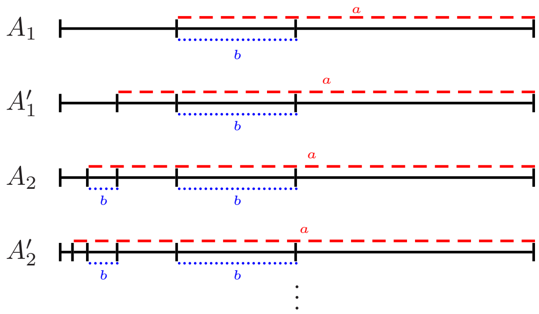
In this section we focus on the logic $L_\nu^{\mathrm{cont}}$ and prove several properties of this logic, namely its expressive power, its decidability and compositionality.

***$L_\nu^{cont}$ is more expressive than $L_\nu$.*** The modality "$[\delta\rangle$" has been introduced for expressing control properties of open systems. We now prove that this operator adds expressive power to $L_\nu$, *i.e.* it can not be expressed with $L_\nu$. As usual we say that two formulae $\varphi$ and $\psi$ 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 $L_\nu^{cont}$ is strictly more expressive than $L_\nu$ over timed automata.*

The full proof is long and technical, we give it in the appendix, page 19. Here we just give the techniques which we have used. Let $\varphi$ be the $L_\nu^{\mathrm{cont}}$ 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 $L_\nu$ formula equivalent to $\varphi$.
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 $L_\nu$ 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 $\varphi$ 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 $L_\nu^{cont}$.** Model-checking of $L_\nu$ 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 $L_\nu^{cont}$ over timed automata is EXPTIME-complete.*

*Proof (Sketch).* The EXPTIME-hardness comes from the EXPTIME-hardness of the model-checking of $L_\nu$. 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] $R_A$ 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 $L_\nu^{\text{cont}}$ 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 $\Phi$ subformulae they satisfy. We can use the same algorithms as for $L_\nu$ 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 $\varphi$ and $\psi$ 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 $\Phi$, and finally exponential in the size of $A$ and $\Phi$ [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 $L_\nu^{\text{cont}}$. 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 $z \cup \{v \mid \exists t > 0 \text{ s.t. } v - t \in z \text{ and } \forall 0 \leq t' < t,\ v - t' \in z\}$ (if $z$ is represented by a DBM in normal form, $z^+$ is computed by relaxing constraints $x < c$ to $x \leq c$). It is then easy to prove that:

$$[\![\varphi\,[\delta\rangle\,\psi]\!] = \Big(\overleftarrow{[\![\varphi]\!]^c}\Big)^c \;\cup\; \Bigg[\bigg(\overleftarrow{\Big(\overrightarrow{[\![\psi]\!]} \cup [\![\varphi]\!]\Big)^c}\bigg)^c \cap \Big([\![\psi]\!] \cup \Big([\![\varphi]\!] \cap \Big(\overleftarrow{[\![\varphi]\!]^+ \cap [\![\psi]\!]}\Big)\Big)\Big)\Bigg]$$

***$L_\nu^{cont}$ is compositional.*** An important property of $L_\nu$ is that it is *compositional* [LL95,LL98] for timed automata. This is also the case for $L_\nu^{\text{cont}}$.
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 $\varphi/s_1$ (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* $\varphi/A$ (we refer to [LL98] for a complete description of this technique). In order to prove that $L_\nu^{\text{cont}}$ 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 $\ell$ 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 $L_\nu^{cont}$ 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 $L_\nu^{\text{cont}}$ 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 $L_\nu$ to specify control objectives on timed plants. We have proved that a deterministic fragment of $L_\nu$ allows us to reduce control problems to a model-checking problem for an extension of $L_\nu$ (denoted $L_\nu^{\text{cont}}$) with a new modality. We have also studied the properties of the extended logic $L_\nu^{\text{cont}}$ and proved that *i*) $L_\nu^{\text{cont}}$ is strictly more expressive than $L_\nu$; *ii*) the model-checking of $L_\nu^{\text{cont}}$ over timed automata is EXPTIME-complete; *iii*) $L_\nu^{\text{cont}}$ inherits the *compositionality* property of $L_\nu$.

Our current and future work is many-fold:

- extend our work to the synthesis of controllers. An interesting point is that we are not limited to the synthesis of controllers which are timed automata. Note however that this problem is strongly related to the satisfiability problem for $L_\nu$ which is still open [LLW95].
- use the features of the logic $L_\nu$ 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


[AD94] Rajeev Alur and David Dill. A theory of timed automata. *Theoretical Computer Science (TCS)*, 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.

[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 IEEE 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.

## A Proof of Theorem 1

Before proving Theorem 1 we need to take some notations.

Let $G = (Q, q_0, \mathsf{Act}, \longrightarrow)$ be a TTS. For $s \in Q$ and $\Delta \in \mathbb{Q}_{\geq 0}$ we define:

- $G^s_\Delta$ to be the sub TTS of $G$ rooted at $s$ such that two consecutive controllable actions (in $\mathsf{Act}_c$) are separated by a time amount $t$ s.t. $t \geq \Delta$;
- $G_\Delta$ stands for $G^{q_0}_\Delta$;
- for $\tau \leq \Delta$, $G^s_{\Delta,\tau}$ is the sub TTS of $G^s_\Delta$ where no controllable action occurs before $\tau$ time units from the root $s$;
- $\mathsf{Contr}(G^s_{\Delta,\tau}, s)$ is the set of controllers for TTS $G^s_{\Delta,\tau}$ from state $s$; $\mathsf{Contr}(G^{q_0}_{\Delta,0}, q_0)$ thus denotes the set of controllers that can let at least $\Delta$ time units between two consecutive controllable actions.

If $G$ is the semantics of a plant $P = (L, \ell_0, \mathsf{Act}, X, \mathsf{inv}, T)$ the TTS $G^{q_0}_\Delta$ can be effectively constructed using a parallel composition with a self-loop automaton (with a fresh clock $x$) enforcing a delay greater than $\Delta$ (e.g. by $x \geq \Delta$) between two controllable actions. We denote $P_\Delta$ this synchronized product.

Theorem 1 is a consequence of the following lemma:

**Lemma 1.** *For any $\sigma \in \mathsf{Act}_c \cup \{\lambda\}$, any state $s \in G$ and any $L^{det}_\nu$ formula $\Phi$, we have:*

$$\Big(\exists f \in \mathsf{Contr}(G^s_{\Delta,\tau}, s)\ s.t.\ f(G,s),(s,v) \models \Phi \wedge \langle\sigma\rangle\, \mathtt{tt}\Big) \Leftrightarrow \Big(G^s_{\Delta,\tau},(s,v) \models \overline{\Phi}^{\,\sigma}\Big)$$

Note how this lemma interprets for formulae $\overline{\Phi}$ :

$$\Big(\exists f \in \mathsf{Contr}(G^s_{\Delta,\tau}, s)\ \text{s.t.}\ f(G,s),(s,v) \models \Phi\Big) \Leftrightarrow \Big(G^s_{\Delta,\tau},(s,v) \models \overline{\Phi}\,\Big)$$

*Proof.* First we assume that the result holds for the fixpoint variables and we show the Lemma by structural induction over $L^{\det}_\nu$ formulae. The cases $\Phi \stackrel{def}{=} x \sim c$ or $\Phi \stackrel{def}{=} r\ \underline{\mathsf{in}}\ \varphi$ are obvious.

- $\Phi \stackrel{def}{=} [a_u]\, \varphi$:
  - $\Rightarrow$ Assume there exists $f \in \mathsf{Contr}(G^s_{\Delta,\tau}, s)$ s.t. $f(G,s),(s,v) \models [a_u]\, \varphi \wedge \langle\sigma\rangle\, \mathtt{tt}$. Then for any $f(G,s) : s \xrightarrow{a_u} s'$, we have $f(G,s),(s',v) \models \varphi$. Then there exists $f' \in \mathsf{Contr}(G^s_{\Delta,\tau}, s')$ s.t. $f'(G,s'),(s',v) \models \varphi$ and the induction hypothesis provides: $G^{s'}_{\Delta,\tau},(s',v) \models \overline{\varphi}$ . Since the strategies cannot block the uncontrollable actions, any action $a_u \in \mathsf{Act}_u$ that can be performed from $(s,v)$ in $G^s_{\Delta,\tau}$, can also be performed in $f(G,s)$ and then $G^s_{\Delta,\tau},(s,v) \models [a_u]\overline{\varphi}$ . Moreover $f(G,s),(s,v) \models \langle\sigma\rangle\mathtt{tt}$ which implies that $G^s_{\Delta,\tau},(s,v) \models \langle\sigma\rangle\, \mathtt{tt}$, and thus $G^s_{\Delta,\tau},(s,v) \models \overline{[a_u]\, \varphi}^{\,\sigma}$.

$\Leftarrow$ Assume $G^s_{\Delta,\tau},(s,v) \models [a_u]\,\overline{\varphi} \wedge \langle\sigma\rangle\,\mathtt{tt}$. For any transition $G^s_{\Delta,\tau}, s \xrightarrow{a_u} s'$, we have $G^s_{\Delta,\tau},(s',v) \models \overline{\varphi}$. By i.h. we know that there exists $f_{a_u} \in \mathsf{Contr}(G^s_{\Delta,\tau}, s')$ s.t. $f_{a_u}(G,s'),(s',v) \models \varphi$. Let $f$ be the strategy defined by: $f(s \xrightarrow{a_u} \rho) \stackrel{def}{=} f_{a_u}(\rho)$ for any $\rho$ starting in state $s'$ and $f(s) = a_c$ if $\sigma = a_c$ (note that in that case, it is possible to do a $\sigma$ because $G^s_{\Delta,\tau},(s,v) \models \langle\sigma\rangle\,\mathtt{tt}$), or $f(s) = \lambda$ otherwise.

– $\Phi \stackrel{def}{=} \langle a_u\rangle\,\varphi$:

$\Rightarrow$ Assume there exists $f \in \mathsf{Contr}(G^s_{\Delta,\tau}, s)$ s.t. $f(G,s),(s,v) \models \langle a_u\rangle\varphi \wedge \langle\sigma\rangle\mathtt{tt}$. Then there exists $f(G,s) : s \xrightarrow{a_u} s'$ with $f(G,s),(s',v) \models \varphi$. Therefore there is $f' \in \mathsf{Contr}(G^s_{\Delta,\tau}, s')$ s.t. $f'(G,s'),(s',v) \models \varphi$ and the i.h. entails: $G^{s'}_{\Delta,\tau},(s',v) \models \overline{\varphi}$. $G^s_{\Delta,\tau}$ contains the behaviours of $f(G,s)$, then $G^s_{\Delta,\tau},(s,v) \models \langle a_u\rangle\,\overline{\varphi}$. Moreover, $f(G,s),(s,v) \models \langle\sigma\rangle\,\mathtt{tt}$, thus $G^s_{\Delta,\tau},(s,v) \models \langle a_u\rangle\,\overline{\varphi} \wedge \langle\sigma\rangle\,\mathtt{tt}$, and thus $G^s_{\Delta,\tau},(s,v) \models \overline{\langle a_u\rangle\,\varphi}^{\sigma}$.

$\Leftarrow$ Assume $G^s_{\Delta,\tau},(s,v) \models \langle a_u\rangle\,\overline{\varphi} \wedge \langle\sigma\rangle\,\mathtt{tt}$. There is a transition $G^s_{\Delta,\tau}, s \xrightarrow{a_u} s'$ s.t. $G^s_{\Delta,\tau},(s',v) \models \overline{\varphi}$. By i.h. we know that there exists $f_{a_u} \in \mathsf{Contr}(G^s_{\Delta,\tau}, s')$ s.t. $f_{a_u}(G,s'),(s',v) \models \varphi$. Let $f$ be the strategy defined by: $f(s \xrightarrow{a_u} \rho) \stackrel{def}{=} f_{a_u}(\rho)$ for any $\rho$ starting in state $s'$ and $f(s) = a_c$ if $\sigma = a_c$, or $f(s) = \lambda$ otherwise.

– $\Phi \stackrel{def}{=} \langle a_c\rangle\,\varphi$:

$\Rightarrow$ There exists $f \in \mathsf{Contr}(G^s_{\Delta,\tau}, s)$ s.t. $f(G,s),(s,v) \models \langle a_c\rangle\,\varphi \wedge \langle\sigma\rangle\,\mathtt{tt}$. Then clearly $\sigma$ is $a_c$: otherwise this would entail that $f$ is not deterministic and requires two different controllable actions from the state $(s,v)$. There exists $f(G,s) : s \xrightarrow{a_c} s'$ such that $f(G,s'),(s',v) \models \varphi$. Moreover defining $f' \in \mathsf{Contr}(G^s_{\Delta,\Delta}, s')$ by $f'(\rho) \stackrel{def}{=} f(s \xrightarrow{a_c} \rho)$ for any $\rho$ starting in $s'$, we get that $f'(G,s'),(s',v) \models \varphi$. By i.h. we have $G^{s'}_{\Delta,\Delta},(s',v) \models \overline{\varphi}$. $G^s_{\Delta,\tau}$ contains the behaviours of $f(G,s)$, then $G^s_{\Delta,\tau},(s,v) \models \langle a_c\rangle\,\overline{\varphi}$ and thus $G^s_{\Delta,\tau},(s,v) \models \overline{\langle a_c\rangle\,\varphi}^{\sigma}$.

$\Leftarrow$ The only possible case is $\sigma = a_c$ and $G^s_{\Delta,\tau},(s,v) \models \langle a_c\rangle\,\overline{\varphi}$. There is a transition $G^s_{\Delta,\tau}, s \xrightarrow{a_c} s'$ s.t. $G^{s'}_{\Delta,\Delta},(s',v) \models \overline{\varphi}$. By i.h. we know that there exists $f' \in \mathsf{Contr}(G^s_{\Delta,\Delta}, s')$ s.t. $f'(G,s'),(s',v) \models \varphi$. Let $f$ be the strategy defined by: $f(s \xrightarrow{a_c} \rho) \stackrel{def}{=} f'(\rho)$ for any $\rho$ run starting in $s'$ and $f(s) = a_c$. $f$ is a $\Delta$-strategy and belongs to $\mathsf{Contr}(G^s_{\Delta,\tau}, s)$ — note that in this case $\tau = 0$ —, and $f(G,s),(s,v) \models \langle a_c\rangle\,\varphi$ and then $\langle a_c\rangle\,\mathtt{tt}$ also holds for $f(G,s),(s,v)$.

– $\Phi \stackrel{def}{=} [a_c]\,\varphi$:

$\Rightarrow$ If $a_c \neq \sigma$ the result is obvious. Now assume $a_c = \sigma$, then there exists $f \in \mathsf{Contr}(G^s_{\Delta,\tau}, s)$ s.t. $f(G,s),(s,v) \models [a_c]\,\varphi \wedge \langle a_c\rangle\,\mathtt{tt}$. The same proof as above (for $\Phi \stackrel{def}{=} \langle a_c\rangle\,\varphi$) gives $G^s_{\Delta,\tau},(s,v) \models \langle a_c\rangle\,\overline{\varphi}$, *i.e.* $G^s_{\Delta,\tau},(s,v) \models \overline{[a_c]\,\varphi}^{\sigma}$.

$\Leftarrow$ First assume $\sigma \in \mathsf{Act}_c \backslash \{a_c\}$ or $\sigma = \lambda$. Then $G^s_{\Delta,\tau},(s,v) \models \langle\sigma\rangle\mathtt{tt}$ we define the strategy $f$ to be $f(s) = \sigma$. This allows us to have $f(G,s),(s,v) \models$

$[a_c]\varphi \wedge \langle\sigma\rangle \mathtt{tt}$ (as $a_c$ is disabled by $f$). Finally assume $\sigma = a_c$. Then we have $G^s_{\Delta,\tau},(s,v) \models \langle a_c\rangle \overline{\varphi}$ . There exists $G^s_{\Delta,\tau}, s \xrightarrow{a_c} s'$ s.t. $G^{s'}_{\Delta,\Delta},(s',v) \models \overline{\varphi}$ . By i.h. there exists $f \in \mathsf{Contr}(G^s_{\Delta,\Delta}, s')$ s.t. $f'(G,s'),(s',v) \models \varphi$. Let $f$ be the strategy defined by $f(s) = a_c$ and $f(s \xrightarrow{a_c} \rho) = f'(\rho)$ for any $\rho$ run starting in state $s'$. We have $f \in \mathsf{Contr}(G^s_{\Delta,\tau}, s)$ and $f(G,s),(s,v) \models [a_c]\,\varphi \wedge \langle\sigma\rangle\,\mathtt{tt}$.

– $\Phi \stackrel{def}{=} \langle\delta\rangle\,\varphi$:
  - ⇒ First assume $\sigma = \lambda$. If there exists $f \in \mathsf{Contr}(G^s_{\Delta,\tau}, s)$ s.t. $f(G,s),(s,v) \models \langle\delta\rangle\,\varphi$, then there is $f(G,s), s \xrightarrow{t} s^t$ (with $t \in \mathbb{R}$) s.t. $f(G,s),(s^t,v+t) \models \varphi$. By i.h. we have $G^{s'}_{\Delta,\tau-t},(s',v+t) \models \overline{\varphi}$ where $\tau - t$ stands for $\max(\tau - t, 0)$. And then $G^s_{\Delta,\tau},(s,v) \models \langle\delta\rangle\,\overline{\varphi}$ because $G^s_{\Delta,\tau}$ is more general than $f(G,s)$.
    Now assume $\sigma = a_c$. There exists $f \in \mathsf{Contr}(G^s_{\Delta,\tau}, s)$ s.t. $f(G,s),(s,v) \models \langle\delta\rangle\,\varphi \wedge \langle a_c\rangle\,\mathtt{tt}$. This means that $f(s) = a_c$ and then no delay is allowed by the strategy and then it is equivalent to $f(G,s),(s,v) \models \varphi \wedge \langle a_c\rangle\,\mathtt{tt}$. By i.h. we have: $G^s_{\Delta,\tau},(s,v) \models \overline{\varphi}^{\,a_c}$.
  - ⇐ Assume $\sigma = \lambda$. If $G^s_{\Delta,\tau},(s,v) \models \langle\delta\rangle\,\overline{\varphi}$ , then there exists $G^s_{\Delta,\tau} : s \xrightarrow{t} s^t$ with $t \in \mathbb{R}$ s.t. $G^s_{\Delta,\tau},(s^t,v+t) \models \overline{\varphi}$ . By i.h. we deduce that there exists $f^t \in \mathsf{Contr}(G^s_{\Delta,\tau-t}, s^t)$ s.t. $f^t(G,s^t),(s^t,v+t) \models \varphi$. Let $f \in \mathsf{Contr}(G^s_{\Delta,\tau}, s)$ be the strategy defined as $f(s \xrightarrow{t'} s^{t'}) = \lambda$ for any $t' < t$, and $f(s \xrightarrow{t} \rho) = f^t(\rho)$ for any run $\rho$ starting in state $s^t$. Clearly we have $f(G,s),(s,v) \models \langle\delta\rangle\,\varphi$.

    Assume $\sigma \in \mathsf{Act}_c$. If $G^s_{\Delta,\tau},(s,v) \models \overline{\varphi}^{\,\sigma}$, we have by i.h. that there exists $f \in \mathsf{Contr}(G^s_{\Delta,\tau}, s)$ s.t. $f(G,s),(s,v) \models \varphi \wedge \langle\sigma\rangle\,\mathtt{tt}$. Then we clearly have $f(G,s),(s,v) \models \langle\delta\rangle\,\varphi \wedge \langle\sigma\rangle\,\mathtt{tt}$.

– $\Phi \stackrel{def}{=} [\delta]\,\varphi$:
  - ⇒ First assume $\sigma \in \mathsf{Act}_c$. Assume there exists $f \in \mathsf{Contr}(G^s_{\Delta,\tau}, s)$ s.t. $f(G,s),(s,v) \models [\delta]\,\varphi \wedge \langle\sigma\rangle\,\mathtt{tt}$. This implies $f(G,s),(s,v) \models \varphi \wedge \langle\sigma\rangle\,\mathtt{tt}$. By i.h. we have: $G^s_{\Delta,\tau},(s,v) \models \overline{\varphi}^{\,\sigma}$.
    Assume $\sigma = \lambda$. Then there exists $f \in \mathsf{Contr}(G^s_{\Delta,\tau}, s)$ s.t. $f(G,s),(s,v) \models [\delta]\,\varphi$, that is for any transition $f(G,s), s \xrightarrow{t} s^t$ (with $t \in \mathbb{R}$), we have $f(G,s),(s^t,v+t) \models \varphi$. We distinguish two cases:
    * Every delay transition from $G^s_{\Delta,\tau}, s$ exists also in $f(G,s), s$. Then the induction hypothesis entails that for any $t$, we have [7] $G^s_{\Delta,\tau-t}(s^t,v+t) \models \overline{\varphi}^{\,\varepsilon}$ and then clearly $G^s_{\Delta,\tau},(s,v) \models [\delta]\overline{\varphi}^{\,\varepsilon}$ and then $G^s_{\Delta,\tau},(s,v) \models \overline{\varphi}^{\,\varepsilon}\,[\delta]\,\overline{\varphi}^{\,a_c}$ for every $a_c \in \mathsf{Act}_c$.
    * There exists some delay transition in $G^s_{\Delta,\tau}$ from $s$ that does not belong to $f(G,s)$. This means that the strategy $f$ requires the execution of some controllable action $a_c$ from some state $s^t$ reachable from $s$ by a delay $t$: $f(s \xrightarrow{t} s^t) = a_c$. We then have $f(G,s),(s^{t'},v+t') \models \varphi$

[7] In the following we always assume that $\tau - t$ stands for $\max(0, \tau - t)$.

for any $t' < t$ and $f(G, s), (s^t, v + t) \models \varphi \wedge \langle a_c \rangle \, \mathtt{tt}$. By i.h. we have $G^{s^{t'}}_{\Delta,\tau-t'}, (s^{t'}, v + t') \models \overline{\varphi}^{\,\lambda}$ for any $t' < t$ and $G^s_{\Delta,0}, (s^t, v + t) \models \overline{\varphi}^{\,a_c}$. This implies $G^s_{\Delta,\tau}, (s, v) \models \overline{\varphi}^{\,\lambda} \, [\delta\rangle \, \overline{\varphi}^{\,a_c}$.

$\Leftarrow$ Assume $\sigma \in \mathsf{Act}_c$. Then $G^s_{\Delta,\tau}, (s, v) \models \overline{\varphi}^{\,\sigma}$ entails that there exists some $f \in \mathsf{Contr}(G^s_{\Delta,\tau}, s)$ s.t. $f(G, s), (s, v) \models \varphi \wedge \langle \sigma \rangle \, \mathtt{tt}$. This means that $f(s) = \sigma$ and that no delay is indeed allowed by the strategy $f$. Then we clearly have $f(G, s), (s, v) \models [\delta] \, \varphi \wedge \langle \sigma \rangle \, \mathtt{tt}$.

Assume $\sigma = \lambda$ and $G^s_{\Delta,\tau}, (s, v) \models \overline{\varphi}^{\,\lambda} \, [\delta\rangle \, \overline{\varphi}^{\,a_c}$ for some $a_c \in \mathsf{Act}_c$. We distinguish two cases:

- $G^s_{\Delta,\tau}, (s, v) \models [\delta] \, \overline{\varphi}^{\,\lambda}$. Then for any $t \in \mathbb{R}$, we have that $G^s_{\Delta,\tau} : s \xrightarrow{t} s^t$ implies $G^s_{\Delta,\tau}, (s^t, v + t) \models \overline{\varphi}^{\,\lambda}$ and then by i.h. there exists $f_t \in \mathsf{Contr}(G^s_{\Delta,\tau-t}, s^t)$ s.t. $f_t(G, s^t), (s^t, v + t) \models \varphi$ and $f_t(s^t) = \lambda$. Let $f$ be the strategy defined by $f(s \xrightarrow{t} s^t) = \lambda$ and $f(s \xrightarrow{t} \rho) = f_t(\rho)$ for any run $\rho$ starting in state $s^t$. Clearly $f$ belongs to $\mathsf{Contr}(G^s_{\Delta,\tau}, s)$. And we have $f(G, s), (s, v) \models [\delta] \, \varphi$.
- There exists $t \in \mathbb{R}$ s.t. $G^s_{\Delta,\tau} : s \xrightarrow{t} s^t$ and $G^s_{\Delta,\tau-t}, (s^t, v + t) \models \overline{\varphi}^{\,a_c}$ for some $a_c \in \mathsf{Act}_c$. By i.h. there exists $f_t \in \mathsf{Contr}(G^s_{\Delta,\tau-t}, s^t)$ s.t. $f_t(G, s^t), (s^t, v + t) \models \varphi \wedge \langle a_c \rangle \, \mathtt{tt}$. Clearly $f_t(s) = a_c$ and $f_t$ forbids time elapsing from $s^t$. Moreover the i.h. applied to states $s^{t'}$ with $t' < t$ gives that there exists $f_{t'} \in \mathsf{Contr}(G^s_{\Delta,\tau-t}, s^{t'})$ s.t. $f_{t'}(G, s^{t'}), (s^{t'}, v + t') \models \varphi$ and $f_{t'}(s^{t'}) = \lambda$. Let $f$ be the strategy defined by: $f(s \xrightarrow{t'} s^{t'}) = \lambda$ for any $t' < t$, $f(s \xrightarrow{t} s^t) = a_c$ and $f(s \xrightarrow{t''} \rho s^{t''}) = f_{t''}(\rho)$ for $t'' \leq t$. This strategy allows us to deduce $f(G, s), (s, v) \models [\delta] \, \varphi$.

– $\Phi \stackrel{def}{=} \bigvee_i \varphi_i$: Direct.

– $\Phi \stackrel{def}{=} \bigwedge_i \varphi_i$:

$\Rightarrow$ Assume there exists $f \in \mathsf{Contr}(G^s_{\Delta,\tau}, s)$ s.t. $f(G, s), (s, v) \models \bigwedge_i \alpha_i \wedge \langle \sigma \rangle \mathtt{tt}$. Then we have $f(G, s), (s, v) \models \alpha_i \wedge \langle \sigma \rangle \, \mathtt{tt}$ for any $i$. By h.i. we have $G^s_{\Delta,\tau}, (s, v) \models \overline{\alpha_i}^{\,\sigma}$ for any $i$, and then $G^s_{\Delta,\tau}, (s, v) \models \bigwedge_i \overline{\alpha_i}^{\,\sigma}$.

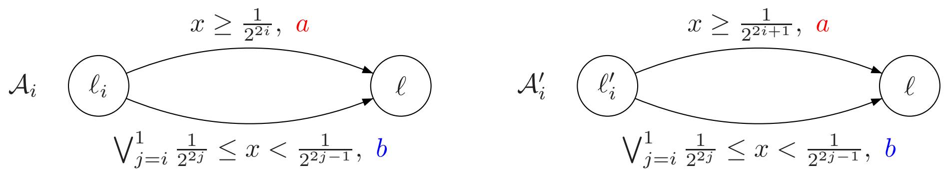
$\Leftarrow$ Assume $G^s_{\Delta,\tau}, (s, v) \models \bigwedge_i \overline{\varphi}^{\,\sigma}$. Then by i.h. we have that there exists some $f_i \in \mathsf{Contr}(G^s_{\Delta,\tau}, s)$ s.t. $f_i(G, s), (s, v) \models \alpha_i \wedge \langle \sigma \rangle \, \mathtt{tt}$. It remains to construct a strategy $f$ by collecting the strategies $f_i$'s. This is possible because $\varphi$ belongs to $L^{\text{det}}_\nu$, indeed any term $\alpha_i$ is prefixed by a modality with a different label of $\mathsf{Act} \cup \{\delta\}$ and then the union of the strategies $f_i$'s provides a strategy $f$ that belongs to $\mathsf{Contr}(G^s_{\Delta,\tau}, s)$. This gives the result.

Then this entails that the Lemma holds for any $L^{\text{det}}_\nu$ formula without fixedpoint. But this clearly entails that it also holds for full $L^{\text{det}}_\nu$. Indeed consider two states $s$ and $s'$ which satisfy the same formulae without fixedpoint. If $s$ does not belong to the greatest fixedpoint of a an equation $Z \stackrel{def}{=} \Psi_Z$, then it entails that this state does not satisfy some unfolding of the formula $\Psi_Z$ (with where the first occurrences of $Z$ have been replaced by $\mathtt{tt}$), then this formula does not hold for the state $s'$. $\square$

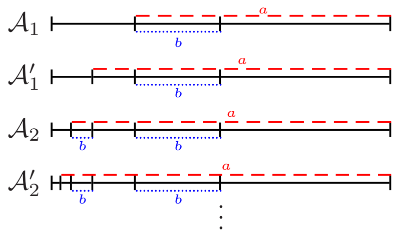
# B Proof of Theorem 2

Let us define $\varphi = ([a]\, \mathtt{ff})\, [\delta\rangle\, (\langle b\rangle\, \mathtt{tt})$. We assume that we can express $\varphi$ over timed automata as a formula $\Phi$ of $L_\nu$. This means that for every timed automaton $\mathcal{A}$, $\mathcal{A} \models \varphi \iff \mathcal{A} \models \Phi$.

***The two families of models.*** For each $i \geq 1$, we define two timed automata as follows:




The behaviors of automata $\mathcal{A}_i$ and $\mathcal{A}'_i$ can be represented by (and infered from) the following picture.




It is easy to verify that for each $i$,

$$\begin{cases} \mathcal{A}_i \models \varphi \text{ for each } i \geq 0 \\ \mathcal{A}'_i \not\models \varphi \text{ for each } i \geq 0 \end{cases}$$

***Eliminating constants from formula $\Phi$ ($\longrightarrow \Phi_1$).*** Time in all $\mathcal{A}_i$'s and $\mathcal{A}'_i$'s is bounded by 1. Let $m$ be an integer greater than the granularity of the constants appearing in $\Phi$. As $\varphi$ is an untimed formula, we get the following lemma:

**Lemma 2.** *For every positive ($> 0$) rational $p$, for every timed automaton $\mathcal{A}$, $\mathcal{A} \models \varphi$ iff $\mathcal{A} \models \Phi$ iff $\mathcal{A} \models \Phi[c \leftarrow c.p]$ where $\Phi[c \leftarrow c.p]$ is the formula $\Phi$ where each constant $c$ is replaced by the product $c.p$.*

We can assume that $\Phi$ has only constants greater than 1 (or equal to 0).

**Lemma 3.** *Under the previous assumption, we have that for every 1-bounded timed automaton $\mathcal{A}$, $\mathcal{A} \models \Phi \iff \mathcal{A} \models \Phi[x \bowtie c \leftarrow \mathit{truth}(1 \bowtie c)$ when $c > 0]$.*

The new formula is denoted $\Phi_1$ and is *a priori* not equivalent to $\Phi$ over all timed automata, but at least over 1-bounded timed automata, and in particular over all automata $A_i$'s and $A'_i$'s, which is sufficient for what we want to prove.
Note that the formula $\Phi_1$ can be generated by the following grammar:

$$\varphi ::= p \mid \mathtt{tt} \mid \mathtt{ff} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid x \ \underline{\mathsf{in}}\ \varphi \ \mid\ x \bowtie 0 \mid [a]\,\varphi \mid \langle a\rangle\,\varphi \mid [\delta]\,\varphi \mid \langle\delta\rangle\,\varphi \mid X \quad (3)$$

***Eliminating clock subformulas.*** The only "clock" information in the formula $\Phi_1$ are formulas of the form "$x \sim 0$" and "$x$ $\underline{\mathsf{in}}$ $\cdot$". We want to get rid off this information. For each $C \subseteq X$, we define inductively new formulas $\{\varphi\}^{(C)}$ as follows:

$$
\begin{aligned}
\{\alpha\}^{(C)} &= \alpha \quad \text{if } \alpha \in \{p, \mathtt{tt}, \mathtt{ff}\} \\
\{\varphi_1 \, op \, \varphi_2\}^{(C)} &= \{\varphi_1\}^{(C)} \, op \, \{\varphi_2\}^{(C)} \text{ if } op \in \{\wedge, \vee\} \\
\{x \ \underline{\mathsf{in}} \ \varphi\}^{(C)} &= x \ \underline{\mathsf{in}} \ \{\varphi\}^{(C \cup \{x\})} \\
\{[a]\varphi\}^{(C)} &= [a]\{\varphi\}^{(C)} \\
\{\langle a\rangle\varphi\}^{(C)} &= \langle a\rangle\{\varphi\}^{(C)} \\
\{X\}^{(C)} &= X_C
\end{aligned}
\qquad
\begin{aligned}
\{x > 0\}^{(C)} &= \begin{cases} \mathtt{tt} \text{ if } x \notin C \\ \mathtt{ff} \text{ if } x \in C \end{cases} \\
\{x = 0\}^{(C)} &= \begin{cases} \mathtt{tt} \text{ if } x \in C \\ \mathtt{ff} \text{ if } x \notin C \end{cases} \\
\{[\delta] \, \varphi\}^{(C)} &= \{\varphi\}^{(C)} \wedge [\delta]^{+}\{\varphi\}^{(\emptyset)} \\
\{\langle\delta\rangle \, \varphi\}^{(C)} &= \{\varphi\}^{(C)} \wedge \langle\delta\rangle^{+}\{\varphi\}^{(\emptyset)}
\end{aligned}
$$

Intuitively the formula $\{\varphi\}^{(C)}$ expresses the fact that $\varphi$ holds while the value of clocks in $C$ is 0 and the value of clocks not in $C$ are strictly greater than 0. The formula $\langle\delta\rangle^{+}\varphi$ is similar to that of $\langle\delta\rangle\,\varphi$ but the delay must be positive. Similarly $[\delta]^{+}\varphi$ means that for all positive delay, $\varphi$ must hold. We do not redefine formally these two operators.

**Lemma 4.** *For each timed automaton $\mathcal{A}$, for each formula $\varphi$ generated by the grammar (3), for each extended configuration $(\ell, u :: v)$,*

$$(\ell, u :: v) \models \varphi \iff (\ell, u :: v) \models \{\varphi\}^{(C)}$$

*where $C = \{x \in X \mid v(x) = 0\}$. In particular, in the initial configuration ($u_0 = \mathbf{0}$ and $v_0 = \mathbf{0}$),*

$$(\ell_0, u_0 :: v_0) \models \varphi \iff (\ell_0, u_0 :: v_0) \models \{\varphi\}^{(X)}$$

The new formula $\{\Phi_1\}^{(X)}$ has no more clock constraints. We can thus erase all operators "$x$ $\underline{\mathsf{in}}$ $\cdot$" because clocks are no more used. We get a new formula $\Phi_2$ (without clocks) which is generated by the grammar

$$\varphi ::= p \mid \mathtt{tt} \mid \mathtt{ff} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid [a]\,\varphi \mid \langle a\rangle\,\varphi \mid [\delta]^{+}\varphi \mid \langle\delta\rangle^{+}\varphi \mid X \qquad (4)$$

and thus such that for every 1-bounded timed automata $\mathcal{A}$, for every configuration $(\ell, u)$ of $\mathcal{A}$, $(\ell, u) \models \varphi \iff (\ell, u) \models \Phi_2$ ($\Phi_2$ is equivalent to $\varphi$ over 1-bounded timed automata).

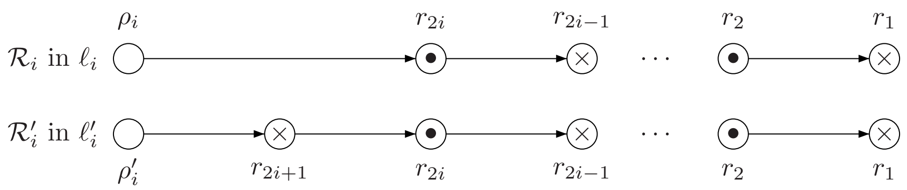
***Region abstraction.*** The regions for automaton $\mathcal{A}_i$ in state $\ell_i$ are the intervals:

$$\left[0, \frac{1}{2^{2i}}\right[, \ \left[\frac{1}{2^{2i}}, \frac{1}{2^{2i-1}}\right[, \ \ldots \ , \left[\frac{1}{2}, 1\right[$$

whereas the regions for automaton $\mathcal{A}'_i$ in state $\ell'_i$ are the intervals:

$$\left[0, \frac{1}{2^{2i+1}}\right[, \ \left[\frac{1}{2^{2i+1}}, \frac{1}{2^{2i}}\right[, \ \ldots \ , \left[\frac{1}{2}, 1\right[$$

For all automata, only one region is needed in state $\ell$, it is the interval $[0, 1[$. It is thus only necessary to see what happens in states $\ell_i$ and $\ell'_i$. All regions are right-open. We denote by $\mathcal{R}_i$ the region automaton for $\mathcal{A}_i$ and $\mathcal{R}'_i$ the region automaton for $\mathcal{A}'_i$. We can redraw automata $\mathcal{R}_i$ and $\mathcal{R}'_i$ (restricted to states $\ell_i$ and $\ell'_i$) as follows:




We have that $r_k = \left[\frac{1}{2^k}, \frac{1}{2^{k-1}}\right[$, that $\rho_k = \left[0, \frac{1}{2^{2k}}\right[$ and $\rho'_k = \left[0, \frac{1}{2^{2k+1}}\right[$.
Two states which have the same label (for example $r_i$) satisfy exactly the same formulas of (4) because they are obtained similarly by a backward computation. The labelling "•" means that both $a$ and $b$ can be done (leading to state $\ell$) whereas the labelling "×" means that only action $a$ can be done (also leading to state $\ell$). Finally no labelling means that no action can be done.

**Lemma 5.** *Let us fix some formula $\psi$ generated by grammar (4).*

- *Let us fix some region $r_i$ and two valuations $u$ and $v$ in $r_i$. Then for every $j$ such that $2j \geq i$ and for every $k$ such that $2k+1 \geq 1$,*

$$(\ell_j, u) \models \psi \Leftrightarrow (\ell_j, v) \models \psi \iff (\ell'_k, u) \models \psi \iff (\ell'_k, v) \models \psi$$

- *Moreover, for all valuations $u$ and $v$ in $\rho_i$, $(\ell_i, u) \models \psi \iff (\ell_i, v) \models \psi$*
- *Finally, for all valuations $u$ and $v$ in $\rho'_i$, $(\ell'_i, u) \models \psi \iff (\ell'_i, v) \models \psi$*

The logic (4) can thus be interpreted over regions: if $\psi$ is generated by (4), and $(\ell, r)$ is a "region state", then $(\ell, r) \models \psi \overset{\text{def}}{\iff} \exists u \in r$ s.t. $(\ell, u) \models \psi$. And the following equivalence holds: $(\ell, r) \models \psi \overset{\text{def}}{\iff} \forall u \in r$ s.t. $(\ell, u) \models \psi$.

***Atomic propositions.*** As already noticed above, formulas $[e]\,\varphi$, $\langle e \rangle\,\varphi$ (with $e$ being either $a$ or $b$) can be viewed as atomic formulas (which justifies notations "•" and "×").
Let us fix a subformula $[a]\,\psi$. Evaluate $[\![\psi]\!]$ in state $\ell$. As there is only one region in $\ell$, it holds that

$$[\![\psi]\!] \cap \{(\ell, u) \mid u \in [0,1[\} = \begin{cases} \{(\ell, u) \mid u \in [0,1[\} \\ \text{or } \emptyset \end{cases}$$

First note that for each valuation $u \in [0,1[$, $(\ell, u) \models [a]\,\psi$ because no $a$ can be done, and similarly $(\ell, u) \models [a]\,\mathtt{ff}$, and $(\ell, u) \models \mathtt{tt}$. Considering the first case, it is easy to prove that for each valuation $u \in [0,1[$, $(\ell_i, u) \models [a]\,\psi$, in which case, we can replace the subformula $[a]\,\psi$ by $\mathtt{tt}$. In the second case, for each $u \in [0,1[$, we have that $(\ell_i, u) \models [a]\,\psi \iff (\ell_i, u) \models [a]\,\mathtt{ff}$, in which case we can replace the subformula $[a]\,\psi$ by $[a]\,\mathtt{ff}$.
Let us fix a subformula $\langle a \rangle\,\psi$. Evaluate $[\![\psi]\!]$ in state $\ell$. As there is only one region in $\ell$, it holds that

$$[\![\psi]\!] \cap \{(\ell, u) \mid u \in [0,1[\} = \begin{cases} \{(\ell, u) \mid u \in [0,1[\} \\ \text{or } \emptyset \end{cases}$$

First note that for each valuation $u \in [0,1[$, $(\ell, u) \not\models \langle a\rangle\, \psi$ because no $a$ can be done. Considering the second case, it is easy to prove that for each valuation $u \in [0,1[$, $(\ell_i, u) \not\models \langle a\rangle\, \psi$, in which case, we can replace the subformula $\langle a\rangle\, \psi$ by $\mathtt{ff}$. In the first case, for each $u \in [0,1[$, we have that $(\ell_i, u) \models \langle a\rangle\, \psi \iff (\ell_i, u) \models \langle a\rangle\, \mathtt{tt}$, in which case we can replace the subformula $\langle a\rangle\, \psi$ by $\langle a\rangle\, \mathtt{tt}$.

With all these remarks we define a new formula $\Phi_4$ from $\Phi_3$ by rewriting a subformula $[a]\, \psi$ into $[a]\, \mathtt{ff}$ if $\psi$ does not hold in $\ell$ and into $\mathtt{tt}$ if $\psi$ holds in $\ell$, and by rewriting a subformula $\langle a\rangle\, \psi$ by $\mathtt{ff}$ if $\psi$ does not hold in $\ell$ and by $\langle a\rangle\, \mathtt{tt}$ if $\psi$ holds in $\ell$. The new formula $\Phi_4$ can be generated by the grammar:

$$\varphi ::= p \mid \mathtt{tt} \mid \mathtt{ff} \mid [a]\, \mathtt{ff} \mid \langle a\rangle\, \mathtt{tt} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid [\delta]^{+}\varphi \mid \langle\delta\rangle^{+}\varphi \mid X \qquad (5)$$

For each $u \in [0,1[$, $(\ell_i, u) \models \varphi \iff (\ell_i, u) \models \Phi_4$ and $(\ell'_i, u) \models \varphi \iff (\ell'_i, u) \models \Phi_4$. We could say that $\Phi_4$ is equivalent to $\varphi$ over all automata $\mathcal{A}_i$'s and $\mathcal{A}'_i$'s.

***Untiming the formula.*** We note $\mathcal{R}_i$ (resp. $\mathcal{R}'_i$) the set of regions in state $\ell_i$ (resp. $\ell'_i$) for the automaton $\mathcal{A}_i$ (resp. $\mathcal{A}'_i$). We define a new logic by the grammar:

$$\varphi ::= \mathtt{tt} \mid \mathtt{ff} \mid [a]\, \mathtt{ff} \mid \langle a\rangle\, \mathtt{tt} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathsf{G}^{+}\varphi \mid \mathsf{F}^{+}\varphi \mid X \qquad (6)$$

This logic is interpreted on the region automata $\mathcal{R}_i$'s and $\mathcal{R}'_i$'s. The semantics (denoted $\vdash$) is defined inductively by:

$$\begin{aligned}
(\{\ell_i, \ell'_i\}, r) \vdash \{\mathtt{tt}, \mathtt{ff}, [a]\, \mathtt{ff}, \langle a\rangle\, \mathtt{tt}\} &\iff (\{\ell_i, \ell'_i\}, r) \models \{\mathtt{tt}, \mathtt{ff}, [a]\, \mathtt{ff}, \langle a\rangle\, \mathtt{tt}\} \\
(\{\ell_i, \ell'_i\}, r) \vdash \{\varphi_1 \wedge \varphi_2, \varphi_1 \vee \varphi_2\} &\iff (\{\ell_i, \ell'_i\}, r) \vdash \varphi_1 \ \{\text{and}, \text{or}\}\ (\{\ell_i, \ell'_i\}, r) \vdash \varphi_2 \\
(\{\ell_i, \ell'_i\}, r) \vdash \mathsf{G}^{+}\varphi &\iff \forall r' \in \operatorname{Succ}(r),\ r \neq r', (\{\ell_i, \ell'_i\}, r') \vdash \varphi \\
(\{\ell_i, \ell'_i\}, r) \vdash \mathsf{F}^{+}\varphi &\iff \exists r' \in \operatorname{Succ}(r),\ r \neq r', (\{\ell_i, \ell'_i\}, r') \vdash \varphi \\
(\{\ell_i, \ell'_i\}, r) \vdash X &\iff (\{\ell_i, \ell'_i\}, r) \vdash \operatorname{def}(X)
\end{aligned}$$

Operators $[\delta]^{+}\psi$ and $G^{+}\psi$ interpreted over regions are not so different: the only difference stands in that the current region may not satisfy property $\psi$. In the same way, operators $\langle\delta\rangle^{+}\psi$ and $F^{+}\psi$ interpreted over regions are not so different: the only difference stands in that a region different from the current one has to satisfy $\psi$.

We rewrite the formula $\Phi_4$ into $\Phi_5$ by replacing each subformula $[\delta]^{+}\psi$ by $\psi \wedge G^{+}\psi$ and each subformula $\langle\delta\rangle^{+}\psi$ by $\psi \vee F^{+}\psi$. It is then easy to prove the following lemma:

**Lemma 6.** *For each $i$, for each region $r \in \mathcal{R}_i$, $(\ell_i, r) \models \Phi_4 \iff (\ell_i, r) \vdash \Phi_5$. Similarly, for each $i$, for each region $r' \in \mathcal{R}'_i$, $(\ell'_i, r') \models \Phi_4 \iff (\ell'_i, r') \vdash \Phi_5$.*

The formula $\Phi_5$ is now an (fully) untimed formula. Moreover, $\Phi_5$ is equivalent to $\varphi$ over the two families of automata $(\mathcal{A}_i)_{i \geq 1}$ and $(\mathcal{A}'_i)_{i \geq 1}$. We finally note $\Psi = \Phi_5$.

***Gluing everything.*** The formula $\Psi$ can be written in normal form as a system of equations $(X_i = f_i(X_1, ..., X_n))_{1 \leq i \leq n}$ and $\Psi = X_1$. We assume that each formula $f_i(X_1, ..., X_n)$ is a boolean combination of subformulas $\alpha_i^j$ (which can be either some formula $\mathsf{F}^+\beta_i^j$, or $\mathsf{G}^+\beta_i^j$, or some atomic-like formula $\langle a\rangle\, \mathtt{tt}$, $[a]\, \mathtt{ff}$, $\mathtt{tt}$ or $\mathtt{ff}$, or some fix-point variable $X_i^j$):

$$\begin{cases} X_1 =_\nu b_1(\alpha_1^1, ..., \alpha_1^{k_1}) \\ \quad\vdots \\ X_n =_\nu b_n(\alpha_n^1, ..., \alpha_n^{k_n}) \end{cases}$$

Without loss of generality we assume that no subformula $\alpha_i^j$ is a fix-point variable $X_k$. The following lemma justifies this fact:

**Lemma 7.** *We assume that $\alpha_i^j = X_k$ (with $i \neq k$). Then the new formula obtained by replacing $X_k$ by its definition formula is equivalent to the previous formula. If $\alpha_i^j = X_i$, then the new formula obtained by replacing this variable $X_i$ by $\mathtt{tt}$ is equivalent to the initial formula.*

Thus, each $\alpha_i^j$ is either an atomic proposition, or its negation, or a formula $\mathsf{F}^+\varphi$ or a formula $\mathsf{G}\varphi$.

**Lemma 8.** *The following implications are true.*

– $\mathcal{R}_i \vdash \mathsf{F}^+\varphi$ *implies* $\mathcal{R}'_i \vdash \mathsf{F}^+\varphi$ — $\mathcal{R}_i \not\vdash \mathsf{G}^+\varphi$ *implies* $\mathcal{R}'_i \not\vdash \mathsf{G}^+\varphi$
– $\mathcal{R}_i \vdash \mathsf{G}^+\varphi$ *implies* $\mathcal{R}'_{i-1} \vdash \mathsf{G}^+\varphi$ — $\mathcal{R}_i \not\vdash \mathsf{F}^+\varphi$ *implies* $\mathcal{R}'_{i-1} \not\vdash \mathsf{F}^+\varphi$
– $\mathcal{R}'_i \vdash \mathsf{F}^+\varphi$ *implies* $\mathcal{R}'_{i+1} \vdash \mathsf{F}^+\varphi$

There is a sequence $(\gamma_j)_{1 \leq j \leq k_1} \in \{\mathtt{tt}, \mathtt{ff}\}^{k_1}$ and an infinite sequence of indexes $I$ such that $b_1(\alpha_1^1 \leftarrow \gamma_1, ..., \alpha_1^{k_1} \leftarrow \gamma_{\ell_1})$ is true and for each $i \in I$, for each $1 \leq j \leq k_1$, $\gamma_j = \mathtt{tt}$ iff $\mathcal{R}_i \models \alpha_1^j$.
We note $\alpha_{\mathsf{F}}$ the set $\{\alpha_1^i \mid \gamma_i = \mathtt{tt} \text{ and } \alpha_1^i = \mathsf{F}^+*\}$, $\alpha_{\neg\mathsf{F}}$ the set $\{\alpha_1^i \mid \gamma_i = \mathtt{ff} \text{ and } \alpha_1^i = \mathsf{F}^+*\}$, $\alpha_{\mathsf{G}}$ the set $\{\alpha_1^i \mid \gamma_i = \mathtt{tt} \text{ and } \alpha_1^i = \mathsf{G}*\}$, and $\alpha_{\neg\mathsf{G}}$ the set $\{\alpha_1^i \mid \gamma_i = \mathtt{ff} \text{ and } \alpha_1^i = \mathsf{G}*\}$.
From the first line of implications of Lemma 8, for every $i \in I$, $\mathcal{R}'_i \models \bigwedge \alpha_{\mathsf{F}} \wedge \bigwedge \alpha_{\neg\mathsf{G}}$. We can even simplify: let's take some $i_0 \in I$, we have that for every $i \geq i_0$, $\mathcal{R}'_i \models \bigwedge \alpha_{\mathsf{F}} \wedge \bigwedge \alpha_{\neg\mathsf{G}}$. From the second line of implications, for every $i \in I$ such that $i > i_0$, we have that $\mathcal{R}'_{i-1} \models \bigwedge \alpha_{\mathsf{G}} \wedge \bigwedge \alpha_{\neg\mathsf{F}}$. Finally we can find some $i$ such that $\mathcal{R}'_i \models \bigwedge \alpha_{\mathsf{F}} \wedge \bigwedge \alpha_{\neg\mathsf{G}} \wedge \bigwedge \alpha_{\mathsf{G}} \wedge \bigwedge \alpha_{\neg\mathsf{F}}$ and thus $\mathcal{R}'_i \models \Psi$ (because for all possible simple terms, there is no problem). $\square$