The Logic WT_mu

Omer Landry Nguena Timo

The Logic WT_μ

Omer-Landry Nguena-Timo
Université Bordeaux 1, LaBRI, CNRS
351 cours de la Libération, 33400 Talence-FRANCE
omer-landry.nguena-timo@labri.fr

Abstract

The power of Model-checking depends on the expressive power of models of systems and models of specifications. The paper introduces $\mathrm{WT}\mu$, a real-time logic with the least and the greatest fixpoint operators. WT_μ is a weak timed extension of the μ-calculus; it is closed to L_ν. As Event-recording logic, WT_μ describes properties on Event-recording automata.

We show that WT_μ is more expressive than Eevent-recording logic. In particular, with $\mathrm{WT}\mu$ formulas, one can require occurrences of an event at all the time instants that satisfies a timing constraint. We provide an exponential-time decision procedure for the model-checking of WT_μ.

1 Introduction

The power of Model-checking depends on the expressive power of models of systems and models of specifications. Our goal is to present a new expressive fixpoint logic for describing properties on a class of real-time systems. A significant property that our logic is able to describe the requirement of the occurrence an event in all the time satisfying a timing constraint (*necessity modal operator*). We argue that such a kind of property can not be described with Event-recording logic (ERL) [Sor02] that has been introduced by Sorea for describing property the same class of real-time systems.

Real-time systems are modeled with *timed processes*. Timed processes are nothing else but event-recording automata [AFH99] without an acceptance condition. Timed processes have local clocks each associated to an event and such a clock gathers the time elapsed since the last occurrence of the corresponding event. A timed process is a finite state labelled transition system whose transitions ($p \xrightarrow{g,a} p'$) are labelled with constraints on clocks and events. A constraint on clocks is just a conjunction of comparisons of values of a clock with an integer constant.Clock are interpreted over real numbers. The value of each clock grows continuously and with the same rate as the time unless it is reset. When the process is in some state, the time elapses continuously (the values of the clocks too) until an event occurs. Then, the process instantaneously selects a transition labelled with that event and checks whether the constraint (g) on the chosen transition is satisfied by the values of clocks before it resets the clock associated to the event and moves to the target state of the transition. If the constraint is not satisfied, the process does not change the state.

The logic that we introduce in this paper is called WT_μ. The logic WT_μ is a weak timed extension of the standard μ-calculus. Formulas of WT_μ are interpreted over timed processes. Timed processes are nothing else but event-recording automata without an

acceptance condition. The modalities of the logic are either indexed with constraints or events, while modalities of ERL are indexed with pairs made of a constraint and an event. They are of WT_μ are of the form $\langle g\rangle$ and $[g]$ in addition to the classical modalities of the μ-calculus indexed with event ($\langle a\rangle$ and $[a]$). Intuitively, a state of a timed process p satisfies $\langle g\rangle\varphi$ from a given time-context described by a valuation v if by letting time elapse in it, it is possible to reach a moment when the values of the clocks satisfy g and in that moment, the formula φ is satisfied. A state p of a timed process satisfies $[g]\varphi$ from a time-context v if whenever starting from v we let the time pass and reach a moment when g is satisfied then φ is satisfied in that moment. We consider the model-checking problem for WT_μ; that is: Does a timed process satisfy a WT_μ formula. We provide an exponential-time decision procedure for that problem.

We compare WT_μ with ERL. ERL is also presented [Sor02] as a timed extension of the μ-calculus; and models of ERL formulas are timed processes. In ERL, modalities are indexed both with an event and a constraint ($[g,a]$, $\langle g,a\rangle$). A state of a timed process p satisfies $\langle g,a\rangle\varphi$ from a given time-context described by a valuation v if by letting time elapse in it, it is possible that the event a occurs in a moment when the values of the clocks satisfy g and after the occurrence of a, the process goes to a state that satisfies φ. A state of a timed process p satisfies $[g,a]\varphi$ from a given time-context described by a valuation v if after the occurrence of a in a moment when the values of the clocks (obtained by letting time elapses in v) satisfy g the process always goes to a state that satisfies φ. We will show that WT_μ is more expressive than ERL as every formula of ERL can be translated into an equivalent WT_μ formula; and there are some formulas of WT_μ that can not be translated into formulas of ERL. In particular with $\mathrm{WT}\mu$, it is possible to require the occurrence an event in all the time satisfying a timing constraint; but it is not with ERL.

Related results: Logics (TML [HLY91], L^t_μ [SS95] L_ν [LLW95]) that enable to describe the *the necessity modal operator* has been considered for describing properties on timed automata but the decidability of the satisfiability problem has not been established. Laroussinie et al. [LLW95] have introduced the logic L_ν as a more powerful logic than the one in [HLY91, SS95] but its satisfiability problem is still open and no disjunctive normal form has been provided [BCL05]. The logics L_ν and WT_μ are incomparable as they are not interpreted over the same model and L_ν does not allow the least fixpoint operator. But, if we restrict the interpretation of L_ν on timed processes, we claim that $\langle g\rangle\varphi$ will have the same meaning as the L_ν formula $\langle\delta\rangle(g\wedge\varphi)$ and $[g]\varphi$ will have the same meaning as the L_ν formula $[\delta](g\rightarrow\varphi)$.

This paper is organised as follows: We present results for the model-checking of the μ-calculus in the next section. We present time processes in Section 3. In that section we also present well known concepts and results concerning region, constraint, and timed abstract bisimulation. In Section 4 we present WT_μ and its semantics. We consider the model-checking problem for WT_μ in Section 5. In Section 6, we present ERL and we show that WT_μ is more expressive than ERL. We conclude the paper with future works on WT_μ.

2 Two Player Parity Game and μ-calculus Results

2.1 Two Player Parity Games and Multi-Parity Games

We present a complexity result for checking a winning strategy in a two player games with parity condition. We also present the notion of two multi-parity game.

Definition 1 A *two player parity game*(see [Zie98]) is a tuple $\mathcal{G} = \langle N_E, N_A, T \subseteq N^2, Acc_\mathcal{G}\rangle$ where $\langle N, T\rangle$ is a graph with the nodes (or positions) $N = N_A \cup N_E$ partitioned into N_E and N_A. N_E denotes the set of nodes of the player *Eve* and N_A denotes

the set of nodes of the player *Adam*. The winning condition $Acc_{\mathcal{G}} \subseteq N^{\omega}$, is a parity condition on the nodes. The game is finite if N is finite.

A *play* between *Eve* and *Adam* from some node $n \in N$ proceeds as follows: if $n \in N_E$ then *Eve* makes a choice of a successor otherwise *Adam* chooses a successor; from this successor the same rule applies and the play goes on forever unless one of the parties cannot make a move. A play is finite if a player cannot make a move and then he loose the play. In the case that the play is an infinite path $\pi = n_0 n_1 n_2 \cdots$, *Eve* wins if $\pi \in Acc_{\mathcal{G}}$. Otherwise *Adam* is the winner. Among winning conditions introduced in the literature, we consider the parity condition. A *strategy* σ for *Eve* is a function assigning to every sequence of nodes $\vec{n}$ ending in a node n from N_E a vertex $\sigma(\vec{n})$ which is a successor of n.

A *play from* n *consistent* with σ is a finite or infinite sequence $n_0 n_1 n_2 \cdots$ such that $n_{i+1} = \sigma(n_i)$ for all i with $n_i \in N_E$. The *strategy* σ *is winning for Eve* from the node n if and only if all the plays starting in n and consistent with σ are winning. The strategies for *Adam* is are defined similarly. A *node is winning* if there exists a strategy winning from it. A game is *determined* if every node is winning for one of the player. A strategy is *positional* if it does not depend on the sequences of nodes that were played till now, but only on the present node. So such a strategy for *Eve* can be represented as a function $\sigma : N_E \to N$ and identified with a choice of edges in the graph of the game.

Now we state the following results on two player games (see [GH82, EJ91, Jur00, VJ00]).

Theorem 2 *Every parity game is determined. In a two player parity game a player has a winning positional strategy from each of his nodes. There is an effective procedure that decides who is a winner from a given node in a finite game, and that procedure works in time*

$$\mathcal{O}\left(|T| \times \left(\frac{2 \times |N|}{d}\right)^{\lceil d/2 \rceil}\right)$$

where, d is the maximal parity index.

2.2 The μ-Calculus

The μ-calculus introduced by Kozen [Koz82] (see also [AN01]) is an expressive temporal logic that extends modal logic with the greatest (ν) and least (μ) fixpoint operators. We present the syntax and the semantics of the μ-calculus. Then we state some well known results that include the complexity of the model-checking problem, the complexity of the satisfiability problem and a disjunctive normal form theorem. The complexity result for the model-checking is obtained by reduction to checking if there is a winning strategy in a two player parity game.

2.2.1 Definitions and Semantics

Definition 3 The syntax of the μ-calculus is defined over a set $Var = \{X, Y, \ldots\}$ of variables, a set Σ of events. It is given by the following grammar:

$$\varphi ::= tt \,|\, ff \,|\, X \,|\, \varphi \vee \psi \,|\, \varphi \wedge \psi \,|\, \langle a \rangle \varphi \,|\, [a]\varphi \,|\, \mu X.\varphi(X) \,|\, \nu X.\varphi(X)$$

In the above, $X \in Var$, $a \in \Sigma$; and tt and ff denote the formula that are always "true" and "false" respectively; $\langle a \rangle$ and $[a]$ denote the existential and the universal modalities indexed with the event a; they represent "exists a-successor and "all a-successor" modalities respectively. The formulas $\mu X.\varphi(X)$ and $\nu X.\varphi(X)$ represent respectively the least and the greatest fixpoint formula.

For a formula φ, the closure [Koz82] of φ, $sub(\varphi)$ is defined as follows:

Definition 4 The *closure* $sub(\varphi)$ of φ is the smallest set of formulas such that:

- $\varphi \in sub(\varphi)$
- if $\psi_1 \vee \psi_2 \in sub(\varphi)$ the both $\psi_1, \psi_2 \in sub(\varphi)$
- if $\psi_1 \wedge \psi_2 \in sub(\varphi)$ the both $\psi_1, \psi_2 \in sub(\varphi)$
- if $\langle a \rangle \psi \in sub(\varphi)$ then $\psi \in sub(\varphi)$
- if $[a]\psi \in sub(\varphi)$ then $\psi \in sub(\varphi)$
- if $\sigma X.\psi(X) \in sub(\varphi)$ then $\psi(X) \in sub(\varphi)$, where $\sigma \in \{\nu, \mu\}$

The formulas in $sub(\varphi)$ are called the *subformulas* of φ. For a formula φ, $sub(\varphi)$ is finite and, by definition, it is not larger that the number of symbols used in φ.

Definition 5 The set $free(\varphi)$ of free variable of a μ-calculus formula φ is defined inductively as follows:

- $free(tt) = free(ff) = \emptyset$
- $free(X) = \{X\}$
- $free(\varphi \vee \psi) = free(\varphi) \cup free(\psi)$
- $free([a]\varphi) = free(\langle a \rangle \varphi) = free(\varphi)$
- $free(\mu X.\varphi(X)) = free(\nu X.\varphi(X)) = free(\varphi) \setminus \{X\}$

A variable X is *free* in a formula φ if $X \in free(\varphi)$.

Definition 6 A variable X is *bound* in a formula φ if there is a subformula $\sigma X.\psi(X)$ of φ with $\sigma \in \{\mu, \nu\}$.

Definition 7 (Well named) We call a formula *well named* if the expression $\mu X.\varphi(X)$ (or $\nu X.\varphi(X)$) occurs at most once for each variable X.

By renaming variables if necessary, every formula can be translated into an equivalent well named formula. In what follows, without loss of generality, we assume that formulas are well named.

Definition 8 (Binding) The *binding definition* of a bound variable X in a well named formula φ, $\mathcal{D}_\varphi(X)$ is the unique subformula of φ of the form $\sigma X.\psi(X)$. We will omit subscript φ when it causes no ambiguity. We call X a *μ-variable* when $\sigma = \mu$, otherwise we call X a *ν-variable*. The function $\mathcal{D}_\varphi$ assigning to every bound variable its binding definition in φ will be called the *binding function* associated with φ.

Definition 9 A *sentence* is a well named formula without free variables.

Definition 10 The *dependency order* $\leq_\varphi$ over the bound variables of a formula φ, is the least partial order such that if X occurs in $\mathcal{D}_\varphi(Y)$ (and $\mathcal{D}_\varphi(Y)$ is a sub formula of $\mathcal{D}_\varphi(X)$) then $X \leq_\varphi Y$. When $X \leq_\varphi Y$, it is also said that Y *depends on* X or X is *older than* Y.

Definition 11 Variable X in $\mu X.\varphi(X)$ is *guarded* if every occurrence of X in $\varphi(X)$ is in the scope of some modality operator $\langle\rangle$ or $[]$. We say that a *formula is guarded* if every bound variable in the formula is guarded.

Alternation depth describes the number of alternations between least and greatest fixpoint operators.

Definition 12 The *alternation depth* of a formula denoted by $alt(\varphi)$ is the number of nesting between μ and ν in φ; it is recursively defined as follows:

- $alt(tt) = alt(\mathit{ff}) = alt(X) = 0$
- $alt(\varphi \wedge \psi) = alt(\varphi \vee \psi) = \max(alt(\varphi), alt(\psi))$
- $alt(\langle a \rangle \varphi) = alt([a]\varphi) = alt(\varphi)$
- $alt(\mu X.\varphi(X)) = \max(\{1, alt(\varphi(X)\} \cup \{1 + alt(\nu Y.\psi(Y)) \,|\, \nu Y.\psi(Y) \in sub(\varphi); X \leq_\varphi Y\})$
- $alt(\nu X.\varphi(X)) = \max(\{1, alt(\varphi(X)\} \cup \{1 + alt(\mu Y.\psi(Y)) \,|\, \mu Y.\psi(Y) \in sub(\varphi); X \leq_\varphi Y\})$

Formulas of the μ-calculus are interpreted over Σ-labelled transition systems. The semantics of a μ-calculus formula φ is a set of states of a Σ-labelled transition system $\mathcal{S} = \langle S, \Sigma, s^0, \Delta_\mathcal{S} \rangle$ where the formula holds under a given valuation of variables $Val : Var \to 2^S$, and it is denoted by $[\![\varphi]\!]^\mathcal{S}_{Val}$. Given a valuation of variables Val and a set of states $T \subseteq S$, the valuation $Val[X/T]$ is the valuation Val with the substitution that associates the states of T with the variable X. Formally, for $Y \in Var$, $Val[X/T](Y) = T$ if $Y = X$ and $Val(Y)$ otherwise. We define the relation $\vDash$ between a state s of a transition system $\mathcal{S}$, a valuation Val and a formula φ. We write $\mathcal{S}, s, Val \vDash \varphi$ when the formula φ holds in s or equivalently s satisfies φ. The relation $\vDash$ is defined as follows:

- $\mathcal{S}, s, Val \vDash X$ if $s \in Val(X)$
- $\mathcal{S}, s, Val \vDash \varphi_1 \vee \varphi_2$ if $\mathcal{S}, s, Val \vDash \varphi_1$ or $\mathcal{S}, s, Val \vDash \varphi_2$
- $\mathcal{S}, s, Val \vDash \varphi_1 \wedge \varphi_2$ if $\mathcal{S}, s, Val \vDash \varphi_1$ and $\mathcal{S}, s, Val \vDash \varphi_2$
- $\mathcal{S}, s, Val \vDash \langle a \rangle \varphi$ if there is $s \xrightarrow{a} s'$ such that $\mathcal{S}, s', Val \vDash \varphi$
- $\mathcal{S}, s, Val \vDash [a]\varphi$ if for all $s \xrightarrow{a} s'$ we have $\mathcal{S}, s', Val \vDash \varphi$
- $\mathcal{S}, s, Val \vDash \mu X.\varphi(X)$ if $s \in \cap\{T \subseteq S \,|\, [\![\varphi(X)]\!]^\mathcal{S}_{Val[X/T]} \subseteq T\}$.
- $\mathcal{S}, s, Val \vDash \nu X.\varphi(X)$ if $s, \in \cup\{T \subseteq S \,|\, T \subseteq [\![\varphi(X)]\!]^\mathcal{S}_{Val[X/T]}\}$

Then we define $[\![\varphi]\!]^\mathcal{S}_{Val} = \{s \in \mathcal{S} \,|\, \mathcal{S}, s, Val \vDash \varphi\}$. It is said that a Σ-labelled transition system $\mathcal{S}$ is a *model* of a formula φ when $s^0 \in [\![\varphi]\!]^\mathcal{S}_{Val}$; in this case we write $\mathcal{S}, Val \vDash \varphi$. The valuation Val is omitted if the formula does not contains free variables.

It is known (see [Eme90] for a survey) that properties expressed in temporal logics LTL, CTL, and CTL* can be encoded as μ-calculus formulas and that there are formulas of the μ-calculus (for instance $\nu X.\langle a \rangle \langle a \rangle X$) that can not be written in CTL*.

Given two formulas φ_1 and φ_2, we often use the notation $\varphi_1 \equiv \varphi_2$ to say that φ_1 is *equivalent* to φ_2, meaning that for every labelled transition system $\mathcal{S}$ and valuation Val, $[\![\varphi_1]\!]^\mathcal{S}_{Val} = [\![\varphi_2]\!]^\mathcal{S}_{Val}$.

It is standard to consider the negation operator ($\neg$) on μ-calculus sentences. This operator is defined as follows:

- $\neg tt \equiv \mathit{ff}$
- $\neg \mathit{ff} \equiv tt$
- $\neg(\varphi_1 \wedge \varphi_2) \equiv \neg\varphi_1 \vee \neg\varphi_2$
- $\neg(\varphi_1 \vee \varphi_2) \equiv \neg\varphi_1 \wedge \neg\varphi_2$
- $\neg\langle a \rangle \varphi \equiv [a]\neg\varphi$
- $\neg[a]\varphi \equiv \langle a \rangle \neg\varphi$
- $\neg\mu X.\varphi(X) \equiv \nu X.\neg\varphi(\neg X)$
- $\neg\nu X.\varphi(X) \equiv \mu X.\neg\varphi(\neg X)$

The following proposition is standard.

Proposition 13 Given a sentence φ, a Σ-labelled transition system $\mathcal{S}$ and a valuation Val, $[\![\neg\varphi]\!]^{\mathcal{S}}_{Val} = S \setminus [\![\varphi]\!]^{\mathcal{S}}_{Val}$

Thanks to the proposition just above, we can use the negation operator can appear in μ-calculus sentences.

Let us present some results on the μ-calculus.

Proposition 14 ([Koz82]) Every formula is equivalent to some guarded formula.

2.2.2 Model-Checking Results

Informally, the task of checking whether a finite state transition system, $\mathcal{S} = \langle S, \Sigma, s^0, \Delta_{\mathcal{S}} \rangle$ is a model of a sentence φ can be seen as two player parity game whose nodes are set of tuples of the form (s, ψ) where $s \in S$ and ψ is a subformula of φ. Positions of the player *Eve* constain subformulas of one of the forms $tt, \varphi_1 \vee \varphi_2, \langle a \rangle \psi$. The other positions belong to the player *Adam*. The initial position of the game is (s^0, φ). The set of moves of the games are such that:

- There is no move from either (s, tt) or (s, ff).
- From $(s, \varphi \wedge \psi)$ as well as from $(s, \varphi \vee \psi)$ there are moves to (s, φ) and to (s, ψ).
- From $(s, [a]\varphi)$ and from $(s, \langle a \rangle \varphi)$ there are moves to $(s', \varphi$, for every s' such that $s \xrightarrow{a} s'$.
- There is a move from $(s, \sigma X.\varphi(X))$ to $(s, \varphi(X))$
- There is a move from X to $(s, \varphi(X))$ where $\mathcal{D}(X) = \sigma X.\varphi(X)$

The acceptance condition is given by the parity function $rank : Q \to \mathbb{N}$ defined by:

$$rank(\psi) = \begin{cases} 0 & \text{if } \psi \text{ is not a variable} \\ 2 \times alt(\mathcal{D}(X)) & \text{where } \varphi = X \text{ and } X \text{ is a } \nu\text{-variable} \\ 2 \times alt(\mathcal{D}(X)) + 1 & \text{where } \varphi = X \text{ and } X \text{ is a } \mu\text{-variable} \end{cases}$$

One can show that S is a model of a formula if player *Eve* has a winning strategy in the the game. This gives an intuitive idea behind the following results.

Theorem 15 ([EJ91, Tho97, Jur00]) *Let $\mathcal{S} = \langle S, \Sigma, s^0, \Delta_{\mathcal{S}} \rangle$ be a Σ-labelled transition system and let φ be a μ-calculus formula. The model-checking problem for φ and $\mathcal{S}$ is solvable in time*

$$\mathcal{O}\left(|\Delta_{\mathcal{S}}| \times |sub(\varphi)| \times \left(\frac{|S| \times |sub(\varphi)|}{\lfloor alt(\varphi)/2 \rfloor}^{\lceil alt(\varphi)/2 \rceil}\right)\right)$$

3 Timed Processes

We present timed processes as event-recording automata without acceptance [AFH99]. We firstly present the notions of region [ACD+92, LY97, AFH99, AD94] and its property. All the results presented in this section are well-known.

3.1 Clocks and Valuations

Clocks are variables evaluated over real numbers. There are two operations on time, the time elapse operation that gives the value of the clock after a delay and the reset operation that sets the value of a clocks to 0.

Let $\mathbb{R}^+$ be the set of non negative real numbers. We consider $\mathcal{H} = \{h_1, h_2, \dots\}$ a set of clocks variables (or clocks for simplicity).

Definition 16 A *valuation* on a set of clock $\mathcal{H}$ is a total function $v : \mathcal{H} \to \mathbb{R}^+$.

The symbol $\mathcal{V}$ represents the set of valuations. Given a valuation $v \in \mathcal{V}$, and a clock $h \in \mathcal{H}$, the valuation $v + t$ is defined by $[v + t](h) = v(h) + t$ and, the valuation $v[h := 0]$ is defined by $v[h := 0](h') = 0$ if $h = h'$ else $v[h := 0](h') = v(h')$. We say that a valuation v is a *successor* of a valuation v' if $v = v' + t$ for some $t \in \mathbb{R}^+$.
Example: Let $\mathcal{H} = \{h_1, h_2\}$ be a set of two clocks. In Table 1, we present some valuations on h are some valuation on $\mathcal{H}$.

$$\begin{cases} v_0(h_1) = 0 \\ v_0(h_2) = 0 \end{cases} \quad \begin{cases} v_1(h_1) = 0.35 \\ v_1(h_2) = 0.35 \end{cases} \quad \begin{cases} v_2(h_1) = 0.35 \\ v_2(h_2) = 0 \end{cases} \quad \begin{cases} v_3(h_1) = 0.85 \\ v_3(h_2) = 0.50 \end{cases}$$

$$\begin{cases} v_4(h_1) = 0 \\ v_4(h_2) = 0.50 \end{cases} \quad \begin{cases} v_5(h_1) = 0.35 \\ v_5(h_2) = 0.85 \end{cases}$$

Table 1: Examples of valuations.

These valuations are such that $v_1 = v_0 + 0.35$, $v_2 = v_1[h_2 := 0]$, $v_3 = v_2 + 0.50$, $v_4 = v_3[h_1 := 0]$, $v_5 = v_4 + 0.35$ and $v_2 = v_5[h_2 := 0]$. In Figure 1 we give another representations of these valuations in Cartesian reference.

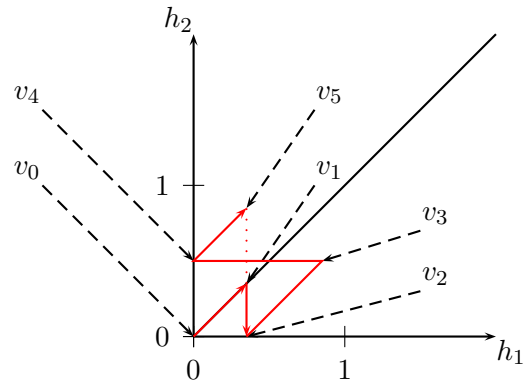

Figure 1: Representation of valuations in Cartesian reference.

□

3.2 Constraints

Constraints are conjunctions of simple constraints; and a simple constraint is a comparison of a clock with an integer (diagonal free simple constraint) or a comparison of the difference between two clocks with and integer. Diagonal free constraints use only diagonal free simple constraints. Constraints are interpreted over valuations. The semantics of a constraint is the set of valuations satisfying it. We will also consider two types of *atomic constraints* : *rectangular constraints* and triangular constraints.

Definition 17 A *simple constraint* defined on a set of clocks $\mathcal{H}$ is an equation of the form $h - h' \bowtie n$ or $h \bowtie n$ where $n \in \mathbb{N}$, $\bowtie$ is one of $\{<, \leq, \geq, >\}$ and $h, h' \in \mathcal{H}$.
A *diagonal free simple constraint* is a simple constraint of the form $h \bowtie n$.

Definition 18 A *clock constraint* over a set of clocks $\mathcal{H}$ is a conjunction of simple constraints. $\Phi_{\mathcal{H}}$, denotes the set of clock constraints over $\mathcal{H}$. A *diagonal-free clock constraint* is a clock constraint that uses only diagonal free simple constraints. $Gds_{\mathcal{H}}$ denotes the set of diagonal-free clock constraints over $\mathcal{H}$.

We will often write $h = n$ or $h - h' = n$ as an abbreviation of $h \leq n \wedge h \geq n$. We also write $h - h' = n$ to represent the constraint $h - h' \leq n \wedge h - h' \geq n$.

Later we consider two special clock constraints tt and ff defined by: $tt = \bigwedge_{h \in \mathcal{H}} h \geq 0$ and $ff = \bigwedge_{h \in \mathcal{H}} h < 0$.

The notion of a constraint satisfied in a given valuation denoted $v \vDash g$ is defined inductively as follows:

- $v \vDash h \bowtie n$ if and only if $v(h) \bowtie n$
- $v \vDash h - h' \bowtie n$ if and only if $v(h) - v(h') \bowtie n$
- $v \vDash g_1 \wedge g_2$ if and only if $v \vDash g_1$ and $v \vDash g_2$

The meaning of a constraint g, denoted $[\![g]\!]$, is the set of valuations in which it is satisfied. Clearly, $[\![g]\!] = \{v : v \vDash g\}$. It becomes obvious that $[\![tt]\!] = \mathcal{H} \to \mathbb{R}^+$ and $[\![ff]\!] = \emptyset$.

Definition 19 A constraint g is inconsistent if $[\![g]\!] = \emptyset$.

Definition 20 The *bound* of a constraint g, denoted by M_g, is the maximal constant that appears in it. The bound of a set of constraints is the maximal value among the bounds of constraint it contains. A set of constraints is M-bounded if every constant in it is smaller than M.

Now we consider atomic constraints and we show how to decompose a constraint into an "equivalent" set of atomic constraints.

Definition 21 For a integer $M \in \mathbb{N}$, a *M-rectangular constraint* is a conjunction of the form $\bigwedge_{h \in \mathcal{H}} g_h$ where g_h is a constraint of the form $c < h < c + 1$ or $h = c$ or $h > M$ with $c \in \mathbb{N} \cap [0..M[$.

The set of all M-rectangular constraints is denoted by $Agds_{\mathcal{H}}(M)$. The symbol $Agds_{\mathcal{H}}$ will denote the set $\bigcup_{M \in \mathbb{N}} Agds_{\mathcal{H}}(M)$

Definition 22 A *M-triangular constraint* is a conjunction of the form $\bigwedge_{h \in \mathcal{H}} g_h \wedge \bigwedge_{(h,h') \in \mathcal{H}^2} g_{h,h'}$ where $g_{h,h'}$ is a constraint of the forms $c < h - h' < c + 1$ or $h - h' = c$ or $h - h' > M$ and g_h is of the form $c < h < c + 1$ or $h = c$ or $h > M$ with $c \in \mathbb{N} \cap [0..M[$.

The symbol $Tgds_{\mathcal{H}}(M)$ denotes the set of all of M-triangular constraints. The symbol $Tgds_{\mathcal{H}}$ denotes the set $\bigcup_{M \in \mathbb{N}} Tgds_{\mathcal{H}}(M)$.

Notation: We often use the symbol $\hat{g}$ to denote a constraint in $Agds_{\mathcal{H}}(M)$ or $Tgds_{\mathcal{H}}(M)$ for some M. Later the terms *atomic constraints* will often be used in place of rectangular constraints or triangular constraints.

Let us first recall the following fact resulting from definitions of atomic constraints.

Fact 23 (atomicity) Let $M \in \mathbb{N}$ be a constant.

- $\forall \hat{g}, \hat{g}' \in Tgds_{\mathcal{H}}(M)$, if $[\![\hat{g}]\!] \neq [\![\hat{g}']\!]$ then $[\![\hat{g}]\!] \cap [\![\hat{g}']\!] = \emptyset$
- $\forall \hat{g}, \hat{g}' \in Agds_{\mathcal{H}}(M)$, if $[\![\hat{g}]\!] \neq [\![\hat{g}']\!]$ then $[\![\hat{g}]\!] \cap [\![\hat{g}']\!] = \emptyset$

- $\forall(\hat{g}, \hat{g}') \in Agds_{\mathcal{H}}(M) \times Tgds_{\mathcal{H}}(M)$, either $[\![\hat{g}']\!] \cap [\![\hat{g}]\!] = \emptyset$ or $[\![\hat{g}']\!] \subseteq [\![\hat{g}]\!]$

The first two items state that either the semantics of two atomic constraints of the same nature are equal, or they are disjoint. The last item of the above fact states that the semantics of a triangular constraint is either included in the semantics of a rectangular constraints, or the two semantics are disjoint.
Example: In Figure 2,we illustrate the concepts of constraints and diagonal free constraints. The constraints g_1 and g_3 are general constraints while the constraint g_2 is diagonal free. Moreover $[\![g_3]\!] = [\![g_1]\!] \wedge [\![g_2]\!]$. The constraint g_2 is a rectangular constraint

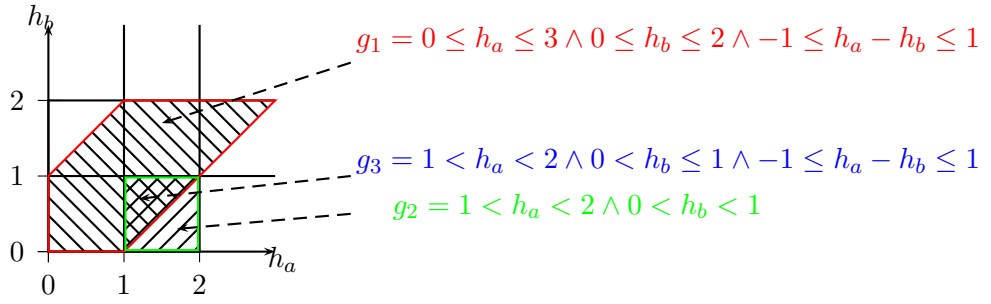

Figure 2: Illustration of constraints and diagonal free constraints.

in $Agds_{\mathcal{H}}(2)$ and the constraint g_3 is a triangular constraint. □

Normalization and Rectangularisation Until the end of this subsection we consider the decomposition of diagonal free constraint into set of rectangular constraints. We will need to consider constraints that do not involve constants greater than a fixed bound. For that purpose, we present the normalisation operation $norm_N$ that we use later to decompose constraints.

Definition 24 The *N-normalization* of a simple constraint C is the constraint $norm_N(C)$ defined by :

- $norm_N(h \bowtie n) = tt$ if $\bowtie \in \{<, \leq\}$ and $n > N$.
- $norm_N(h - h' \bowtie n) = tt$ if $\bowtie \in \{<, \leq\}$ and $n > N$.
- $norm_N(h \bowtie n) = h > N$ if $\bowtie \in \{>, \geq\}$ and $n > N$.
- $norm_N(h - h' \bowtie n) = h - h' > N$ if $\bowtie \in \{>, \geq\}$ and $n > N$.
- In the other cases $norm_N$ does not modify the constraint.

Given a constraint g and an integer N, the *N-normalization* of g, $norm_N(g)$ is obtained by normalizing each simple constraint occurring in g.

Lemma 25 Let C, a diagonal-free simple constraint, there is a constant M such that:

- for every $N \geq M$, $[\![norm_M(C)]\!] = [\![norm_N(C)]\!] = [\![C]\!]$
- for every $N < M$, $[\![norm_M(C)]\!] \subsetneq [\![norm_N(C)]\!]$

Proof

1. When C has the form $h \bowtie n$ with $\bowtie \in \{<, \leq\}$ and consider $M = n$,

(a) Let $N \geq M$, $norm_N(h \bowtie n)$ is equal to $norm_M(h \bowtie n)$ and they are equal to $h \bowtie n$ and we get the result that $[\![norm_M(C)]\!] = [\![norm_N(C)]\!] = [\![C]\!]$.

(b) Let $N < M$, $norm_N(h \bowtie n) = h \geq 0$. Clearly $[\![norm_M(C)]\!] \subsetneq [\![norm_N(C)]\!]$.

2. When C has the form $h \bowtie n$ with $\bowtie \in \{>, \geq\}$ and consider $M = n$,

(a) Let $N \geq M$, $norm_N(h \bowtie n)$ is equal to $norm_M(h \bowtie n)$ and they are equal to $h \bowtie n$ and we get the result that $[\![norm_M(C)]\!] = [\![norm_N(C)]\!] = [\![C]\!]$.

(b) Let $N < M$, then $norm_N(h \bowtie n) = h \bowtie N$ and $[\![norm_M(C)]\!] = h \bowtie M$. Clearly, $[\![norm_M(C)]\!] \subsetneq [\![norm_N(C)]\!]$.

□

Let us recall that for a constraint g, M_g denotes the maximal constant occurring in g. We use the lemma above to show that the M-normalisation of a constraint does modify its semantics when M is greater or equal to M_g.

Proposition 26 Let $g \in Gds_{\mathcal{H}}$,

- for every $M \geq M_g$, $[\![norm_M(g)]\!] = [\![norm_N(g)]\!] = [\![g]\!]$
- for every $M < M_g$, $[\![norm_M(g)]\!] \subsetneq [\![norm_N(g)]\!]$

Proof
By definitions $g = \bigwedge_{i=1..n} C_i$ and, $[\![norm_M(g)]\!] = \bigcap_{i=1..n} [\![Norm_M(C_i)]\!]$. As M_g is greater that the constant used in every C_i, we get, using 25 that for $M \geq M_g$, $[\![norm_M(g)]\!] = [\![norm_N(g)]\!] = [\![g]\!]$and for $M < M_g$, $[\![norm_M(g)]\!] \subsetneq [\![norm_N(g)]\!]$ □

Example: Considering the constraint $g = 0 \leq h_a \leq 3 \wedge 0 \leq h_b \leq 2$, we present in Table 2 the results of M-normalisation operations depending on the value of M. It is easy to see

M	$norm_M(g)$
0	tt
1	tt
2	$0 \leq h_b \leq 2$
3	$0 \leq h_a \leq 3 \wedge 0 \leq h_b \leq 2$

Table 2: Illustration of the normalisation operation.

that for every $M < 2$, $[\![g]\!] \subseteq [\![norm_M(g)]\!]$ and for every $M \geq 2$, $[\![g]\!] = [\![norm_M(g)]\!]$ □

To obtain the decomposition of diagonal constraints, we firstly decompose diagonal free constraints into a set (possibly infinite) of unbounded rectangular constraints. Then, we use the normalisation procedure above on each atomic constraint in that set to have a finite set of bounded rectangular constraints. The decomposition of diagonal free constraints into a set of unbounded rectangular constraints is performed in two steps: in Lemma 27 we decompose simple diagonal free constraints and we use that decomposition in Proposition 28 to decompose diagonal free constraints.

Lemma 27 For every diagonal free simple constraint C, there is a set $Rect(C)$ of atomic diagonal free simple constraints such that $[\![C]\!] = \bigcup_{C' \in Rect(C)} [\![C']\!]$.

Proof
Let C be a diagonal free constraint C. We construct a set $Rect(C)$ depending on the form of C; and we show that for every $v \in \mathcal{V}$, $v \vDash C$ if and only if there is $C' \in Rect(C)$ such that $v \vDash C'$.

1. if C is of the form $h < n$ then set $Rect(C) = \{i < h < i + 1, h = i \mid i = 0..n - 1\}$

2. if C is of the form $h \leq n$ then set $Rect(C) = \{i < h < i+1, h = i \,|\, i = 0..n-1\} \cup \{h = n\}$
3. if C is of the form $h > n$ then set $Rect(C) = \{i < h < i+1, h = i+1 \,|\, i = n..\infty\}$
4. if C is of the form $h \geq n$ then set $Rect(C) = \{i < h < i+1, h = i+1 \,|\, i = n..\infty\} \cup \{h = n\}$

The proof that in each case, $[\![C]\!] = \cup_{C' \in Rect(C)} [\![C']\!]$, is obvious. □

We observe that simple constraints of the form $h > n$ to $h \geq n$ are decomposed into infinite set of constraints.

Proposition 28 For every diagonal-free constraint g, there is a set $Rect(g)$ of rectangular constraints such that $[\![g]\!] = \bigcup_{\hat{g} \in Rect(g)} [\![\hat{g}]\!]$.

Proof
The result is a consequence of the Lemma 27 above as a constraints is a conjunction of simple constraints. □

We say that $Rect(g)$ is the unbounded rectangular decomposition of g.

Now that we have decomposed diagonal free constraints into sets (possibly infinite) of unbounded rectangular constraints, we will apply the normalisation operation on each rectangular constraint in these sets; the result of the application of the normalisation operation with respect to a constant M will be finite set of M-rectangular constraints. But we need to show that the semantics of the constraint resulting from the application of the M-normalisation operation on a simple diagonal free constraint is the same as the union of the semantics of rectangular constraints in its unbounded rectangular decomposition.

Lemma 29 For every diagonal free simple constraint C of the form $h \leq n$ or $h \geq n$, for every $M \in \mathbb{N}$, $[\![norm_M(C)]\!] = \cup_{C' \in Rect(C)} [\![norm_M(C')]\!]$.

Proof
If C is of the form:

- $h \leq n$,
  - If $M \geq n$ then $norm_M(C) = C$ and for every $C' \in Rect(C)$, $norm_M(C') = C'$. Then we get the result.
  - If $M < n$ then $norm_M(C) = tt$. Let $C' h = n$. From Lemma 27 $C' \in Rect(C)$and $norm_M(C') = tt$ then $\cup_{C' \in Rect(C)} [\![norm_M(C')]\!] = tt$ and $[\![norm_M(C)]\!] = \cup_{C' \in Rect(C)} [\![norm_M(C')]\!]$.
- $h \geq n$,
  - The case when $M \geq n$ is obvious because every constraint in $Rect(C) \cup \{C\}$ is not modified by $norm_M$.
  - The case when $M < n$ is also obvious because $norm(C) = h > M$ and $norm_M(C') = h > M$ for every $C' \in Rect(C)$

□

Now we can easily extend results in the lemma above to diagonal free constraints.

Proposition 30 For every diagonal-free constraint g, for every $M \in \mathbb{N}$, $[\![norm_M(g)]\!] = \bigcup_{\hat{g} \in Rect(g)} [\![norm_M(\hat{g})]\!]$.

Proof
It is a consequence of Lemma 29 above and Proposition 28 □

Definition 31 Given a $g \in Gds$, and and integer $M \in \mathbb{N}$ we define the set

$$Rect_M(g) = \{norm_M(\hat{g}) \mid \hat{g} \in Rect(g)\}$$

.

From Proposition 26, we get that every diagonal-free constraint using constant smaller than an integer M can be decomposed into a finite set of M-rectangular constraints.

Proposition 32 For every constraint $g \in Gds$, for every $M \geq M_g$, $[\![g]\!] = \bigcup_{\hat{g} \in Rect_M(g)} [\![\hat{g}]\!]$.

Proof
From Proposition 30 $[\![norm_M(g)]\!] = \bigcup_{\hat{g} \in Rect(g)} [\![norm_M(\hat{g})]\!]$ or equivalently $[\![norm_M(g)]\!] = \bigcup_{\hat{g} \in Rect_M(g)} [\![\hat{g}]\!]$. From Proposition 26 for $M \geq M_g$, $[\![g]\!] = [\![norm_M(g)]\!]$ and we get the result. □

Remark: The same kind of property can be established for general constraints and triangular constraints. As rectangular constraints contain triangular constraints every M-bounded diagonal free atomic constraint can be decomposed into a finite union of M-bounded triangular constraints.

From the remark above we have the following corollary.

Corollary 33 Every constraint or diagonal free constraint can be decomposed into a finite equivalent set of triangular constraints.

3.3 Regions

We present a partitioning of the valuations into a finite number of equivalence classes called *regions*. Valuations in the same region must satisfy the same clock constraints, their time successors must also satisfy the same clock constraints, and they must satisfy the same clock constraints after a clock is reset.

The definition of a region we present here has been introduced by Alur and Dill [AD94] for analysing timed automata using only diagonal -free constraints. The equivalence relation between valuations is defined with respect to some integer M representing the maximal value used in constraints. The definition of that relation is somehow related to the definition of atomic constraints as atomic constraints can not be decomposed into smaller constraints. Thus, two equivalent valuations agree on the integral part of each clock whose values are smaller than M and they also agree on the order on the fractional part of the values of the clocks.

For a real number n let $\lfloor n \rfloor$ denote the integral part of n and $\{n\}$ denote the fractional part of n.

Let M be a natural number. Consider the parametrised binary relation $\sim^M \subseteq \mathcal{V}_{\mathcal{H}} \times \mathcal{V}_{\mathcal{H}}$ over valuations defined by, $v \sim^M v'$ if:

1. $v(h) > M$ if and only if $v'(h) > M$ for each $h \in \mathcal{H}$;
2. if $v(h) \leq M$, then $\lfloor v(h) \rfloor = \lfloor v'(h) \rfloor$ for every $h \in \mathcal{H}$;
3. if $v(h) \leq M$, then $\{v(h)\} = 0$ if and only if $\{v'(h)\} = 0$ for every $h \in \mathcal{H}$, and;
4. if $v(h) \leq M$ and $v(h') \leq M$, then $\{v(h)\} \leq \{v(h')\}$ if and only if $\{v'(h)\} \leq \{v'(h')\}$ for every $h, h' \in \mathcal{H}$.

Proposition 34 ([AD94]) The relation $\sim^M$ is an equivalence relation over the set of valuations with at most $2^{3|\mathcal{H}|-1} \times |\mathcal{H}|! \times (M+1)^{|\mathcal{H}|}$ equivalence classes.

Proof
The relation $\sim^M$ is defined as a conjunction of four properties. Each property defines an equivalence relation; let us denote them by $\sim_1^M, \ldots, \sim_4^M$, respectively. For each of these four relations we will give an upper bound on the number of its equivalence classes. The product of these bounds will give an upper bound on $\sim^M$ as the later is the intersection of the four equivalence relations.

The relation defined by the first condition has $2^{|\mathcal{H}|}$ equivalence classes, as the only thing that counts is whether the value of a clock is bigger than M or not. Similarly the third relation has $2^{|\mathcal{H}|}$ equivalence classes. The number of classes of the second relation is $(M+1)^{|\mathcal{H}|}$ as there are $M+1$ possible integer values of interest. Finally, the number of classes of the fourth relation is bounded by the number of permutations of the set of clocks multiplied by $2^{|\mathcal{H}|-1}$ as for every two clocks consecutive in a permutation we need to decide if they are equal or if the second is strictly bigger than the first.

Summarizing, we get $2^{3|\mathcal{H}|-1}|\mathcal{H}!|(M+1)^{|\mathcal{H}|}$.

□

We use $\mathcal{R}eg(M)$ (or $\mathcal{R}eg$ for short) to represent the set of equivalence classes of the relation $\sim^M$.

Definition 35 A *region* [AD94] is an equivalence class of the relation $\sim^M \subseteq \mathcal{V}_\mathcal{H} \times \mathcal{V}_\mathcal{H}$ defined above.

In Figure 3 we illustrate region for diagonal free constraints for the maximal constant $M = 2$. In Figure 3 valuations earlier presented in Table 1 are not equivalent. A region in the figure is either a corner point (for example $(0, 2)$), an open line segment (for example $0 < h_1 = h_2 < 1$) or an open box (for example $0 < h_1 < h_2 < 1$).

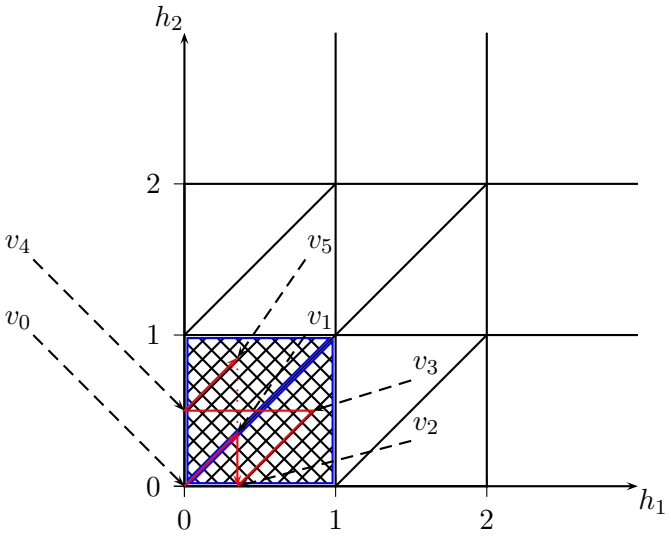

Figure 3: Region illustration.

From the definition of $\sim^M$, it comes that an equivalence class can be represented using a triangular constraint in g. According to the definition of $\sim^M$, two valuations that belong to the same equivalence class satisfy constraint of the form:

- $h = i_h$ or $i_h < h < i_h + 1$ for each $h \in H$ where $i_h \in \{0, 1, \ldots, M\}$ and we assume $M + 1 = \infty$. This is a consequence of $\sim_1^M$, $\sim_2^M$, $\sim_3^M$.
- $h - h' = i_{hh'}$ or $i_{hh'} < h - h' < i_{hh'} + 1$ for each couple $(h, h') \in H^2$ such that $h \bowtie M$ and $h' \bowtie M$ with $\bowtie \in \{=, <\}$. This is a consequence of $\sim_4^M$.

Given a valuation v, $[v]$ denotes the equivalence class (region) of v. We also use the letter r to represent a region. Given a region r, we define $r + t = \{[v + t] \mid v \in r\}$, $r\!\uparrow\! = \{r + t \mid t \in \mathbb{R}_{\geq 0}\}$, and $r[h := 0] = \{v[h := 0] : v \in r\}$. We write $r \subseteq g$ for $r \subseteq [\![g]\!]$.

Proposition 36 Let G be a set of M-bounded constraints then $\mathcal{R}eg(M)$ satisfies:

P1 $\forall g \in G, r \in \mathcal{R}eg$, either $r \subseteq [\![g]\!]$ or $[\![g]\!] \cap r = \emptyset$.

P2 $\forall r, r' \in \mathcal{R}eg$, if there exists some $v \in r$ and $t \in \mathbb{R}_{\geq 0}$ such that $v + t \in r'$, then for every $v' \in r$ there is some $t' \in \mathbb{R}_{\geq 0}$ such that $v' + t' \in r'$.

P3 $\forall r, r' \in \mathcal{R}eg, \forall h \in \mathcal{H}$, if $r[h := 0] \cap r' \neq \emptyset$, then $r[h := 0] \subseteq r'$.

Proof
We show P1 in the first item, P2 in the second item and P3 in the last item.

1. Let $g \in G$, from Proposition 32 let $[\![g]\!] = \bigcup_{g_i \in Rect_M(g)} [\![\hat{g}_i]\!]$. Each $\hat{g}_i$ is a rectangular constraint. $[\![g]\!] \cap r = \bigcup_{g_i \in Rect_M(g)} [\![\hat{g}_i]\!] \cap r)$. From Fact 23 there is at most one i such that r intersects $\hat{g}_i$. It follows that r intersects a constraint $\hat{g}_i$ of $Rect_M(g)$ if and only if $\hat{g}_i$ contains r. We have that if $v \vDash r$ then $v \vDash g$.
2. Let $v, v' \in r$, adding t to v may modify the integer part of the value (with respect to v) of some clocks or may modify the order on the fractional part of the value (with respect to v) of clocks. We aim at find a time t' such that:

   - The integer part of the value of each clock with respect to $v' + t'$ is equal to the integer part of the value of each clock with respect to $v + t$
   - The order of the fractional parts of clocks in $v' + t'$ is the same in $v + t$.
   - The set of clocks with zero fractional part in $v + t$ is the same in $v' + t'$.

   Let $|\mathcal{H}| = n$ and assume a permutation π of $\{1, \ldots, n\}$ such that

   $$\{v(h_{\pi_1})\} \bowtie_1 \{v(h_{\pi_2})\} \bowtie_2, \ldots, \bowtie_{n-1} \{v(h_{\pi_n})\} (*)$$

   with $\bowtie_i \in \{<, =\}$.

   Let $t \in \mathbb{R}_{\geq 0}$. It is clear that $\{v(h) + t\} = \{v(h) + \{t\}\}$. Only the fractional part of t may affect the order in $(*)$.

   There may be a largest index j such that
   $\{v(h_{\pi_j}) + \{t\}\} = \{v(h_{\pi_j})\} + \{t\}$. In case, no such j exists, take $j = n$.
   Clearly, $\{v(h_{\pi_j}) + \{t\}\} \geq \{v(h_{\pi_j})\}$ and; $\forall k > j$ we have:
   $\{v(h_{\pi_k}) + \{t\}\} < \{v(h_{\pi_k})\}$ and $\{v(h_{\pi_k}) + \{t\}\} < \{v(h_{\pi_j}) + \{t\}\}$.

   We get that:

   $$\{v(h_{\pi_{j+1}}) + \{t\}\} \bowtie_j \ldots\ldots \bowtie_{n-1} \{v(h_{\pi_n}) + \{t\}\} < \{v(h_{\pi_j}) + \{t\}\}$$

   Similarly, we establish that

   $$\{v(h_{\pi_j}) + \{t\}\} < \{v(h_{\pi_{j-1}}) + \{t\}\} \overline{\bowtie}_{j-2} \ldots \overline{\bowtie}_1 \{v(h_{\pi_1}) + \{t\}\}$$

   where $\overline{\bowtie}_k$ => if $\bowtie_j \in \{<\}$ otherwise $\bowtie_j \in \{=\}$, $\forall k \leq j$

- If $\{v'(h_{\pi_{j+1}}) + \{t'\}\} \neq 0$, in order to have

$$\{v'(h_{\pi_{j+1}}\{t'\}\} \bowtie_j \ldots \bowtie_{n-1} < \{v'(h_{\pi_n}\{t'\}\} < \{v'(h_{\pi_j}) + \{t'\}\} \text{ and}$$

$$\{v'(h_{\pi_{j+1}}) + \{t'\}\} \bowtie_j < \{v'(h_{\pi_{j-1}}) + \{t'\}\} \overline{\bowtie}_{j-2} \ldots \overline{\bowtie}_1 \{v'(h_{\pi_1}) + \{t'\}\}$$

  We take $\{t'\} \in [0, 1 - \{v'(h_{\pi_j})\}[\cap[1 - \{v'(h_{\pi_{j+1}})\}, 1[$.
- If $\{v'(h_{\pi_{j+1}}) + \{t'\}\} = 0$ then $\{t\} = 1 - \{v(h_{\pi_{j+1}})\}$; and we take $\{t'\} = 1 - \{v'(h_{\pi_{j+1}})\}$.

It comes that $\lfloor\{v'(h_{\pi_i})\} + \{t'\}\rfloor = \lfloor\{v(h_{\pi_i})\} + \{t\}\rfloor$.

To ensure that $\lfloor\{v'(h_{\pi_i})\} + t'\rfloor = \lfloor\{v(h_{\pi_i})\} + t\rfloor$ we must take $\lfloor t\rfloor = \lfloor t'\rfloor$.

3. Let $v_1, v_2 \in r$, then v_1 and v_2 satisfy all the conditions in the definition of an equivalence class. Its obvious that $v_1[h := 0]$ and $v_2[h := 0]$ also satisfy those three conditions and then $v_1[h := 0]$ and $v_2[h := 0]$ belong to $r[h := 0]$.

   If $v \in r[h := 0] \cap r'$ then every $v' \in r'$ is equivalent to v which is also equivalent to every $v'' \in r[h := 0]$. Thus $v \in r[h := 0]$ if and only if $v \in r[h := 0]$.

□

3.4 Timed Processes

3.4.1 Definitions

Let $\Sigma = \{a_1, a_2 \ldots\}$ be a set of *events* . We consider $\mathcal{H}_\Sigma = \{h_1, h_2, \ldots\}$ the set of clocks. The clock h_i is the unique clock associated to the event a_i. When there is no confusion, a will denote an event and h_a will denote the unique clock associated to a. There are as many clocks as events. The symbol Gds_Σ will denote the set of constraints defined over $\mathcal{H}_\Sigma$, the symbol $Agds_\Sigma$ will denote the set of rectangular constraints over $\mathcal{H}_\Sigma$, and the symbol $\mathcal{V}_\Sigma$ will denote the set of valuations over $\mathcal{H}_\Sigma$.

Definition 37 A *timed process* , or process for short, is a tuple

$$\mathcal{P} = \langle P, \Sigma \times Gds_\Sigma, p^0, \Delta_P \rangle$$

where,

- P is a finite set of states,
- $p^0 \in P$ is the initial state,
- $\Delta_P \subseteq P \times Gds_\Sigma \times \Sigma \times P$ is a transition relation.

Sometimes, we shortly write $p \xrightarrow{g,a} p'$ for a transition (p, g, a, p') in Δ_P. The *bound* of a timed process is the maximal constant that occurs in its guards. For a timed process $\mathcal{P}$, $M_\mathcal{P}$ denotes its bound. Given a constant M, we say a timed process is M-bounded if its bound is smaller that M.

Definition 38 A timed process is *deterministic* if whenever there are two transitions $p \xrightarrow{g_1,a} p_1$ and $p \xrightarrow{g_2,a} p_2$ with $p_1 \neq p_2$, the constraint $g_1 \wedge g_2$ is inconsistent.

In the figures 4, 5, 6, we illustrate, three timed processes. The timed process in Figure 5 and Figure 6 are deterministic and timed process in Figure 4 is not deterministic.

The timed process in Figure 4 is not deterministic as the conjunction of the guards in the two transitions outgoing from p_0 is consistent while their events are the same. In Figure 5, the conjunction of the guards is inconsistent and, in Figure 6 the transitions outgoing from p_0 are not labelled with the same event.

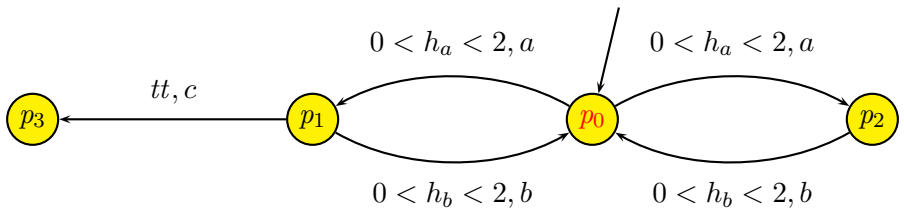

Figure 4: A non deterministic timed process:$\mathcal{P}_0$.

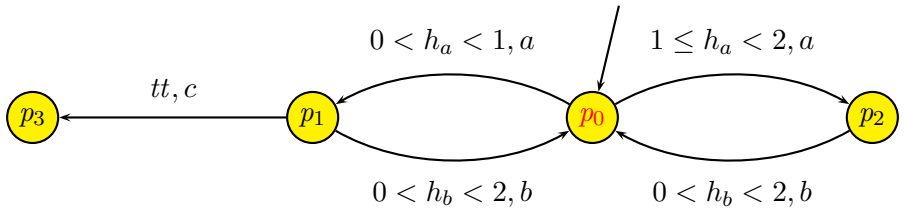

Figure 5: A deterministic timed process:$\mathcal{P}_1$.

3.4.2 Semantics

The semantics of a timed process is a transition system that represents all possible behaviours of the timed process. The idea is that each clock h_a records the amount of time elapsed since the last occurrence of the corresponding event a. The time elapses continuously at a state. Whenever an action a is executed, the clock h_a is automatically reset. No other clock assignments are permitted.

Definition 39 The semantics of a timed process $\mathcal{P}$ as above is the transition system

$$[\![\mathcal{P}]\!] = \langle P \times \mathcal{V}_\Sigma, \Sigma \cup \mathcal{V}_\Sigma, (s^0, v^0), \rightarrow \rangle$$

where $\rightarrow \subseteq (P \times \mathcal{V}_\Sigma) \times (\Sigma \cup \mathcal{V}_\Sigma) \times (P \times \mathcal{V}_\Sigma)$ is defined by:

- $(p, v) \xrightarrow{v+t} (p, v + t)$ for every $t \geq 0$.
- $(p, v) \xrightarrow{a} (p', v[h_a := 0])$ if there is $(p, g, a, p') \in \Delta_P$ such that $v \in [\![g]\!]$.

Delay transitions are transitions labelled with valuations and *discrete transitions* are transitions labelled with events.

Remark: When presenting the semantics of timed automata [AD94, DM02, BCL05] and event-recording automata [AD94], it is usual to label delay transitions with non negative real numbers. In the semantics presented above, delay transition are labelled with valuations. We remark that these two presentations are equivalent. The choice of the presentation above will be justified in the next chapters when the semantics of formulas will be defined.

Notation: Later we use the notation $s \xrightarrow{v,a} s'$ if there exists s'' such that $s \xrightarrow{v} s''$ and $s'' \xrightarrow{a} s'$.

Let us use the following example to illustrate the notion of semantics of timed processes. We consider process in Figure 4 and Figure 5 and transitions from p_0 to p_1 and p_2. In Figure 7, we present the beginning of the semantics of the process in Figure 4. As that process is not deterministic, at the same time (for example $t = 0.4$), it is possible to

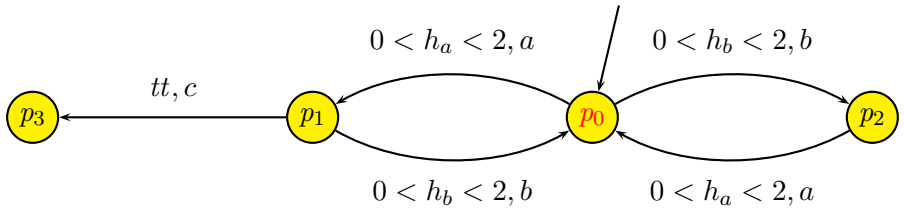

Figure 6: A deterministic timed process:$\mathcal{P}_2$.

trigger the event a and either move to p_1 or p_2. From p_1 it is possible to do immediately c while it is not the case from p_2.

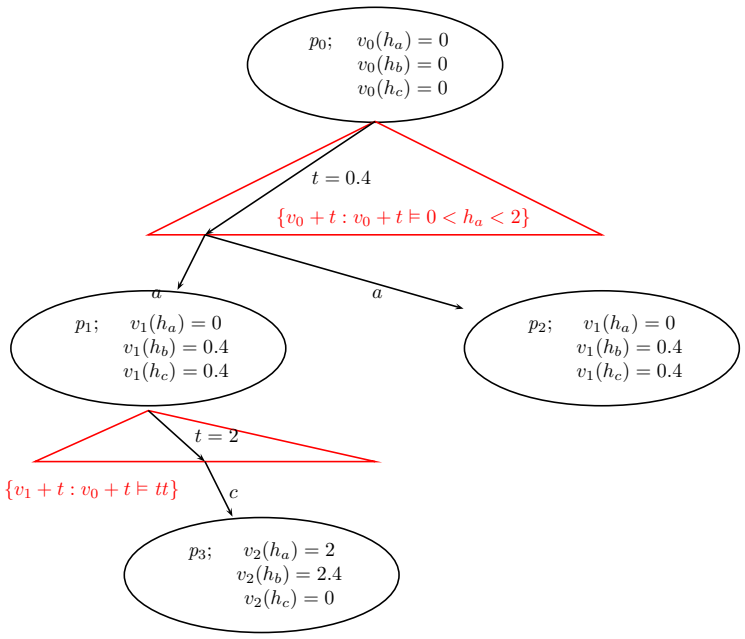

Figure 7: A part of the semantics of $\mathcal{P}_0$.

3.4.3 Representations for Timed Processes

The above semantics is not very convenient as both the set of states and the set of labels occurring in transitions are uncountable. We will consider two more abstract semantics of processes. The first will abstract from valuations in the labels of transitions. The second will replace valuations in states by regions. In order for the abstractions to be finite, they will be parametrized by a bound M on the clock values.

Definition 40 The *M-action abstraction* of a timed process $\mathcal{P}$ is the $(\Sigma \cup Agds_\Sigma(M))$-labeled transition system

$$\langle\!|\mathcal{P}|\!\rangle^M = \langle P \times \mathcal{V}_\Sigma, \Sigma \cup Agds_\Sigma(M), (s^0, v^0), \Delta_v \rangle,$$

where $\Delta_v \subseteq (P \times \mathcal{V}_\Sigma) \times (\Sigma \cup Agds_\Sigma(M)) \times (P \times \mathcal{V}_\Sigma)$ is defined by:

- $(p, v) \xrightarrow{\hat{g}} (p, v + t)$ for any $t \in \mathbb{R}^+$ such that $v + t \vDash \hat{g}$ and
- $(p, v) \xrightarrow{a} (p', v[h_a := 0])$ if there is $(p, g, a, p') \in \Delta_P$ with $v \vDash g$.

We observe that the M-action representation is obtained from the semantics by replacing valuations on delay-transitions with M-rectangular constraints they satisfy. Then for every timed process $\mathcal{P}$ and every natural constant M, there is an isomorphism between $[\![\mathcal{P}]\!]$ and $(\!|\mathcal{P}|\!)^M$.

Definition 41 The *M-region abstraction* of a timed process $\mathcal{P}$ is the $(\Sigma \cup Agds_\Sigma(M))$-labeled transition system

$$(\!|\mathcal{P}|\!)^M_{reg} = \langle P \times \mathcal{R}eg(M), \Sigma \cup Agds(M), (p^0, r^0), \Delta_r \rangle,$$

where $v^0 \in r^0$, $\Delta_r \subseteq (P \times \mathcal{R}eg(M)) \times (\Sigma \cup Agds_\Sigma(M)) \times (P \times \mathcal{R}eg(M))$ is defined by:

- $(p, r) \xrightarrow{\hat{g}} (p, r')$ with $r' \subseteq r\uparrow$ and $r' \subseteq \hat{g}$.
- $(p, r) \xrightarrow{a} (p', r[h_a := 0])$ if there is $(p, g, a, p') \in \Delta_P$ with $r \subseteq g$.

Proposition 42 For every timed process $\mathcal{P}$, and every $M \geq M_\mathcal{P}$: $(\!|\mathcal{P}|\!)^M$ is bisimilar to $(\!|\mathcal{P}|\!)^M_{reg}$.

Proof
We consider a relation $\sim \subseteq (P \times \mathcal{V}_\Sigma) \times (P \times \mathcal{R}eg_\Sigma(M))$ defined by $(p, v) \sim (p, [v])$ for every $p \in P, v \in \mathcal{V}_\Sigma$. We show that it is a bisimulation.

- First, we consider delay transitions. Assume that $(p, v) \sim (p, [v])$. If $(p, v) \xrightarrow{\hat{g}} (p, v')$, then there is $t \in \mathbb{R}^+$ such that $v + t \in [\![\hat{g}]\!]$. According to Proposition 36, $[v + t] \subseteq \hat{g}$ and obviously $[v + t] \subseteq [v]\uparrow$. Then, we get that $(p, [v]) \xrightarrow{\hat{g}} (p, [v + t])$ and $(p, v + t) \sim (p, [v + t])$. Reciprocally, if $(p, r) \xrightarrow{\hat{g}} (p, r')$, then $r' \subseteq \hat{g}$ and $r' \subseteq r\uparrow$. Let $v \in r$ according to Proposition 36, there is $t \in \mathbb{R}^+$ such that $v + t \in r'$. Since $r' \subseteq \hat{g}$, we get $v + t \in [\![\hat{g}]\!]$ and then $(p, v) \xrightarrow{\hat{g}} (p, v')$.
- Next, we consider discrete transitions. Assume that $(p, v) \sim (p, [v])$. If $(p, v) \xrightarrow{a} (p', v')$, then $v' = v[h_a := 0]$ and there is $p \xrightarrow{g,a} p'$ such that $v \in [\![g]\!]$. Let $\hat{g} \in Agds(M)$ be an atomic guard such that $v \in [\![\hat{g}]\!]$. Then we get $(p, [v]) \xrightarrow{a} (p', [v'])$ and $(p, v') \sim (p, [v'])$. Reciprocally, if $(p, r) \xrightarrow{a} (p', r')$, then $r' = r[h_a := 0]$ and there is $p \xrightarrow{g,a} p'$ such that $r \in [\![g]\!]$. Let $v \in r$, obviously $v \in [\![g]\!]$, and $v[h_a := 0] \in r'$. It follows that $(p, v) \xrightarrow{a} (p', v[h_a := 0])$ and $(p, v[h_a := 0]) \sim (p, r')$.

□

Notation: Later we use the notation $s \xrightarrow{g,a} s'$ if there exists s'' such that $s \xrightarrow{g} s''$ and $s'' \xrightarrow{a} s'$.

4 The Logic WT_μ

We define the syntax of WT_μ formulas. WT_μ formulas have modalities indexed with constraints and modalities indexed with events. We define rectangular formulas that use only rectangular constraints and we show that every formula can be transformed into an M-equivalent rectangular formula.

4.1 Definitions

The logic WT_μ is an adaptation of the μ-calculus and ERL. Apart from the usual events modalities, it has also modalities indexed by constraints. The formulas of WT_μ describe properties on timed processes.

Definition 43 Let X, Y range over the set of variables denoted Var. A formula φ of WT_μ is generated using the following grammar:

$$\varphi ::= tt \mid ff \mid X \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle a \rangle \varphi \mid \langle g \rangle \varphi \mid [a]\varphi \mid [g]\varphi \mid \mu X.\varphi \mid \nu X.\varphi$$

where $a \in \Sigma$ is an event and $g \in Gds_\Sigma$ is a constraint.

The *bound* of a formula is the maximal constant that occurs in its constraints. For a formula φ, M_φ denotes its bound. Given a constant M, we say that a formula is M-bounded if its bound is smaller that M.

Notion of bound variables, sentences, subformulas, well named formula, ν-variables, μ-variable, dependency order, alternation depth, guarded formulas, expansion, and definition list are obvious from the definitions of similar notions for the setting of the μ-calculus in Section 2.

4.2 Semantics of WT_μ

A formula is interpreted over timed processes, or rather its semantics. Intuitively, we say that a state (p, v) satisfies a formula $[g]\varphi$, if whenever starting from v we let the time pass and reach a valuation $v' \vDash g$ then $(p, v') \vDash_t \varphi$. Similarly, a formula $\langle g \rangle \varphi$ is satisfied if by letting the time pass it is possible to go from valuation v to a valuation $v' \vDash g$ with $(p, v') \vDash_t \varphi$. The meaning for the modalities $[a]$ and $\langle a \rangle$ is classical.

We will be mainly interested in describing timed processes, but actually the formulas of WT_μ can be evaluated in any $(\mathcal{V}_\Sigma \cup \Sigma)$- labelled transition system. Let us fix such a system $\mathcal{S} = \langle S, \Sigma \cup \mathcal{V}_\Sigma, s^0, \Delta_\mathcal{S} \rangle$. The semantics of a formula φ, denoted $[\![\varphi]\!]^\mathcal{S}_{Val}$, defined with respect to an assignment $Val : Var \to 2^S$ is the set of states of $\mathcal{S}$ which satisfy φ.

We write $\mathcal{S}, s, Val \vDash_t \varphi$ to say that the state s satisfies φ with respect to the valuation Val.

Definition 44 For a given $(\mathcal{V}_\Sigma \cup \Sigma)$-labelled transition system $\mathcal{S}$, a given formula φ and an assignment $Val : Var \to \mathcal{P}(S)$, we define the satisfaction relation $\vDash_t$ inductively as follows:

- $\mathcal{S}, s, Val \vDash_t tt$.
- $\mathcal{S}, s, Val \vDash_t X$ if $s \in Val(X)$.
- $\mathcal{S}, s, Val \vDash_t \varphi_1 \vee \varphi_2$ if $\mathcal{S}, s, Val \vDash_t \varphi_1$ or $\mathcal{S}, s, Val \vDash_t \varphi_2$.
- $\mathcal{S}, s, Val \vDash_t \varphi_1 \wedge \varphi_2$ if $\mathcal{S}, s, Val \vDash_t \varphi_1$ and $\mathcal{S}, s, Val \vDash_t \varphi_2$.
- $\mathcal{S}, s, Val \vDash_t \langle a \rangle \varphi$ if there is $s \xrightarrow{a} s'$ such that $\mathcal{S}, s', Val \vDash_t \varphi$.
- $\mathcal{S}, s, Val \vDash_t \langle g \rangle \psi$ if there is $s \xrightarrow{v} s'$ such that $v \in [\![g]\!]$ and $\mathcal{S}, s', Val \vDash_t \psi$.
- $\mathcal{S}, s, Val \vDash_t [a]\varphi$ if for all $s \xrightarrow{a} s'$ we have $\mathcal{S}, s', Val \vDash_t \varphi$.
- $\mathcal{S}, s, Val \vDash_t [g]\psi$ if for all $s \xrightarrow{v} s'$ with $v \in [\![g]\!]$, we have $\mathcal{S}, s', Val \vDash_t \psi$.
- $\mathcal{S}, s, Val \vDash_t \mu X.\varphi(X)$ if $s \in \cap\{T \subseteq S \mid [\![\varphi(X)]\!]^\mathcal{S}_{Val[X/T]} \subseteq T\}$.
- $\mathcal{S}, s, Val \vDash_t \nu X.\varphi(X)$ if $s, \in \cup\{T \subseteq S \mid T \subseteq [\![\varphi(X)]\!]^\mathcal{S}_{Val[X/T]}\}$.

The meaning of a formula is formally defined as follows:

$$[\![\varphi]\!]^{\mathcal{S}}_{Val} = \{s \mid \mathcal{S}, s, Val \vDash_t \varphi\}.$$

We will write $\mathcal{S} \vDash_t \varphi$ for $\mathcal{S}, s^0 \vDash_t \varphi$ to say that $\mathcal{S}$ is a *model* of the sentence φ.

To ensure the existence of fixpoints, we need to show that modal operators indexed with constraints and modal operators indexed with events are monotone.

Proposition 45 The operators $\langle\alpha\rangle$ and $[\alpha]$ are monotone for every $\alpha \in Gds_\Sigma \cup \Sigma$.

Proof
The cases for operators other than $\langle g\rangle$ and $[g]$ are standard. We show that modal operators indexed with constraints are monotonic. Assume that there is φ_1 and φ_2 and a transition system $\mathcal{S}$ such that $[\![\varphi_1]\!]^{\mathcal{S}}_{Val} \subseteq [\![\varphi_2]\!]^{\mathcal{S}}_{Val}$

- If $s \in [\![\langle g\rangle\varphi_1]\!]^{\mathcal{S}}_{Val}$ then there is $s \xrightarrow{v} s'$ with $v \in [\![g]\!]$ such that $s' \in [\![\varphi_1]\!]^{\mathcal{S}}_{Val}$ and then $s \in [\![\varphi_2]\!]^{\mathcal{S}}_{Val}$ as $[\![\varphi_1]\!]^{\mathcal{S}}_{Val} \subseteq [\![\varphi_2]\!]^{\mathcal{S}}_{Val}$. Then, there is there is $s \xrightarrow{v} s'$ with $v \in [\![g]\!]$ such that $s' \in [\![\varphi_2]\!]^{\mathcal{S}}_{Val}$ meaning that $s \in [\![\langle g\rangle\varphi_2]\!]^{\mathcal{S}}_{Val}$
- If $s \in [\![[g]\varphi_1]\!]^{\mathcal{S}}_{Val}$ and $s \notin [\![[g]\varphi_2]\!]^{\mathcal{S}}_{Val}$ then there is $s \xrightarrow{v} s'$ with $v \in [\![g]\!]$ such that $s' \notin [\![\varphi_2]\!]^{\mathcal{S}}_{Val}$. As $s \in [\![\varphi_2]\!]^{\mathcal{S}}_{Val}$ as $[\![\varphi_1]\!]^{\mathcal{S}}_{Val} \subseteq [\![\varphi_2]\!]^{\mathcal{S}}_{Val}$ we get that $s' \notin [\![\varphi_1]\!]^{\mathcal{S}}_{Val}$. Then there is $s \xrightarrow{v} s'$ with $v \in [\![g]\!]$ such that $s' \notin [\![\varphi_1]\!]^{\mathcal{S}}_{Val}$ and we get a contradiction.

□

We introduce the negation operator $\neg$. Given a sentence φ, a $(\mathcal{V}\times\Sigma)$-labelled transition system $\mathcal{S}$, and a Valuation Val, we define $[\![\neg\varphi]\!]^{\mathcal{S}}_{Val} = S \setminus [\![\varphi]\!]^{\mathcal{S}}_{Val}$

Proposition 46 We have the following equivalences.

1. $\neg tt \equiv ff$
2. $\neg ff \equiv tt$
3. $\neg(\varphi_1 \wedge \varphi_2) \equiv \neg\varphi_1 \vee \neg\varphi_2$
4. $\neg(\varphi_1 \vee \varphi_2) \equiv \neg\varphi_1 \wedge \neg\varphi_2$
5. $\neg\langle\alpha\rangle\varphi \equiv [\alpha]\neg\varphi$ with $\alpha \in \Sigma \cup Gds$
6. $\neg[\alpha]\varphi \equiv \langle\alpha\rangle\neg\varphi$
7. $\neg\mu X.\varphi(X) \equiv \nu X.\neg\varphi(\neg X)$
8. $\neg\nu X.\varphi(X) \equiv \mu X.\neg\varphi(\neg X)$

Proof
Let $\mathcal{S}$ be a $(\Sigma \cup \mathcal{V}_\Sigma)$-labelled transition system and let s be a state of $\mathcal{S}$. As the cases for operators other than $\langle g\rangle$ and $[g]$ are standard, we consider the following cases:

- If $s \in [\![\neg\langle g\rangle\varphi]\!]$ then $s \notin [\![\langle g\rangle\varphi]\!]$. It is equivalent to say that for every $v \in [\![g]\!]$, for every $s \xrightarrow{v} s'$, we have that $s' \notin [\![\varphi]\!]$ meaning by definition that $s' \notin [\![[g]\neg\varphi]\!]$.
- The case of $\neg[g]\varphi \equiv \neg\langle g\rangle\neg\varphi$ is obvious from the previous case.

□

We write $\varphi_1 \equiv \varphi_2$ when the formulas φ_1 and φ_2 are equivalent.

Proposition 47 Let $g, g_1, g_2, \ldots, g_n$ such that $[\![g]\!] = \bigcup_{i=1..n} [\![g_i]\!]$ then,

1. $\langle g\rangle\varphi \equiv \bigvee_{i=1..n}\langle g_i\rangle\varphi$
2. $[g]\varphi \equiv \bigwedge_{i=1..n}[g_i]\varphi$

Proof
We will consider the first case since the proof of the second case is easy by using Proposition 46. Let $\mathcal{S}$ be a $(\Sigma \cup \mathcal{V}_\Sigma)$-labelled transition system and s be a state of $\mathcal{S}$

($\Longrightarrow$) If $\mathcal{S}, s \vDash_t \langle g \rangle \varphi$ then there is $s \xrightarrow{v} s'$ with $v \in [\![g]\!]$ such that $\mathcal{S}, s' \vDash_t \varphi$. As $[\![g]\!] = \bigcup_{i=1..n} [\![g_i]\!]$, there is $i \in [1..n]$ such that $v \in [\![g_i]\!]$. Then, $s \xrightarrow{v} s'$ with $v \in [\![g_i]\!]$ and $\mathcal{S}, s' \vDash_t \varphi$, meaning that $\mathcal{S}, s \vDash_t \langle g_i \rangle \varphi$ or equivalently $\mathcal{S}, s \vDash_t \bigvee_{i=1..n} \langle g_i \rangle \varphi$.

($\Longleftarrow$) If $\mathcal{S}, s \vDash_t \bigvee_{i=1..n} \langle g_i \rangle \varphi$ then $\mathcal{S}, s \vDash_t \langle g_i \rangle \varphi$ for some $i \in [1..n]$ meaning that, there is $s \xrightarrow{v} s'$ with $v \in [\![g_i]\!]$ such that $\mathcal{S}, s' \vDash_t \varphi$. But $v \in [\![g_i]\!]$ implies $v \in [\![g]\!]$ as $[\![g]\!] = \bigcup_{i=1..n} [\![g_i]\!]$. Then we get that $\mathcal{S}, s \vDash_t \langle g \rangle \varphi$.

□

Meaning of a formula over a timed process Consider φ a formula, $\mathcal{P}$ a timed process. We say that φ is satisfied in a state p, a valuation v and a valuation $Val : Var \to \mathcal{P}(P \times \mathcal{V}_\Sigma)$ of propositional variables and we write $\mathcal{P}, (p, v), Val \vDash \varphi$ when $[\![\mathcal{P}]\!], (p, v), Val \vDash_t \varphi$.

The meaning $[\![\varphi]\!]^{\mathcal{P}}_{Val} \subseteq P \times \mathcal{V}_\Sigma$ of a formula over a timed process $\mathcal{P}$ is defined by

$$[\![\varphi]\!]^{\mathcal{P}}_{Val} = [\![\varphi]\!]^{[\![\mathcal{P}]\!]}_{Val}$$

We will write $\mathcal{P} \vDash \varphi$ if $[\![\mathcal{P}]\!]$ is a model of φ and we say that $\mathcal{P}$ is a *model* of φ.

4.3 Rectangular Formulas

We introduce rectangular form for WT_μ formulas and we show the equivalence between a formula and its rectangular forms.

Definition 48 A *rectangular formula* is a formula defined using rectangular constraints.

Recall that $Rect_M(g)$ was presented in Definition 31. The M-rectangular formula associated to the formula φ is the formula $Rect_M(\varphi)$ inductively defined by:

- $Rect_M(ff) = ff$
- $Rect_M(tt) = tt$
- $Rect_M(X) = X$
- $Rect_M(\varphi_1 \wedge \varphi_2) = Rect_M(\varphi_1) \wedge Rect_M(\varphi_2)$
- $Rect_M(\varphi_1 \vee \varphi_2) = Rect_M(\varphi_1) \vee Rect_M(\varphi_2)$
- $Rect_M(\langle g \rangle \varphi) = \bigvee_{\hat{g} \in Rect_M(g)} \langle \hat{g} \rangle \varphi$
- $Rect_M([g]\varphi) = \bigwedge_{\hat{g} \in Rect_M(g)} [\hat{g}] \varphi$
- $Rect_M(\langle a \rangle \varphi) = \langle a \rangle Rect_M(\varphi)$
- $Rect_M([a]\varphi) = [a] Rect_M(\varphi)$
- $Rect_M(\sigma X.\varphi(X)) = \sigma X.Rect_M(\varphi(X))$ where σ is one of $\{\mu, \nu\}$

We can state the following proposition.

Proposition 49 For every $M \geq M_\varphi$, $\mathcal{S}, s, Val \vDash_t \varphi$ if and only if $\mathcal{S}, s, Val \vDash_t Rect_M(\varphi)$

Proof
The proof uses structural induction.

- The cases of ff, tt, X are standard.

- The cases of formulas of the form $\varphi_1 \wedge \varphi_2$ or $\varphi_1 \vee \varphi_2$ are also standard.
- If $\mathcal{S}, s, Val \models_t \langle a \rangle \varphi$, then there is $s \xrightarrow{a} s' \in \rightarrow$ with $v' = v[h_a := 0]$ such that $\mathcal{S}, s' \models_t \varphi$. By induction hypothesis, $\mathcal{S}, s' \models_t Rect_M(\varphi)$. It follows that $\mathcal{S}, s, Val \models_t Rect_M(\langle a \rangle \varphi)$. The other way of the proof use similar argumentation.
- The case of $[a]\varphi$ uses dual argumentation.
- The case when $\varphi = \langle g \rangle \varphi$. $Rect_M(\varphi) = \bigvee_{\hat{g} \in Rect_M(g)} \langle \hat{g} \rangle \varphi$. From Proposition 32, $[\![g]\!] = \bigcup_{\hat{g} \in Rect_M(g)} [\![\hat{g}]\!]$. We use Proposition 47 to conclude.
- Argumentation for the case when $\varphi = [g]\varphi$ is similar to the case when $\varphi = \langle g \rangle \varphi$.
- The cases of fixpoint formulas are standard.

□

5 Model-Checking of WT_μ

We consider the model-checking of WT_μ. We define the abstract semantics of formulas in which formulas are interpreted over $(Gds_\Sigma \cup \Sigma)$-labelled transition systems. In that semantics constraints in transitions are directly compared (identity test) with the constraints in formulas. Then we use that semantics for the model-checking by showing that checking if a timed process is a model of a formula is the same as checking if the M-region semantics of that timed process is an abstract model (with respect to the abstract semantics) of the M-rectangular formula of the formula for M sufficiently big.

5.1 Abstract Semantics for Formulas

We would also like to evaluate our formulas in models of the form $\langle\!\langle \mathcal{P} \rangle\!\rangle$ or $\langle\!\langle \mathcal{P} \rangle\!\rangle^M_{reg}$. More generally, we can define a semantics of WT_μ in any $(Gds_\Sigma \cup \Sigma)$-labelled transition system $\mathcal{S} = \langle S, Gds_\Sigma \cup \Sigma, s^0, \rightarrow \rangle$ as follows:

Definition 50 The *symbolic relation of satisfaction* is defined between a symbolic representation $\mathcal{S}$, a valuation of variables Val and a formula φ as follows:

- $\mathcal{S}, s, Val \models_g tt$
- $\mathcal{S}, s, Val \models_g X$ when $s \in Val(X)$
- $\mathcal{S}, s, Val \models_g \varphi_1 \vee \varphi_2$ when $\mathcal{S}, s, Val \models_g \varphi_1$ or $\mathcal{S}, s, Val \models_g \varphi_2$.
- $\mathcal{S}, s, Val \models_g \varphi_1 \wedge \varphi_2$ when $\mathcal{S}, s, Val \models_g \varphi_1$ and $\mathcal{S}, s, Val \models_g \varphi_2$.
- $\mathcal{S}, s, Val \models_g \langle a \rangle \psi$ if there is $s \xrightarrow{a} s'$ such that $\mathcal{S}, s', Val \models_g \psi$
- $\mathcal{S}, s, Val \models_g \langle g \rangle \psi$ if there is $s \xrightarrow{g} s'$ such that $\mathcal{S}, s', Val \models_g \psi$
- $\mathcal{S}, s, Val \models_g [a] \psi$ if for all $s \xrightarrow{a} s'$ we have $\mathcal{S}, s', Val \models_g \psi$
- $\mathcal{S}, s, Val \models_g [g] \psi$ if for all $s \xrightarrow{g} s'$ we have $\mathcal{S}, s', Val \models_g \psi$
- $\mathcal{S}, s, Val \models_g \mu X.\varphi(X)$ if $s \in \cap \{T \subseteq S \mid [\![\varphi(X)]\!]^{\mathcal{S}}_{Val[X/T]} \subseteq T\}$
- $\mathcal{S}, s, Val \models_g \nu X.\varphi(X)$ if $s \in \cup \{T \subseteq S \mid T \subseteq [\![\varphi(X)]\!]^{\mathcal{S}}_{Val[X/T]}\}$

The abstract meaning of a formula is formally defined as follows:

$$\langle\!\langle \varphi \rangle\!\rangle^{\mathcal{S}}_{Val} = \{s \mid \mathcal{S}, s, Val \models_g \varphi\}.$$

We will write $\mathcal{S} \models_g \varphi$ for $\mathcal{S}, s^0 \models_g \varphi$ to say that $\mathcal{S}$ is an *abstract model* of the sentence φ.

Observe that this is nothing but the standard semantics of the mu-calculus. We use this observation in the next subsection for the model-checking decision procedure. Results we present in that subsection use the framework of Subsection **??**

5.2 Model-Checking Results

Let us now consider the model-cheching of WT_μ. From Proposition 49, we can consider rectangular formula as "good" abstraction of formula and for sufficienty big M, we will use the M-region representation of timed process $\mathcal{P}$, to check whether it is a model of a given formulas.

Proposition 51 For every process $\mathcal{P}$, for every M_φ-rectangular formula φ, for every $M \geq M_\varphi$: $[\![\mathcal{P}]\!], (p,v), Val \models_t \varphi$ if and only if $(\!|\mathcal{P}|\!)^M, (p,v), Val \models_g \varphi$.

Proof
The proof is by induction on the structure of the formula. The cases of *ff*, *tt*, $\varphi \vee \varphi$, $\varphi \wedge \varphi$ and $\sigma X.\varphi(X)$ are immediate. We consider the cases of $\langle g \rangle \varphi$, $[g]\varphi$, $\langle a \rangle \varphi$ and $[a]\varphi$.

- Assume that the formula has the form $\langle g \rangle \varphi$ where, $g \in Agds(M)$.

  $\Rightarrow$ if $[\![\mathcal{P}]\!], (p,v), Val \models_t \langle g \rangle \varphi$, then there is $(p,v) \xrightarrow{v'} (p,v')$ such that $v' \in [\![g]\!]$ and $[\![\mathcal{P}]\!], (p,v'), Val \models_t \varphi$. By the induction hypothesis, $(\!|\mathcal{P}|\!)^M, (p,v'), Val \models_g \varphi$. But, $(p,v) \xrightarrow{v'} (p,v')$, $v' \in [\![g]\!]$ and $g \in Agds(M)$ involve that $(p,v) \xrightarrow{g} (p,v')$ is a transition in $(\!|\mathcal{P}|\!)^M$. It follows that $(\!|\mathcal{P}|\!)^M, (p,v), Val \models_g \langle g \rangle \varphi$.

  $\Leftarrow$ $(\!|\mathcal{P}|\!)^M, (p,v), Val \models_g \langle g \rangle \varphi$, then there is $(p,v) \xrightarrow{g} (p,v')$ such that $[\![\mathcal{P}]\!], (p,v'), Val \models_g \varphi$. By the induction hypothesis, $[\![\mathcal{P}]\!], (p,v'), Val \models_t \varphi$. But if $(p,v) \xrightarrow{g} (p,v')$ is a transition in $(\!|\mathcal{P}|\!)^M$ then $v' \in [\![g]\!]$ and there is $t \in \mathbb{R}^+$ such that $v' = v + t$. It follows that, the transition $(p,v) \xrightarrow{v'} (p,v')$ belongs to $[\![\mathcal{P}]\!]$ and then $[\![\mathcal{P}]\!], (p,v), Val \models_t \langle g \rangle \varphi$.

- In the case of $[g]\varphi$, we use a dual argumentation.

- Assume that the formula has the form $\langle a \rangle \varphi$,

  $\Rightarrow$ if $[\![\mathcal{P}]\!], (p,v), Val \models_t \langle a \rangle \varphi$, then there is $(p,v) \xrightarrow{a} (p',v')$ such that $[\![\mathcal{P}]\!], (p,v'), Val \models_t \varphi$ with $v' = v[h_a := 0]$. By the induction hypothesis, $(\!|\mathcal{P}|\!)^M, (p',v'), Val \models_g \varphi$. But if $(p,v) \xrightarrow{a} (p',v')$ is a transition of $[\![\mathcal{P}]\!]$ then, there is a transition $p \xrightarrow{g,a} p'$ in $\mathcal{P}$ for which $v \in [\![g]\!]$. According to the definition of $(\!|\mathcal{P}|\!)^M$, there is also the transition $(p,v) \xrightarrow{a} (p',v')$ in $(\!|\mathcal{P}|\!)^M$. It follows that $(\!|\mathcal{P}|\!)^M, (p,v), Val \models_g \langle a \rangle \varphi$.

  $\Leftarrow$ if $(\!|\mathcal{P}|\!)^M, (p,v), Val \models_g \langle a \rangle \varphi$ then there is $(p,v) \xrightarrow{a} (p',v')$ such that $[\![\mathcal{P}]\!], (p,v'), Val \models_g \varphi$ with $v' = v[h_a := 0]$. By the induction hypothesis, $[\![\mathcal{P}]\!], (p',v'), Val \models_t \varphi$. Because $(p,v) \xrightarrow{a} (p',v')$ belong to $[\![\mathcal{P}]\!]$, we get that $[\![\mathcal{P}]\!], (p,v), Val \models_t \langle g \rangle \varphi$.

- A dual argumentation holds in the case of $[a]\varphi$.

□

Using bisimilarity between $(\!|\mathcal{P}|\!)^M_{reg}$ and $(\!|\mathcal{P}|\!)^M$, for sufficiently big M, and that every formula are equivalent to some rectangular formula (see Proposition 49) we get the following lemma.

Lemma 52 For every process $\mathcal{P}$, for every formula φ, for every $M \geq \max(M_\varphi, M_\mathcal{P})$: $[\![\mathcal{P}]\!], (p,v), Val \models_t \varphi$ if and only if $(\!|\mathcal{P}|\!)^M_{reg}, (p,[v]_M), Val \models_g Rect_M(\varphi)$.

Theorem 53 is nothing else but a consequence of Lemma 52 and Theorem 15 as our model-checking procedure is similar to the one of the μ-calculus over $(Agds_\Sigma(M) \cup \Sigma)$-labelled transition systems.

Theorem 53 *There is an exponential time procedure that checks whether a process is a model of a formula.*

6 Comparison with Event-Recording Logic

Event-Recording Logic [Sor02] is an extension of the μ-calculus that has been introduced to describe properties on timed process. The extension is made on modal operators by considering modal operators of the form $\langle g, a\rangle$ and $[g, a]$.

6.1 Syntax and Semantics of ERL

Definition 54 Let Σ be a set of events, Var a set of variables. The set of formulas of Event-Recording Logic denoted by $\mathcal{F}_{erl}$ is the set of formulas given by the following grammar:

$$\varphi ::= tt \mid ff \mid X \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle g, a\rangle\varphi \mid [g, a]\varphi \mid \mu X.\varphi \mid \nu X.\varphi$$

where,

- a is an event from Σ
- g is a constraint from Gds_Σ
- X is a variable from Var

ERL formulas are interpreted over timed processes. Because, the meaning of a timed process is a $(\mathcal{V}_\Sigma \times \Sigma)$-labelled transition system, we give the interpretation of a formula over such type of transition systems. As a formula may contain free variables we will need a valuation of such variables. Given a valuation of variables $Val : Var \to \mathcal{P}(S)$ and a set of states $T \subseteq S$, the valuation $Val[X/T]$ is the valuation Val with the substitution that associates the set of states T with the variable X. Formally, for $Y \in Var$, $Val[X/T](Y) = T$ if $Y = X$ and $Val(Y)$ otherwise. We write $\mathcal{S}, s, Val \vDash_t \varphi$ when the formula φ holds in s or equivalenty s satisfies φ.

Definition 55 (Meaning of a formula over $(\mathcal{V}_\Sigma \times \Sigma)$-labelled transition systems) For a given $(\mathcal{V}_\Sigma \times \Sigma)$-labelled transition system $\mathcal{S}$, a given formula φ and an assignment $Val : Var \to \mathcal{P}(S)$, we define the satisfaction relation $\vDash_t$ inductively as follows:

- $\mathcal{S}, s, Val \vDash_t tt$.
- $\mathcal{S}, s, Val \vDash_t X$ if $s \in Val(X)$.
- $\mathcal{S}, s, Val \vDash_t \varphi_1 \vee \varphi_2$ if $\mathcal{S}, s, Val \vDash_t \varphi_1$ or $\mathcal{S}, s, Val \vDash_t \varphi_2$.
- $\mathcal{S}, s, Val \vDash_t \varphi_1 \wedge \varphi_2$ if $\mathcal{S}, s, Val \vDash_t \varphi_1$ and $\mathcal{S}, s, Val \vDash_t \varphi_2$.
- $\mathcal{S}, s, Val \vDash_t [g, a]\psi$ if for every $s \xrightarrow{v,a} s' \in \Delta_\mathcal{S}$ such that $v \in [\![g]\!]$ we have $\mathcal{S}, s', Val \vDash_t \psi$.
- $\mathcal{S}, s, Val \vDash_t \langle g, a\rangle\psi$ if there exists $s \xrightarrow{v,a} s' \in \Delta_\mathcal{S}$ such that $v \in [\![g]\!]$ and $\mathcal{S}, s', Val \vDash_t \psi$.
- $\mathcal{S}, s, Val \vDash_t \mu X.\varphi(X)$ if $s \in \cap\{T \subseteq S \mid [\![\varphi(X)]\!]^\mathcal{S}_{Val[X/T]} \subseteq T\}$.
- $\mathcal{S}, s, Val \vDash_t \nu X.\varphi(X)$ if $s, \in \cup\{T \subseteq S \mid T \subseteq [\![\varphi(X)]\!]^\mathcal{S}_{Val[X/T]}\}$.

The meaning $[\![\varphi]\!]^\mathcal{S}_{Val}$ of a formula over $\mathcal{S}$ is a subset of S defined by

$$[\![\varphi]\!]^\mathcal{S}_{Val} = \{s \mid \mathcal{S}, s, Val \vDash_t \varphi\}.$$

We will sometimes write $s \in [\![\varphi]\!]^\mathcal{S}_{Val}$ instead of $\mathcal{S}, s, Val \vDash_t \varphi$. If φ is a *sentence*, i.e., does not have free variables, then its meaning does not depend on a valuation and we can write just $\mathcal{S}, s \vDash_t \varphi$. Finally, we will write $\mathcal{S} \vDash_t \varphi$ for $\mathcal{S}, s^0 \vDash_t \varphi$ to say that $\mathcal{S}$ is a *model* of φ.

Remark: The presentation of the semantics above is different (but it is equivalent) from the one in [Sor02]. In particular, the presentation of the semantics of modal operators indexed with a constraint and an event seems simpler and it benefits from that delay transitions in the semantics of timed processes (see Definition 39) are labelled with valuations.

Let us consider φ a formula and $\mathcal{P}$ a timed process. We say that φ is satisfied in a state p, a valuation v and a valuation $Val : Var \to \mathcal{P}(P \times \mathcal{V}_\Sigma)$ of propositional variables and we write $\mathcal{P}, (p, v), Val \vDash \varphi$ when $[\![\mathcal{P}]\!], (p, v), Val \vDash_t \varphi$.

6.2 WT_μ is more expressive that ERL

We show that ERL is a fragment of WT_μ. With an example, we show that modal operators we have introduced are useful for describing some relevant real-time properties on timed processes in particular the necessity modal property on time delay; that is, to require the existence of a discrete transition for all the time successors satisfying some timing constraints.

Proposition 56 Consider a property that can be written using a WT_μ formula φ or an ERL formula ψ, then for every timed process $\mathcal{P}$, state p of $\mathcal{P}$ and valuation $v \in \mathcal{V}_\Sigma$,

- $\mathcal{P}, (p, v), Val \vDash_t \langle g \rangle \langle a \rangle \varphi$ if and only if $\mathcal{P}, (p, v), Val \vDash_t \langle g, a \rangle \psi$.
- $\mathcal{P}, (p, v), Val \vDash_t [g][a]\varphi$ if and only if $\mathcal{P}, (p, v), Val \vDash_t [g, a]\psi$.

Lemma 57 There is a property that can be described with a WT_μ formula and that can not be described with an ERL formula.

Proof
Consider the property *"in the time interval* $(0, 1)$ *there is a time instance when no action* a *is possible.*. This property can be expressed by WT_μ formula:

$$\varphi = \langle 0 \leq h_a < 1 \rangle [a] ff$$

Observe that we use the clock associated to action a, but we could use any other clock as we assume that initial valuation of all clocks is 0. Of course φ is satisifiable, moreover it is consistent with the formula

$$\varphi' = \langle 0 \leq h_a < 1 \rangle \langle a \rangle tt$$

saying that there is a time instance when a is possible. We show that φ is not equivalent to a ERL formula. We claim that any ERL formula consistent with φ' is not equivalent to φ. Indeed, every ERL formula can be transformed into a boolean combination of formulas starting with modalities $\langle g, b \rangle$ or $[g, b]$. It is easy to verify that every such formula that is consistent with φ' has a model where action a is possible at every time instance between 0 and 1. □

In consequence of Lemma 57 and Proposition 56 we get the following.

Theorem 58 *WT_μ is strictly more expressive than ERL*

Example: Assume that we aim at checking the following property of timed process in Figure 8.

The system operates at any time within the 10 time units after the first ***d*** *signal by sending a* ***s*** *signal or by receiving a* ***b*** *signal.*
This property is described with the following WT_μ formula:

$$\varphi ::= [tt][d][h_d \leq 10]\langle d \rangle (\langle h_b \leq 10 \rangle \langle b \rangle tt \vee \langle h_d > 10 \rangle \langle s \rangle)$$

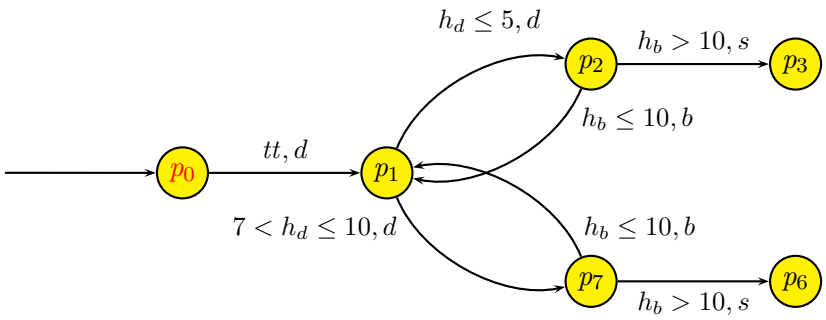

Figure 8: A toy-car model.

The system modeled in Figure 8 is not a model of φ. For example, if the second "danger signal" occurs 6 time units after the first "danger signal" the system will never compute the following "brake signal" unless another "danger signal" occurs 2 time units after the second. So there is a risk that the car goes into collision. $\square$

The formula φ in the example above can not be described with an ERL formula. Indeed, WT_μ formulas and ERL formulas are closed under the negation operator. If φ is an ERL formula, then $\neg(\varphi)$ should also be an ERL formula. But $\neg(\varphi)$ contains the formula $\langle h_d \leq 10\rangle[d]$ that is not and ERL formula (see Lemma 57 above).

7 Conclusion

The paper has introduced the logic WT_μ as a real-time fixpoint logic for describing properties on real-time systems modeled with timed processes (timed processes are event-recording automata without acceptance condition). We have consider the model-checking problem for WT_μ. We have used the notions of region and bisimilarity between representations of semantics of timed process for reducing the model-checking problem of WT_μ to the model-checking problem of the standard μ-calculus. In consequence we have got an exponential time algorithm for the model-checking problem of WT_μ.
We compared WT_μ with Event-recording logic when both logics are interpreted over timed processes. WT_μ is strictly more expressive than Event-recording logic. In particular, WT_μ enables to require an occurrence of an event at all the times at which a constraint is satisfied.

Our current work in progress with WT_μ includes: the logical characterization of event-recording automata and the satisfiability checking.

8 Acknowledgement

The author wishes to thank Igor Walukiewicz for having directed this work and for its fruitful comments.

References

[ACD+92] Rajeev Alur, Costas Courcoubetis, David L. Dill, Nicolas Halbwachs, and Howard Wong-Toi. An implementation of three algorithms for timing verifica-

tion based on automata emptiness. In *Proceedings of the 13th Symposium on Real-Time Systems(RTS'92:)*, pages 157–166. IEEE Computer Society Press, december 1992.

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

[AFH99] Rajeev Alur, Limor Fix, and Thomas A. Henzinger. Event-clock automata: A determinizable class of timed automata. *Theoretical Computer Science*, 211(1-2):253–273, 1999.

[AN01] André Arnold and Damian Niwiński. *Rudiments of μ-calculus*, volume 146 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 2001.

[BCL05] Patricia Bouyer, Franck Cassez, and François Laroussinie. Modal logics for timed control. In *Proceedings of the 16th International Conference on Concurrency Theory (CONCUR'05)*, volume 3653 of *Lecture Notes in Computer Science*, pages 81–94, San Francisco, CA, USA, august 2005. Springer.

[DM02] Deepak D'Souza and P. Madhusudan. Timed control synthesis for external specifications. In *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS '02)*, pages 571–582, London, UK, 2002. Springer-Verlag.

[EJ91] E. Allen Emerson and Charanjit S. Jutla. Tree automata, μ-calculus and determinacy. In *Proceedings of the 32nd annual symposium on Foundations of computer science (SFCS '91)*, pages 368–377, Washington, DC, USA, 1991. IEEE Computer Society.

[Eme90] E. Allen Emerson. *Temporal and modal logic*, volume B: formal models and semantics, pages 995–1072. MIT Press, Cambridge, MA, USA, 1990.

[GH82] Yuri Gurevich and Leo Harrington. Trees, automata, and games. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing (STOC '82)*, pages 60–65, New York, NY, USA, 1982. ACM.

[HLY91] Uno Holmer, Kim Larsen, and Wang Yi. Deciding properties of regular real timed processes. In *Proceedings of the 3th International Conference on Computer Aided Verification (CAV '91)*, volume 575 of *Lecture Notes in Computer Science*, pages 432–442. Springer-Verlag, 1991.

[Jur00] Marcin Jurdzinski. Small progress measures for solving parity games. In *Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science(STACS '00)*, pages 290–301, London, UK, 2000. Springer-Verlag.

[Koz82] Dexter Kozen. Results on the propositional μ-calculus. In *Proceedings of the 9th International Colloquium on Automata, Languages and Programming (ICALP'82)*, pages 348–359, 1982.

[LLW95] François Laroussinie, Kim G. Larsen, and Carsten Weise. From timed automata to logic - and back. In *Proceedings of the 20th International Symposium on Mathematical Foundations of Computer Science (MFCS '95)*, pages 529–539, London, UK, 1995. Springer-Verlag.

[LY97] Kim G. Larsen and Wang Yi. Time abstracted bisimulation: Implicit specification and decidability. *Information and Computation*, 134:75–103, 1997.

[Sor02] Maria Sorea. A decidable fixpoint logic for time-outs. In *Proceedings of the 13th International Conference on Concurrency Theory (CONCUR '02)*, pages 255–271, London, UK, 2002. Springer-Verlag.

[SS95] Oleg Sokolsky and Scott A. Smolka. Local model checking for real-time systems (extended abstract). In *Proceedings of the 7th International Conference on Computer Aided Verification (CAV'95)*, pages 211–224, London, UK, 1995. Springer-Verlag.

[Tho97] Wolfgang Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3. Springer-Verlag, 1997.

[VJ00] Jens Vöge and Marcin Jurdziński. A discrete strategy improvement algorithm for solving parity games. In *Proceedings of the 12th International Conference on Computer Aided Verification(CAV '00)*, pages 202–215, London, UK, 2000. Springer-Verlag.

[Zie98] Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998.