I N F O R M A T I K

The Logic of Event Clocks

Jean-François Raskin
Pierre-Yves Schobbens

MPI–I–99–3–002 August 1999

FORSCHUNGSBERICHT RESEARCH REPORT

M A X - P L A N C K - I N S T I T U T
F Ü R
I N F O R M A T I K

Im Stadtwald 66123 Saarbrücken Germany

Authors' Addresses

Jean-François Raskin
Max-Planck-Institut für Informatik
Im Stadtwald
66123 Saarbrücken
`jfr@mpi-sb.mpg.de`

Pierre-Yves Schobbens
Computer Science Department
University of Namur
Namur
Begium `pys@info.fundp.ac.be`

Publication Notes

A revised version of this report will be published in the journal of Automata, Languages and Combinatorics.

Acknowledgements

The authors would like to thank Thomas Henzinger for helpfull comments and insightfull discussions on the subject of real-time logics. This work was supported by the Belgian National Fund for Scientific Research (FNRS).

Keywords

Temporal Logic, Real-Time Logic, Decidability, Complexity, Expressiveness.

1 Introduction

It is now widely recognized that the use of formal methods is useful and often necessary for developing correct concurrent and reactive systems. This observation is even clearer when dealing with real-time [AH92b] and hybrid systems [Hen96]. Among the favorite formalisms to specify and verify concurrent systems are temporal logics. Temporal logics [Eme90, MP92] are modal logics that enable the expression of properties about the ordering of events in executions of concurrent programs [Pnu77]. For example, the linear temporal logic (LTL) formula $\Box(p \rightarrow \Diamond q)$ expresses the property that every p-event is always followed by some q-event. In that context, reactive systems are usually modeled by a product of finite state machines and properties of these systems are specified by temporal logic formulas. In the linear time framework[1], the verification problem, also called the model-checking problem, can be stated as follows: "Are all the possible executions of the reactive system modeled by the product of finite state machines, models of the temporal logic formula that specifies the property to verify?" or equivalently "is the ω-regular language defined by the product of automata included in the ω-regular language defined by the temporal formula?". Beside their nice expressive power (most important properties of reactive systems can be naturally expressed in temporal logic), the propositional fragments of those logics are decidable and are used in tools where the verification problem is automated [GPVW95, BCM+90].

The properties that can be expressed in propositional temporal logics are *qualitative* constraints about the ordering of events along a trace (infinite sequence of events that models an execution of a reactive system); *quantitative* timing constraints cannot be expressed. Logics that are able to express quantitative timing requirements are called *real-time logics* [AH94, AH93]. Real-time logics have received a lot of attention from the research community [Koy92, AH93, AH94, ACD90, AH92b]. The results about decidability of the real-time logics depends crucially on how the time is added to the traces that model reactive systems.

Semantically, there are two radically different ways to model time:

1. The first is to consider a discrete time domain, the natural numbers for example.

2. The other possibility is to use a dense time domain, as the real numbers or the rational numbers, for time stamps. Both are equivalent for our purpose. Choosing a dense time domain is more natural and presents advantages, compositionality for example; the interested reader is invited to consult [AH92b, HMP92, RS97a, BKP86] for a study and

[1] A framework where time is modeled by a branching structure can also be considered, see [BCM+90] for example and [Sti87] for a systematic comparison between the two approaches.

comparison of the two approaches. Unfortunately, when modeling time with a dense time domain, a lot of problems related to real-time logics become undecidable.

Having chosen the time domain, there are still two common ways to introduce real-time information into traces:

1. The *pointwise* way, that we adopt for the main part of the paper, consists in associating a time stamp (from the chosen time domain) with each observation of the trace. The intuitive meaning is that the observation of an event occurred at the time indicated by the time stamp. Those traces are called *timed traces.*

2. The *continuous* way consists in associating an interval with each observation of the trace. Intuitively, this interval indicates the interval of time during which the system is in the state described by the observation. Those traces are called *timed state sequences.*

Syntactically, there are two natural ways of extending temporal logics with timing constraints. The Metric Temporal Logic MetricTL (also called MTL [AH93]) adds time bounds to temporal operators; for example, the MetricTL formula $\Box(p \rightarrow \Diamond_{=1} q)$ specifies that every p event is followed by a q event such that the difference between the two time stamps is exactly 1. The Clock Temporal Logic ClockTL (also called TPTL [AH94]) adds clock variables to LTL; for example, the time-bounded response requirement from above can be specified by the ClockTL formula $\Box(p \rightarrow (x := 0)\Diamond(q \wedge x = 1))$, where x is a variable representing a clock that is started by the quantifier $(x := 0)$. Interestingly, over natural-numbered time, both ways of expressing timing constraints are equally expressive. Furthermore, the satisfiability problems of the two logics are decidable.

If time stamps are real instead of natural numbers, then the situation seems much less satisfactory. In fact, the logic MetricTL associated with a dense time domain allows the encoding of Turing machines computations and the halting problem of Turing machines can be reduced to the satisfiability of a MetricTL formula. The excessive expressive power of MetricTL comes from formulas such as $\Box(p \rightarrow \Diamond_{=1}q)$ that allow us to relate two arbitrary events that are separated by exactly one time unit. This ability coupled with the density of the time domain, permits us to relate consecutive contents of the memory of a Turing machine, contents that can be encoded using an interval of one time unit. The problem is the same with the logic ClockTL, as the formula $\Box(p \rightarrow (x := 0)\Diamond(x = 1 \wedge q))$ expresses the same property. Hence the search for decidable subsets of MetricTL and ClockTL is an interesting and important issue.

A first restriction to obtain a decidable logic concerns the style of specifying timing constraints using time-bounded temporal operators. The Metric-Interval Logic MetricIntervalTL (also called MITL [AFH96]) is obtained from

MetricTL by restricting the time bounds on temporal operators to nonsingular intervals. For example, the MetricIntervalTL formula $\Box(p \to \Diamond_{[0.9,1.1]}\, q)$ specifies that every p event is followed by a q event such that the difference between the two time stamps is at least 0.9 and at most 1.1. The restriction to non-singularity prevents specifying exact real time differences between events.

In this paper, we propose an alternative restriction, quite different in flavour, that concerns the style of specifying timing constraints using clock variables. The Event-Clock Logic rEventClockTL (also called SCL [RS97b]) is obtained from ClockTL by restricting the use of clocks to refer to the times of previous and next occurrences of events only. For example, if y_q is a clock that always refers to the time difference between now and the next q event, then the rEventClockTL formula $\Box(p \to y_q = 1)$ specifies that every p event is followed by a q event such that the difference between time stamps of the p event and the first subsequent q event is exactly 1. A clock such as y_q, which is permanently linked to the next q event, does not need to be started explicitly, and is called an *event clock*. The restriction to event clocks prevents the specification of time differences between a p event and any subsequent (rather than the first subsequent) q event.

The idea to associate clocks with events has first been introduced in the context of timed automata in [AFH94] where they propose a determinizable class of timed automata called Event Clock Automata (EventClockTA). As we will see later, in those automata, each clock is associated with an atomic event (a proposition for example). The main contribution of this paper is to show how this concept of event clocks can be generalized: we show that clocks can not only be associated with atomic propositions but recursively with temporal formulas. By defining rEventClockTL, we introduce the nice concept of event clock in the domain of real-time logics. Furthermore, we show that the logic of event clocks is quite expressive, in fact, most important real-time properties have a nice and direct formulation in rEventClockTL. Finally we show that the satisfiability problem for rEventClockTL is decidable, we characterize its complexity and present a decision procedure. This procedure can also be used to solve the real-time model checking problem: "Is the timed ω-regular language defined by a product of timed automata contained in the timed ω-regular language defined by an rEventClockTL formula?".

The rest of this paper is organized as follows. Real-time models are formally defined in section 2. In section 3 we recall EventClockTA. The logic rEventClockTL is defined in section 4 and its expressive power is illustrated by showing how to specify most important real-time requirements with rEventClockTL formulas. Section 5 proposes a decision procedure for the satisfiability problem of rEventClockTL formulas and proves its correctness. The decision procedure relies on the construction, for each formula of rEventClockTL, of a suitable EventClockTA whose language is empty if and only if the associated formula is not satisfiable. This EventClockTA needs auxiliary symbols when the formula is recursive. The complexity of the satisfiability problem is also studied there. Section 6 deals with expressiveness: $\mathsf{rEventClockTL}$ as expressive as $\mathsf{MetricIntervalTL}_{0,\infty}$, but less expressive than $\mathsf{MetricIntervalTL}$in dense pointwise models. This contrasts with continuous models, where $\mathsf{MetricIntervalTL}$ is as expressive as $\mathsf{MetricIntervalTL}_{0,\infty}$ as shown in [HRS98, RSH98]. The same property holds for the future fragments, which are each less expressive than their logics with past. We also compare the expressive power of $\mathsf{rEventClockTL}$ and $\mathsf{EventClockTA}$. They turn out to be incomparable, since $\mathsf{rEventClockTL}$ allows recursion, while $\mathsf{EventClockTA}$ allow counting.

2 Real-Time Models

The execution of a reactive system can be modeled by an infinite sequence of observations $\bar{\sigma} = \sigma_0\sigma_1 \dots \sigma_n \dots$, where each $\sigma_i \subseteq \mathcal{P}$ (a subset of propositions that describes the observed state of the system). Such a sequence is called a *trace*. When considering executions of *real-time* reactive systems, timing information about the occurrence of the observations must be added to traces. As mentioned in the introduction, we consider a dense time domain: the nonnegative real numbers. We present the details in the context of timed traces since it will slightly facilitate the presentation of the region construction in section 5. For the interested reader, we give in annex the definition of the logic $\mathsf{rEventClockTL}$ in the context of timed state sequences. There, we recall the decidability and complexity results for the logic that are the same for the two models.

Definition 1 A *timed trace* is a pair $\theta = (\bar{\sigma}, \bar{\tau})$ where $\bar{\sigma}$ is a trace and $\bar{\tau} = \tau_1\tau_2 \dots \tau_n \dots$ is an infinite sequence of positive real numbers, called a timing, representing the time at which each observation occurred. Furthermore the timing $\bar{\tau} = \tau_0\tau_1 \dots \tau_n \dots$ respects (i) *monotonicity*: for all $i \geq 0$, $\tau_i < \tau_{i+1}$, (ii) *divergence*: for all $t \in \mathbb{R}^+$, there exists i such that $\tau_i > t$.

3 Event Clock Automata

Timed automata [AD94] are finite state machines extended with clocks. Clock can be reset and compared to integer constants. Unfortunately, the formalism of timed automata is not closed under complement. This is due to the fact that, in timed automata, clocks can be reset nondeterministically. This feature allows the specifier to define the timed language of the negation of the formula that allows the encoding of Turing machine computations in $\mathsf{MetricTL}$, see [AD94] for details and examples.

In [AFH94], Alur et al. present a determinizable class of timed automata called *event clock automata*. This class of automata is closed under *union,* *intersection and complement.* Consequently the language inclusion problem is decidable for this class of automata. For event clock automata, the complement closure property is obtained by restricting the use of clocks: the clocks have a predefined association with symbols of the input alphabet. Clocks are reset implicitly whenever their event occurs. This resetting is thus determined by the timed trace the automaton is reading, which is key to their determinization. The *event-history clock* of the input symbol $a \in \Sigma$, denoted x_a, is a history variable whose value is the time elapsed since the *last* occurrence of a relative to the current time. Symmetrically, the *event-prophecy clock* of $a \in \Sigma$, denoted y_a is a prophecy variable whose value is the time to wait for the *next* occurrence of a relative to the current time.

Example 1 Let us consider the automaton of figure 1. This event-clock automaton contains 3 locations, l_0 is the start location. The constraint $x_a = 5$ decorating the edge starting from l_1 with the character b imposes that a previous a character must have been read exactly 5 time units before the edge is crossed. On the other hand the constraint $y_a < 2$ decorating the edge from l_1 to l_2 requires that each time this edge is crossed, the next a-edge must be crossed within 2 time units.

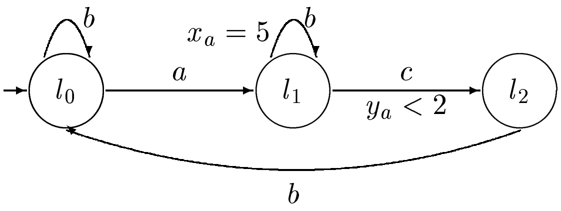

Figure 1: Event-Clock automaton A_1.

Let us consider the execution of the automaton on the timed trace $(\bar{\sigma}, \bar{\tau}) = (a, 1), (b, 6), (c, 7), (b, 7.3), (b, 7.5), (a, 8), (c, 11), \ldots$. The automaton starts at location l_0. At time $t = 1$, the automaton reads a and goes to l_2. At time $t = 6$ it reads b and checks that the previous a in $\bar{\sigma}$ is distant of exactly 5 time units, and so on. Thus $(\bar{\sigma}, \bar{\tau})$ is a possible prefix of a timed trace accepted by the automaton.

As we can see in example 1, the values of the clocks are solely determined by the input word, not by the automaton. Thanks to this important feature EventClockTA are determinizable and can be complemented.

Definition 2 (Event Clocks) Given a set of proposition $\mathcal{P}$, the set of *event clocks* associated to $\mathcal{P}$ is the set $\mathbb{C} = \mathbb{H} \cup \mathbb{P}$ where $\mathbb{H} = \{x_p \mid p \in \mathcal{P}\}$ is the set of *history clocks*, i.e. an history clock x_p is associated to each proposition $p \in \mathcal{P}$, and where $\mathbb{P} = \{y_p \mid p \in \mathcal{P}\}$ is the set of *prophecy clocks*, i.e. a prophecy clock is associated to each proposition of $\mathcal{P}$. In what follows, we note $x \in \mathbb{H}$ any history clock of $\mathbb{C}$, y any prophecy clock of $\mathbb{C}$, z_p the history clock or the prophecy clock associated to p and z any clock of $\mathbb{C}$. $\square$

We now define formally the value of history and prophecy clocks along a timed trace. We use $\mathbb{R}^+ \cup \{\bot\}$ to denote the nonnegative real numbers together with the special value $\bot$ (undefined).

Definition 3 The *value* of the history clock $x_p \in \mathbb{H}$ associated with the proposition p at position i of the timed trace $(\bar{\sigma}, \bar{\tau})$, denoted $\mathsf{Val}_{x_p}(\bar{\sigma}, \bar{\tau}, i)$, is defined as follows:

$$\mathsf{Val}_{x_p}(\bar{\sigma}, \bar{\tau}, i) = \begin{cases} \tau_i - \tau_j & \text{if there exists } j \text{ such that } 0 \leq j < i,\ p \in \sigma_j \\ & \text{and for all } k \text{ such that } j < k < i,\ p \notin \sigma_k \\ \bot & \text{if for all } j,\ 0 \leq j < i,\ p \notin \sigma_j \end{cases}$$

The *value* of the prophecy clock $y_p \in \mathbb{P}$ associated with the proposition p at position i of the timed trace $(\bar{\sigma}, \bar{\tau})$, denoted $\mathsf{Val}_{y_p}(\bar{\sigma}, \bar{\tau}, i)$, is defined as follows:

$$\mathsf{Val}_{y_p}(\bar{\sigma}, \bar{\tau}, i) = \begin{cases} \tau_j - \tau_i & \text{if there exists } j \text{ such that } i < j,\ p \in \sigma_j \\ & \text{and for all } k \text{ such that } i < k < j,\ p \notin \sigma_k \\ \bot & \text{if for all } j,\ i < j,\ p \notin \sigma_j \end{cases}$$

Constraints about the value of clocks are used to express real-time requirements on the occurrences of events.

Definition 4 A *clock constraint* is a boolean combination of formulas of the form $z \sim c$ where $z \in \mathbb{C}$ is a history or a prophecy clock, $\sim \in \{<, \leq, =, \geq, >\}$ and c is an integer constant.

Clock constraints are evaluated in positions of timed traces. Here are the rules of evaluation:

Definition 5 A timed trace θ *satisfies* a clock constraint ψ at a position i according to the following usual rules:

- $(\theta, i) \models z \sim c$ iff $\mathsf{Val}_z(\bar{\sigma}, \bar{\tau}, i) \sim c$;
- $(\theta, i) \models \neg\psi$ iff not $(\theta, i) \models \psi$;
- $(\theta, i) \models \psi_1 \vee \psi_2$ iff $(\theta, i) \models \psi_1$ or $(\theta, i) \models \psi_2$.

where $\sim$ are evaluated as usual in nonnegative real numbers and $\bot \sim c$ always evaluates to false.

Definition 6 An $\mathsf{EventClockTA}$ is 6-tuple $A = (L, L_0, \mathcal{P}, \mathbb{C}, E, \mathcal{F})$ where:

- L is a finite set of *locations*;

- $L_0 \subseteq L$ is the subset of *start locations*;
- $\mathcal{P}$ is a finite set of *propositions*;
- $\mathbb{C}$ is a set of clocks partitioned into a set $\mathbb{H}$ of history clocks and a set $\mathbb{P}$ of prophecy clocks;
- E is a finite set of edges; each edge is a quadruple (l_1, l_2, s, ψ) where $l_1 \in L$ is the source location, $l_2 \in L$ is the target location, $s \subseteq \mathcal{P}$ is a state description and ψ is a clock constraint;
- $\mathcal{F} = \{F_1, \ldots, F_n\}$ with each $F_i \subseteq L$, is a set of sets of *accepting locations*. (generalized Büchi acceptance condition).

As finite state automata define set of traces, that are called languages, EventClockTA define set of timed traces, that are called *timed languages*. To define formally the timed language defined by an EventClockTA, we first introduce the notion of computation of an EventClockTA:

Definition 7 An *accepted computation* of an EventClockTA A on a timed trace θ is an infinite sequence

$$\gamma = l_0 \xrightarrow{e_0} l_1 \xrightarrow{e_1} \cdots l_n \xrightarrow{e_n} \cdots$$

where each $l_i \in L$, and:

(C1) $l_0 \in L_0$ (initiality);

(C2) $e_i = (l_i, l_{i+1}, s_i, \psi_i) \in E$ (consecution), and:

  (C3) $(\bar{\sigma}, \bar{\tau}, i) \models \psi_i$ (timing);

  (C4) $s_i = \sigma_i$ (adequacy);

(C5) for every $F_i \in \mathcal{F}$, there exists infinitely many positions j such that $l_j \in F_i$ (generalized Büchi acceptance).

Definition 8 The *timed language* of an EventClockTA A, denoted $L(A)$, is the set of timed traces for which A has an accepted computation.

The formalism of EventClockTA is closed under all boolean operations:

Theorem 1 *[AFH94] For every* EventClockTA *A_1 and A_2, we can construct an* EventClockTA *$A_1 + A_2$ that accepts the union of the languages of A_1 and A_2, i.e. $L(A_1 + A_2) = L(A_1) \cup L(A_2)$, an* EventClockTA *$A_1 \times A_2$ that accepts the intersection of the languages of A_1 and A_2, i.e. $L(A_1 \times A_2) = L(A_1) \cap L(A_2)$, for every* EventClockTA *A, we can construct an an* EventClockTA *$\bar{A}$ that accepts the complement of the language of A, i.e. $L(\bar{A}) = \overline{L(A)}$.*

4 The Event Clock Logic

In this section, we introduce event clocks in temporal logic call this rEventClockTL. This logic is a real-time extension of the usual temporal logic. We extend LTL (with past operators) by two indexed modal operators $\rhd$ and $\lhd$ which express real-time constraints. The semantics of those two operators is closely related to the notions of *prophecy* and *history* clock variables. The formula $\rhd_{\sim c}\ \phi$ expresses that the delay before the next observation of ϕ satisfies constraint $\sim c$; symmetrically, the formula $\lhd_{\sim c}\ \phi$ constrains the previous observation of ϕ. The modal operators $\rhd$ and $\lhd$ generalize the semantics of history/prophecy variables of [AFH94]: They are more general in that they allow *recursion*, i.e. the operators can constrain any formula ϕ rather than proposition symbols. As we show later, all interesting properties of EventClockTA are preserved in our logic, even though it is more expressive. We now present formally the rEventClockTL logic. Examples of specifications written in rEventClockTL are given at the end of this section.

Definition 9 A *formula* of rEventClockTL is composed of proposition symbols $p, p_1, p_2, .., q, ...$, usual boolean connectives $\vee$ and $\neg$, qualitative temporal operators: Until (U) and Since (S), real-time operators: prophecy operator ($\rhd$), history operator ($\lhd$). A well-formed formula of rEventClockTL satisfies the following syntactical rule:

$$\phi ::= p \mid \phi_1 \vee \phi_2 \mid \neg\phi \mid \ \bigcirc\phi \ \mid \ \ominus\phi \ \mid \ \phi_1 U \phi_2 \mid \phi_1 S \phi_2 \mid \rhd_{\sim c}\ \phi \mid \lhd_{\sim c}\ \phi$$

where $\sim \in \{<, \leq, =, \geq, >\}$, $p \in \mathcal{P}$ and ϕ, ϕ_1, ϕ_2 are well formed formulas and c is an integer constant

We use the usual precedence of operators: modal operators are more binding than boolean ones, and their scope is as small as possible. A formula is *non-recursive* if the real-time operators only contain proposition symbols: the clauses $\rhd_{\sim c}\ p \mid \lhd_{\sim c}\ p$ replace $\rhd_{\sim c}\ \phi \mid \lhd_{\sim c}\ \phi$ in the syntax.

Definition 10 A timed trace $\theta = (\bar{\sigma}, \bar{\tau})$ *satisfies* at position i an rEventClockTL formula ϕ when:

- $(\theta, i) \models p$ iff $p \in \sigma_i$;
- $(\theta, i) \models \neg\phi$ iff not $(\theta, i) \models \phi$;
- $(\theta, i) \models \phi_1 \vee \phi_2$ iff $(\theta, i) \models \phi_1$ or $(\theta, i) \models \phi_2$;
- $(\theta, i) \models \bigcirc\phi$ iff $(\theta, i+1) \models \phi$;
- $(\theta, i) \models \ominus\phi$ iff $i > 0$ and $(\theta, i-1) \models \phi$;
- $(\theta, i) \models \phi_1 U \phi_2$ iff there exists $j \geq i$ such that $(\theta, j) \models \phi_2$ and for all k, $i \leq k < j$, $(\theta, k) \models \phi_1$;
- $(\theta, i) \models \phi_1 S \phi_2$ iff there exists j, $0 \leq j \leq i$ such that $(\theta, j) \models \phi_2$ and for all k, $j < k \leq i$, $(\theta, k) \models \phi_1$;
- $(\theta, i) \models \rhd_{\sim c}\ \phi$ iff there exists $j > i$, such that $(\theta, j) \models \phi$, for all k, $i < k < j$, $(\theta, k) \not\models \phi$ and $\tau_j - \tau_i \sim c$;

$(\theta, i) \models \lhd_{\sim c} \phi$ iff there exists j, $0 \leq j < i$, such that $(\theta, j) \models \phi$, for all k, $j < k < i$, $(\theta, k) \not\models \phi$ and $\tau_i - \tau_j \sim c$.

As usual, we can define other boolean and temporal operators as syntactical abbreviations:

- **boolean**: $\top \equiv \neg\phi_1 \vee \phi_1$, $\bot \equiv \neg\top$, $\phi_1 \wedge \phi_2 \equiv \neg(\neg\phi_1 \vee \neg\phi_2)$, $\phi_1 \rightarrow \phi_2 \equiv \neg\phi_1 \vee \phi_2$, $\phi_1 \leftrightarrow \phi_2 \equiv \phi_1 \rightarrow \phi_2 \wedge \phi_2 \rightarrow \phi_1$;
- **for the future**:
  - $\Diamond\phi_1 \equiv \top U \phi_1$, meaning "eventually in the present or future";
  - $\Box\phi_1 \equiv \neg\Diamond\neg\phi_1$ "always in the present and future";
  - $\rhd_{[l,u]}\ \phi \equiv \rhd_{\geq l}\ \phi \wedge\ \rhd_{\leq u}$, "next ϕ occurs between l and u from now";
  - similarly, $\rhd_I\ \phi \equiv \rhd_{\sim l}\ \phi \wedge\ \rhd_{\sim u}$, "next ϕ occurs within I from now", where I is an interval with bounds l, u and $\sim$ is the adequate constraint;
  - $\phi_1 U_{\sim c} \phi_2 \equiv \phi_1 U \phi_2 \wedge (\rhd_{\sim c}\ \phi_2 \vee \phi_2)$, with $\sim \in \{<, \leq\}$, meaning "Until within c next";[2]
  - $\Box_{\sim c}\phi_1 \equiv \neg \rhd_{\sim c} \neg\phi_1$, with $\sim \in \{<, \leq\}$, meaning "always for the following c time units";
- **for the past**:
  - $\Diamond\!\!\!\!-\ \phi_1 \equiv \top S \phi_1$, meaning "eventually in the past or present";
  - $\boxminus\phi_1 \equiv \neg\Diamond\!\!\!\!-\ \neg\phi_1$, meaning "always in the past and present";
  - $\lhd_{[l,u]}\ \phi \equiv \lhd_{\geq l}\ \phi \wedge\ \lhd_{\leq u}$, "last ϕ occurred between l and u ago";
  - similarly, $\lhd_I\ \phi \equiv \lhd_{\sim l}\ \phi \wedge\ \lhd_{\sim u}$, "last ϕ occurred within I ago", where I is an interval with bounds l, u and $\sim$ is the adequate constraint;
  - $\phi_1 S_{\sim c} \phi_2 \equiv \phi_1 S \phi_2 \wedge (\lhd_{\sim c}\ \phi_2 \vee \phi_2)$, with $\sim \in \{<, \leq\}$, meaning "since within c";
  - $\boxminus_{\sim c}\phi_1 \equiv \neg \lhd_{\sim c} \neg\phi_1$, with $\sim \in \{<, \leq\}$, meaning "always in the past and present within c";

Definition 11 A rEventClockTL formula ϕ defines a *timed language* $L(\phi)$: the set of timed traces θ such that $(\theta, 0) \models \phi$.

[2] If this definition is used with $>, \geq$ or $=$, it has a different meaning than the until of MetricIntervalTL.

Example 2 Here are some examples of $\mathsf{rEventClockTL}$ formulas with their verbal meaning. These examples cover all typical real-time requirements classified in [Koy92]. In these examples, we mainly use atomic events p, q, ... for readability but they can be replaced by more complicated $\mathsf{rEventClockTL}$ formulas. The verbal interpretation characterizes intuitively the infinite timed traces θ that the formula ϕ defines.

- $\Box q$: q is always true. Such a formula asserts *invariance* properties of a system.
- $\Box(p \to \rhd_{\leq 5}\, q)$: a p position is always followed by a q position within 5 time units. Such a formula specifies a maximal distance between a request p and its response q. Such a property is called a *bounded time response*. Here, it assumes that only one request can be outstanding.
- $\Box(p \to \rhd_{=3}\, q)$: when a p position is encountered, the first following q position is at exactly 3 time units. Such a formula allows the assertion of *exact response time* (assuming no intervening request p).
- $p \wedge \Box(p \to \rhd_{=1}\, p)$: this formula asserts that p is true every integer time unit. Such a formula allows the specifier to define *periodicity of events*. Here p can model the tick of an ideal clock, that ticks every time unit.
- $\Box(p \to (\rhd_{>5}\, p) \vee (\bigcirc\Box\neg p))$: every p event is either followed by another p event distant of more than 5 time units or never followed by another p event. This formula expresses a *minimal distance between events*, for example the rate of input from the environment.
- $\Diamond\Box q$: q will eventually hold permanently.
- $\Box((\lhd_{=3}\, q) \to p)$. This formula asserts that if the last q position is exactly distant of 3 time units then p must be true now. It is a typical *time-out requirement*.
- $\Box(q \to (p S_{\leq 3} r))$. When a q position is encountered then the last r position is distant at most of 3 time units and all intermediary positions were p positions.
- $\Box((\boxminus_{<3}\neg p) \to q)$. If p is constantly false during the last 3 time units then q is true now. This is a typical example of the specification of an *alarm* q if a monitored event p does not occur within a fixed delay.

As we can see, the $\mathsf{rEventClockTL}$ logic is quite expressive. Most of the properties that are encountered when dealing with real-time systems, e.g. bounded response time, bounded invariance, time-out,..., can be easily and elegantly specified. In practice, the use of the $\mathsf{MetricTL}$ operator $\Diamond_{\sim c}$ can often be replaced by the stronger but less expressive operator $\rhd_{\sim c}$. The presence of the past operators is also a facility for the specifier. However, there are properties that cannot be expressed using $\mathsf{rEventClockTL}$ logic:

Example 3 Every p state is followed by a q state exactly 1 time unit later. Such a property can be expressed in $\mathsf{MetricTL}$ [AH93] as follows:

$$\Box(p \rightarrow \Diamond_{=1} q)$$

This property is **not** expressed by the $\mathsf{rEventClockTL}$ formula:

$$\Box(p \rightarrow \rhd_{=1} q)$$

which is stronger since it requires that the **first** q is at exactly 1 time unit.

In fact, as already mentioned in the introduction, if $\mathsf{rEventClockTL}$ could express this $\mathsf{MetricTL}$ property, the logic would be undecidable. In the next section, we show that it is not the case by defining a decision procedure for the satisfiability problem of $\mathsf{rEventClockTL}$. The expressive power of $\mathsf{rEventClockTL}$ is considered in section 6.

5 A Decision Procedure for $\mathsf{rEventClockTL}$

The principle of the decision procedure for LTL is to construct a Büchi automaton that accepts exactly the traces that are models of the formula and then to test the automaton for emptiness, see [Wol85, MP95] for details.

Here we propose a similar approach: for every $\mathsf{rEventClockTL}$ formula ϕ, we construct an $\mathsf{EventClockTA}$ A_ϕ whose timed language is empty if and only if the formula ϕ is not satisfiable. The procedure that we propose relies on a construction that uses the subformulas of ϕ.

Definition 12 The *closure set* of an $\mathsf{rEventClockTL}$ formula ϕ, denoted $\overline{\mathsf{Cl}}(\phi)$, is defined with the help of the recursive function Cl:

- $\mathsf{Cl}(p) = \{p\}$;
- $\mathsf{Cl}(\phi_1 \vee \phi_2) = \{\phi_1 \vee \phi_2\} \cup \mathsf{Cl}(\phi_1) \cup \mathsf{Cl}(\phi_2)$;
- $\mathsf{Cl}(\neg\phi_1) = \mathsf{Cl}(\phi_1)$;
- $\mathsf{Cl}(\bigcirc\phi_1) = \{\bigcirc\phi_1\} \cup \mathsf{Cl}(\phi_1)$;
- $\mathsf{Cl}(\ominus\phi_1) = \{\ominus\phi_1\} \cup \mathsf{Cl}(\phi_1)$;
- $\mathsf{Cl}(\phi_1 U \phi_2) = \{\phi_1 U \phi_2\} \cup \{\bigcirc(\phi_1 U \phi_2)\} \cup \mathsf{Cl}(\phi_1) \cup \mathsf{Cl}(\phi_2)$;
- $\mathsf{Cl}(\phi_1 S \phi_2) = \{\phi_1 S \phi_2\} \cup \{\ominus(\phi_1 S \phi_2)\} \cup \mathsf{Cl}(\phi_1) \cup \mathsf{Cl}(\phi_2)$;

- $\mathsf{Cl}(\rhd_{\sim c} \phi_1) = \{\rhd_{\sim c} \phi_1\} \cup \mathsf{Cl}(\phi_1)$;
- $\mathsf{Cl}(\lhd_{\sim c} \phi_1) = \{\lhd_{\sim c} \phi_1\} \cup \mathsf{Cl}(\phi_1)$;

To obtain $\overline{\mathsf{Cl}}(\phi)$, we close $\mathsf{Cl}(\phi)$ by negation and identify $\neg\neg\phi_1$ with ϕ_1 in any context to keep $\overline{\mathsf{Cl}}(\phi)$ finite. The set of atomic propositions appearing in ϕ is denoted $\mathcal{P}_\phi$. Note that $\mathcal{P}_\phi \subseteq \overline{\mathsf{Cl}}(\phi)$.

In our case, the EventClockTA A_ϕ does not accept the models of the formula ϕ but its timed Hintikka sequences. EventClockTA as defined in [AFH94] and recalled in section 3 are not expressive enough to define all rEventClockTL-timed languages, as shown in section 6. Nevertheless Event-ClockTA can be used to define a decision procedure for rEventClockTL as we show in this section.

Definition 13 The *timed Hintikka sequences* of ϕ are the timed traces θ defined on the set of propositions $\{p_\rho | \rho \in \overline{\mathsf{Cl}}(\phi)\}$ (i.e. a proposition is associated with each formula of $\overline{\mathsf{Cl}}(\phi)$) that satisfy the following requirements, for all $i \geq 0$:

(H1) $p_\phi \in \sigma_0$;

(H2) $p_\rho \in \sigma_i$ iff $p_{\neg\rho} \notin \sigma_i$;

(H3) $p_{\rho_1 \vee \rho_2} \in \sigma_i$ iff $p_{\rho_1} \in \sigma_i$ or $p_{\rho_2} \in \sigma_i$;

(H4) $p_{\bigcirc\rho} \in \sigma_i$ iff $p_\rho \in \sigma_{i+1}$;

(H5) $p_{\ominus\rho} \in \sigma_i$ iff $i > 0$ and $p_\rho \in \sigma_{i-1}$;

(H6) $p_{\rho_1 U \rho_2} \in \sigma_i$ iff there exists $j \geq i$ such that $p_{\rho_2} \in \sigma_j$ and for all k, $i \leq k < j$, $p_{\rho_1} \in \sigma_k$;

(H7) $p_{\rho_1 S \rho_2} \in \sigma_i$ iff there exists j, $0 \leq j \leq i$, such that $p_{\rho_2} \in \sigma_j$ and for all k, $j < k \leq i$, $p_{\rho_1} \in \sigma_k$;

(H8) $p_{\rhd_{\sim c}\rho} \in \sigma_i$ iff there exists $j > i$ such that $p_\rho \in \sigma_j$, for all k, $i < k < j$, $p_\rho \notin \sigma_k$ and $\tau_j - \tau_i \sim c$;

(H9) $p_{\lhd_{\sim c}\rho} \in \sigma_i$ iff there exists j, $0 \leq j < i$ such that $p_\rho \in \sigma_j$, for all k, $j < k < i$, $p_\rho \notin \sigma_k$ and $\tau_i - \tau_j \sim c$;

Requirements H2 and H3 ensure propositional consistency of timed Hintikka sequences, H4, H5, H6 and H7 ensure consistency with the semantics of temporal operators, and, H8 and H9 ensure consistency with the semantics of real-time operators. H1 is related to the following theorem:

Proposition 1 A rEventClockTL formula ϕ is satisfiable iff it has a timed Hintikka sequence.

Proof. Let us prove that given an Hintikka sequence $(\bar{\sigma}, \bar{\tau})$ for ϕ, the timed trace $(\bar{\sigma}', \bar{\tau})$, where $\sigma'_i = \{q | p_q \in \sigma_i \wedge q \in \mathcal{P}_\phi\}$, has the Hintikka property: for all formula $\rho \in \overline{\mathsf{Cl}}(\phi)$, $(\bar{\sigma}', \bar{\tau}, i) \models \rho$ iff $p_\rho \in \sigma_i$. We reason by induction on the structure of formulas. *Base case.* $\phi = q$, a proposition. By definition of $(\bar{\sigma}', \bar{\tau})$, $q \in \sigma'_i$ iff $p_q \in \sigma_i$. Thus by the definition of $\models$, we have $(\bar{\sigma}', \bar{\tau}, i) \models q$ iff $p_q \in \sigma_i$. *Induction case.* We treat two cases: $\phi = \phi_1 U \phi_2$ and $\phi = \rhd_{\sim c} \phi_1$, the other cases are treated similarly and left to the careful reader. By induction hypothesis, we know that: for all $i \geq 0$, $(\bar{\sigma}', \bar{\tau}, i) \models \phi_j$ iff $p_{\phi_j} \in \sigma_i$, for $j \in \{1, 2\}$. We now treat the two cases:

- $\phi = \phi_1 U \phi_2$. $(\bar{\sigma}', \bar{\tau}, i) \models \phi_1 U \phi_2$ is defined as "there exists $j \geq i$ such that $(\bar{\sigma}', \bar{\tau}, i) \models \phi_2$ and for all k, $i \leq k < j$, $(\bar{\sigma}', \bar{\tau}, i) \models \phi_1$". By induction hypothesis, there exists $j \geq i$ such that $p_{\phi_2} \in \sigma_j$ and for all k, $i \leq k < j$, $p_{\phi_1} \in \sigma_k$. By rule (H6) of the definition of timed Hintikka sequences, this is equivalent to $p_{\phi_1 U \phi_2} \in \sigma_i$.

- $\phi = \rhd_{\sim c} \phi_1$. $(\bar{\sigma}', \bar{\tau}, i) \models \rhd_{\sim c} \phi_1$ is defined as "there exists $j > i$ such that $(\bar{\sigma}', \bar{\tau}, j) \models \phi_2$, $\tau_j - \tau_i \sim c$ and for all k, $i < k < j$, $(\bar{\sigma}', \bar{\tau}, j) \models \phi_1$". By induction hypothesis, this is also: "there exists $j > i$ such that $p_{\phi_2} \in \sigma_j$, $\tau_j - \tau_i \sim c$ and for all k, $i < k < j$, $p_{\phi_1} \in \sigma_k$". By rule (H8), this is the same as $p_{\rhd_{\sim c}\phi_1} \in \sigma_i$.

As we have that for all $\phi_1 \in \overline{\mathsf{Cl}}(\phi)$ and for all $i \geq 0$, $(\bar{\sigma}', \bar{\tau}, i) \models \phi_1$ iff $p_{\phi_1} \in \sigma_i$, by rule (H1), we have that $p_\phi \in \sigma_0$ and thus $(\bar{\sigma}', \bar{\tau}, 0) \models \phi$. As a consequence, $(\bar{\sigma}', \bar{\tau})$ is a model of ϕ.

Now, let us consider the other direction. If $(\bar{\sigma}, \bar{\tau})$ is a model of ϕ we prove that the timed trace $(\bar{\sigma}', \bar{\tau})$, with $\sigma'_i = \{p_\rho | \rho \in \overline{\mathsf{Cl}}(\phi) \wedge (\bar{\sigma}, \bar{\tau}, i) \models \rho\}$, has the timed Hintikka property for ϕ. Again, the proof is by induction on the structure of formulas. The proof is easy since the Hintikka properties (H1-H8) express the semantics of the operators. □

Construction of A_ϕ

The locations of the $\mathsf{EventClockTA}$ A_ϕ will be subsets of $\overline{\mathsf{Cl}}(\phi)$. If a formula ρ belongs to a location l of A_ϕ, the intuitive meaning is that when the automaton A_ϕ is in location l then all the accepted timed traces passing through l are timed Hintikka sequences underlying the models of ρ. Obviously, all possible subsets of the closure set are not candidate for representing a position in a model. For example, a subset of $\overline{\mathsf{Cl}}(\phi)$ which contains both ρ and $\neg\rho$ cannot be a candidate for a position in a model as the conjunction of this set of formulas is not satisfiable. To make the notion of candidate for a model position clearer, we define the notion of atom.

Definition 14 An *atom* over ϕ is a subset $\Theta \subseteq \overline{\mathsf{Cl}}(\phi)$ satisfying the following requirements:

- Θ is propositionally consistent and complete. More formally:
  - (A1) For every $\rho_1 \in \overline{\mathsf{Cl}}(\phi)$, $\rho_1 \in \Theta$ iff $\neg\rho_1 \notin \Theta$;
  - (A2) For every $\rho_1 \vee \rho_2 \in \overline{\mathsf{Cl}}(\phi)$, $\rho_1 \vee \rho_2 \in \Theta$ iff $\rho_1 \in \Theta$ or $\rho_2 \in \Theta$.
- Θ respects local constraints of the U and the S operators:
  - (A3) For every $\rho_1 U \rho_2 \in \overline{\mathsf{Cl}}(\phi)$, $\rho_1 U \rho_2 \in \Theta$ iff either:
    - $\rho_2 \in \Theta$;
    - $\rho_1, \bigcirc(\rho_1 U \rho_2) \in \Theta$.
  - (A4) For every $\rho_1 S \rho_2 \in \overline{\mathsf{Cl}}(\phi)$, $\rho_1 S \rho_2 \in \Theta$ iff either:
    - $\rho_2 \in \Theta$;
    - $\rho_1, \ominus(\rho_1 S \rho_2) \in \Theta$.

We build the components of A_ϕ:

Propositions: It will accept timed traces defined on the set $\mathcal{P} = \{p_\rho \mid \rho \in \overline{\mathsf{Cl}}(\phi)\}$.

Clocks: It uses the clocks $\mathbb{C} = \{x_{p_\rho} \mid \lhd_{\sim c}\, \rho \in \overline{\mathsf{Cl}}(\phi)\} \cup \{y_{p_\rho} \mid \rhd_{\sim c}\, \rho \in \overline{\mathsf{Cl}}(\phi)\}$.

Locations: The locations L of A_ϕ are the atoms of $\overline{\mathsf{Cl}}(\phi)$, requirement denoted (L) in what follows.

Start locations: The start locations L_0 are the atoms Θ such that: (S1) $\phi \in \Theta$ and (S2) for all formula $\ominus\rho \in \overline{\mathsf{Cl}}(\phi)$, $\neg \ominus \rho \in \Theta$.

Let us now see how to define the edges of the automaton A_ϕ. First we examine when two locations must be linked by an edge. After we consider the labels that decorate edges. The formulas $\bigcirc\rho$ and $\ominus\rho$ of $\overline{\mathsf{Cl}}(\phi)$ are used to formulate the connection requirement of the automaton A_ϕ. $\bigcirc\rho \in l_1$ means that from the location l_1 all suffixes respect p_ρ in their second observation, or equivalently that from all locations l_2 that are connected to l_1, the accepted suffixes are suffixes where p_ρ is true at the first observation. Symmetrically for $\ominus$-formulas. As the suffixes starting from a location l_1 must satisfy the propositions associated with the set of formulas that belong to atom l_1 then the propositional labels are simply the propositions that are related to the formulas of l_1. To ensure the semantics of real-time formula, we simply use the history and prophecy clocks. If $\rhd_{\sim c}\, \rho \in l$ then all edges that start from l are labelled by the constraint $y_{p_\rho} \sim c$ that ensures the real-time rule (H8) of timed Hintikka sequences. The situation is similar for history formulas. We can now formulate more rigorously the edges of the automaton.

Edges: in A_ϕ, the location $l_1 \in L$ is connected by an edge to the location $l_2 \in L$, i.e. $(l_1, l_2, s, \psi) \in E$ iff the following requirements are satisfied:

(E1) For every $\bigcirc\rho \in \overline{\mathsf{Cl}}(\phi)$: $\bigcirc\rho \in l_1$ iff $\rho \in l_2$;

(E2) For every $\ominus\rho \in \overline{\mathsf{Cl}}(\phi)$: $\rho \in l_1$ iff $\ominus\rho \in l_2$;

(E3) $s = \{p_\rho \mid \rho \in l_1\}$ (propositional edge labelling function);

(E4) $\psi = \bigwedge\{x_{p_\rho} \sim c| \lhd_{\sim c} \rho \in l_1\} \bigwedge\{y_{p_\rho} \sim c| \rhd_{\sim c} \rho \in l_1\} \bigwedge\{\neg(x_{p_\rho} \sim c)|\neg(\lhd_{\sim c} \rho) \in l_1\} \bigwedge\{\neg(y_{p_\rho} \sim c)|\neg(\rhd_{\sim c} \rho) \in l_1\}$ (real-time edge labelling function).

At this stage we have only defined necessary conditions for the formula automaton A_ϕ to accept timed traces that are timed Hintikka sequences ϕ. We still have to ensure the fulfillment of fatalities. Let us examine how to cope with the fulfillment of fatalities induced by a formula of the form $\rho_1 U \rho_2$. The semantics of the formula $\rho_1 U \rho_2$ expresses that the formula ρ_1 must stay true until a ρ_2 state is eventually reached. In our case, p_{ρ_2} is a fatality in the sense that in all timed Hintikka sequences, a $p_{\rho_1 U \rho_2}$ observation is always followed by some p_{ρ_2} observation. The fulfillment of fatalities can be ensured by the mechanism of acceptance of Büchi automata and relies on the following lemma adapted from [MP95]:

Lemma 1 *Let θ be a timed Hintikka sequence of the* EventClockTA *formula ϕ and $p_{\rho_1 U \rho_2}$ a proposition promising p_{ρ_2}. Then, θ contains infinitely many positions $j \geq 0$ such that:*

$$p_{\neg(\rho_1 U \rho_2)} \in \sigma_j \textit{ or } p_{\rho_2} \in \sigma_j$$

Proof. Let us first make the hypothesis that θ contains infinitely many $p_{\rho_1 U \rho_2}$-positions. By requirement (H6) of timed Hintikka sequences, each of those positions is followed by a p_{ρ_2}-position and thus there also exists an infinite number of p_{ρ_2}-positions.
If we make the hypothesis that θ contains only finitely many $p_{\rho_1 U \rho_2}$-positions then by requirement (H2) there are infinitely many positions j s.t. $p_{\neg(\rho_1 U \rho_2)} \in \sigma_j$ and thus the theorem is verified. $\Box$

We say that a computation of A_ϕ fulfills the fatalities of a formula ϕ iff for every formula $\rho \in \overline{\mathsf{Cl}}(\phi)$ promising a formula ρ_2, the computation contains infinitely many $\neg\rho$ locations or ρ_2 locations. To restrict the accepted computation of A_ϕ to computations that fulfill the fatalities of ϕ, we use the mechanism of accepting sets.

Accepting sets: $\mathcal{F} = \{\{l|\neg(\rho_1 U \rho_2) \in l \text{ or } \rho_2 \in l\}\,|\rho_1 U \rho_2 \in \overline{\mathsf{Cl}}(\phi)\}$. The accepting sets are chosen to ensure the fatalities.

This definition completes the procedure for constructing the automaton A_ϕ. Now, let us prove that our construction is correct:

Theorem 2 *The set of timed traces accepted by the* $\mathsf{EventClockTA}$ A_ϕ *is exactly the set of timed Hintikka sequences of formula* ϕ.

Proof. First, let us show that if θ is a timed Hintikka sequence of the $\mathsf{rEventClockTL}$ formula ϕ then $\theta \in L(A_\phi)$. Let us construct a computation $\gamma = l_0 \xrightarrow{e_0} l_1 \xrightarrow{e_1} \ldots$ of A_ϕ on θ. Take $l_i = \{\rho | p_\rho \in \sigma_i\}$ for each $i \geq 0$. Let us first note that by requirement (H2) and (H3) of the definition of timed Hintikka sequences, each l_i is propositionally consistent and complete. Thus each l_i is an atom and by (L) a location of A_ϕ;

(C1) γ respects the initiality requirement of computation: by requirement (H1) of the definition of timed Hintikka sequence, $p_\phi \in \sigma_0$ and thus $\phi \in l_0$, further, by (H5), we know that for all $\ominus\rho \in \overline{\mathsf{Cl}}(\phi)$, $p_{\ominus\rho} \notin \sigma_0$ and thus $\neg \ominus \rho \in l_0$. Thus the conditions (S1) and (S2) are verified and $l_0 \in L_0$;

(C2) γ respects the consecution requirement: by points (H4) and (H5) of the definition of timed Hintikka sequences, we know that $p_{\bigcirc\rho} \in \sigma_i$ iff $p_\rho \in \sigma_{i+1}$ which transposes to γ as $\bigcirc\rho \in l_i$ iff $\rho \in l_{i+1}$. A similar reasoning can be applied to the past ($\ominus$-operators) and thus the consecution requirement is respected;

(C3) γ respects the timing requirement: if the constraint $y_{p_\rho} \sim c$ appears in the conjunction ψ_i at position i of γ, we must show that $\mathsf{Val}_{y_{p_\rho}}(\bar{\sigma}, \bar{\tau}, i) \sim c$. If $y_{p_\rho} \sim c$ appears on ψ_i then $\rhd_{\sim c}\, \rho \in l_i$ and by definition of the labelling function, we have $p_{\rhd_{\sim c}\rho} \in \sigma_i$. By (H8), we have that there exists a position $j > i$ in $\bar{\sigma}$ such that $p_\rho \in \sigma_j$, $\tau_j - \tau_i \sim c$ and for all k, $i < k < j$, $p_\rho \notin \sigma_k$. This is exactly what we wanted. A similar reasoning apply to other cases.

(C4) γ is *adequate*: direct consequence of the definition of the labelling function (E3);

(C5) γ respects the acceptance condition: by (H4), every observation σ_i s.t. $p_{\rho_1 U \rho_2} \in \sigma_i$, is followed by an observation σ_j ($j \geq i$) s.t. $p_{\rho_2} \in \sigma_j$. By construction of γ and the edge labelling function, we have that every location l_i s.t. $\rho_1 U \rho_2 \in l_i$ is followed by a location l_j ($j \geq i$) s.t. $\rho_2 \in l_j$. Thus for every formula $\rho_1 U \rho_2 \in \overline{\mathsf{Cl}}(\phi)$, γ contains infinitely many locations that either contain ρ_2 (if there are infinitely many locations that contain $\rho_1 U \rho_2$) or there are infinitely many locations that do not contain $\rho_1 U \rho_2$ and thus the generalized Büchi condition is verified by γ.

Second, we show that if $\theta \in L(A_\phi)$ then θ is a timed Hintikka sequence of ϕ. To prove that direction, we show that for all positions i in θ every condition of the definition of timed Hintikka sequences is verified. We assume $\gamma = l_0 \xrightarrow{e_0} l_1 \xrightarrow{e_1} \dots$ of A_ϕ on θ as above.

(H1) As γ is a computation of A_ϕ, γ respects the initiality requirement (C1), i.e. $l_0 \in L_0$, and thus by (S1), $\phi \in l_0$. The propositional labelling function of A_ϕ is adequate (C4), so that $p_\phi \in \sigma_0$ and thus requirement (H1) is verified.

(H2) Let us first show that if $p_\rho \in \sigma_i$ where $\rho = \neg\rho_1$, then $p_{\rho_1} \notin \sigma_i$. By the definition of the edge labelling function (E3), adequacy requirement of computation (C4), we know that $\neg\rho_1 \in l_i$. By requirement (A1) of atoms, $\rho_1 \notin l_i$ which implies by definition of the propositional edge labelling function of A_ϕ (E3) and by the adequacy requirement of computation (C4) that $p_{\rho_1} \notin \sigma_i$. If $p_\rho \in \sigma_i$ where $\rho = \rho_1$, then $p_{\neg\rho_1} \notin \sigma_i$ is established by a similar reasoning. Requirement (H2) is thus verified.

(H3) Let $p_\rho \in \sigma_i$ where $\rho = \rho_1 \vee \rho_2$. By construction of γ, this means $\rho_1 \vee \rho_2 \in l_i$. By requirement (A2) of atoms, this means either $\rho_1 \in l_i$ or $\rho_2 \in l_i$. By definition of the propositional labelling function (E3), this means either $p_{\rho_1} \in s_i$ or $p_{\rho_2} \in s_i$ where s_i is the propositional labelling of edge e_i. Thus $p_{\rho_1} \in \sigma_i$ or $p_{\rho_2} \in \sigma_i$.

(H4) We have to show that $p_{\bigcirc\rho_1} \in \sigma_i$ iff $p_{\rho_1} \in \sigma_{i+1}$. $p_{\bigcirc\rho_1} \in \sigma_i$, by (E3) and (C4), means that $\bigcirc\rho_1 \in l_i$. By the consecution requirement (C2) and (E1), this is $\rho_1 \in l_{i+1}$. Finally, by (E3) and (C4), we obtain $p_{\rho_1} \in \sigma_{i+1}$. The other direction is similar.

(H5) This case is similar to the previous one and is left to the reader.

(H6) We first show that if $p_\rho \in \sigma_i$ where $\rho = \rho_1 U \rho_2$ then there exists a position j, $j \geq i$ s.t. $p_{\rho_2} \in \sigma_j$ and for all positions k, $i \leq k < j$, $p_{\rho_1} \in \sigma_k$. First, we show that j exists. By contradiction, consider the hypothesis that there does not exists a first l_j s.t. $\rho_2 \in l_j$ with $i \leq j$. But in that case, we have $\neg\rho_2 \in l_k$ for all $i \leq k$. As $\rho_1 U \rho_2 \in l_i$, a inductive reasoning similar to the next one allows us to conclude that $\rho_1 U \rho_2 \in l_k$ for all $i \leq k$. Thus, γ would not be accepting, contradicting the definition of γ. So we can take the first such j. By definition of γ, (E3) and (C4), $\rho_1 U \rho_2 \in l_i$. We note l_j with $j \geq i$, the first location after l_i in γ such that $\rho_2 \in l_j$. Since we have taken the first j, for all k, $i \leq k < j$, $\rho_2 \notin l_k$. Let us show that $\rho_1 \in l_k$ and $\rho_1 U \rho_2 \in l_{k+1}$for each of those k. We reason by induction:

- *Base case*: $k = i < j$, as $\rho_1 U \rho_2 \in l_i$ and $\rho_2 \notin l_i$, the requirement (A3) of atoms allow us to conclude that $\rho_1 \in l_i$ and $\rho_1 U \rho_2 \in l_{i+1}$.

– *Induction case*: by inductive hypothesis we have $\rho_1 \in l_l$ and $\rho_1 U \rho_2 \in l_{l+1}$, for all l s.t. $i \leq l < k < j$, let us show that we have that $\rho_1 \in l_k$ and $\rho_1 U \rho_2 \in l_{k+1}$. As $\rho_1 U \rho_2 \in l_k$ and $\rho_2 \notin \sigma_k$ as $k < j$, the connection requirement (E1) and the requirement (A3) of atoms allow us to conclude $\rho_1 \in l_k$ and $\rho_1 U \rho_2 \in l_{k+1}$.

So we have shown that $\rho_1 \in l_k$ for all k s.t. $i \leq k < j$ and by hypothesis $\rho_2 \in l_j$. By the definition of the propositional edge labelling function (E3) and adequacy (C4), $p_{\rho_1} \in \sigma_k$ for all k s.t. $i \leq k < j$ and $p_{\rho_2} \in \sigma_j$.

We consider now the other direction: let us make the hypothesis that there exists $j \leq i$ s.t. $p_{\rho_2} \in \sigma_j$ and for all k, $i \leq k < j$, $p_{\rho_1} \in \sigma_k$ then we must establish that $p_{\rho_1 U \rho_2} \in \sigma_i$. Again, we can use the first such j. From that, let us show that for all k, $i \leq k \leq j$, $p_{\rho_1 U \rho_2} \in \sigma_k$. We reason by induction.

– *Base case*: $k = j$. As $\rho_2 \in l_j$, we have, by (E3), (C4) and requirement (A3) of atoms, we have $\rho_1 U \rho_2 \in l_j$ and thus $p_{\rho_1 U \rho_2} \in \sigma_j$.
– *Induction case*: by induction hypothesis, we have that for all m, $i < k \leq m \leq j$, $p_{\rho_1 U \rho_2} \in \sigma_m$. Let us show that $p_{\rho_1 U \rho_2} \in \sigma_{k-1}$. We know that $p_{\rho_1 U \rho_2} \in \sigma_k$, $p_{\rho_1} \in \sigma_{k-1}$ and $p_{\rho_2} \notin \sigma_k$. By (E3) and (C4), we have $\bigcirc(\rho_1 U \rho_2), \rho_1 \in l_{k-1}$. By requirement (A3) of atoms, we obtain $\rho_1 U \rho_2 \in l_{k-1}$ and thus $p_{\rho_1 U \rho_2} \in \sigma_{k-1}$ by (E3) and (C4).

(H7) This case is similar to the previous one and is left to the reader.

(H8) First let us prove that if $p_\rho \in \sigma_i$ where $\rho = \triangleright_{\sim c} \rho_1$ then there exists a position j such that $j > i$ and $p_{\rho_1} \in \sigma_j$, $\tau_j - \tau_i \sim c$ and for all k, $i < k < j$, $p_{\rho_1} \notin \sigma_k$. By definition of γ and (E3), we know that $\triangleright_{\sim c} \rho_1 \in l_i$. By definition of the real-time edge labelling function, we know that ψ_i is of the form $y_{p_{\rho_1}} \sim c \wedge \psi'_i$ and thus by the timing requirement of computation (C3): $(\theta, i) \models y_{p_{\rho_1}} \sim c$ which implies exactly what we had to prove.

Now let us show that if there exists a position j such that $j > i$ and $p_{\rho_1} \in \sigma_j$, $\tau_j - \tau_i \sim c$ and for all k, $i < k < j$, $p_{\rho_1} \notin \sigma_k$ then $p_{\triangleright_{\sim c} \rho_1} \in \sigma_i$. Let us make the hypothesis that $p_{\triangleright_{\sim c} \rho_1} \notin \sigma_i$. Then $\triangleright_{\sim c} \rho_1 \notin l_i$ and thus, by atom propositional completeness (A1), $\neg \triangleright_{\sim c} \rho_1 \in l_i$. By the real-time labelling function, $\psi_i = \psi'_i \wedge \neg (y_{p_{\rho_1}} \sim c)$ which by the semantics of prophecy clock constraint contradicts that γ is a computation of A_ϕ. Thus $p_{\triangleright_{\sim c} \rho_1} \in \sigma_i$.

(H9) This case is similar to the previous one and is left to the reader.

□

Corollary 1 *The* rEventClockTL *formula* ϕ *is satisfiable iff the language accepted by the* EventClockTA A_ϕ *is not empty.*

Proof. Direct consequence of proposition 1 and theorem 2. □

The usual next step is to show that we can restrict the symbols on edges of A_ϕ to propositions. However, for EventClockTA, this works only for non-recursive formulas. Let $A_\phi \cap \mathcal{P}$ be as A_ϕ, but with edges labelled with proposition symbols only: $s = \{p_\pi | \pi \in l_1 \wedge \pi \in \mathcal{P}\}$.

Corollary 2 *The models of a non-recursive* rEventClockTL *formula* ϕ *form the language accepted by the* EventClockTA $A_\phi \cap \mathcal{P}$.

To have a decision procedure for our rEventClockTL logic, it remains us to show how the emptiness of EventClockTA can be decided. The principles of the region construction [AD94] which transforms a timed automaton into an untimed finite state machine can be applied to EventClockTA automata. The idea is to construct a finite state machine that accepts $\mathsf{Untimed}(L(A_\phi))$, i.e. $\{\bar{\sigma} | (\bar{\sigma}, \bar{\tau}) \in L(A_\phi)\}$. The results presented here are adapted from [AD94, AFH94] and are recalled to allow the reader, not familiar with real-time automata, to fully understand the decision procedure.

Definition 15 An *extended state* of an EventClockTA $A = (L, L_0, \mathcal{P}, \mathbb{C}, E, \mathcal{F})$ is a pair (l, η) where $l \in L$ is a location and $\eta : \mathbb{C} \to \mathbb{R}^+ \cup \{\perp\}$, is a clock valuation which associates a value of $\mathbb{R}^+ \cup \{\perp\}$ to each clock $z \in \mathbb{C}$ of the automaton.

The following definition formalizes the effect of time passing on valuations of clocks:

Definition 16 ($\eta + t$) The clock valuation η' obtained from the clock valuation η by letting time elapse during t, denoted $\eta + t$, is defined as follows:

- For all prophecy clocks $y \in \mathbb{P}$: $(\eta + t)(y) = \eta(y) - t$ if $\eta(y) - t \geq 0$; otherwise $\eta + t$ is not defined.
- For all history clocks $x \in \mathbb{H}$: $(\eta + t)(x) = \eta(x) + t$

with the addition $+$ and subtraction $-$ interpreted as usual in the real numbers and as follows for the special value $\perp$: $\perp + t = \perp, \perp - t = \perp$.

The number of extended states is uncountable, as we model time by the nonnegative real numbers ($\mathbb{R}^+$). But to evaluate real-time constraints labelling edges of EventClockTA, only the integer value of clocks and whether their fractional part is zero is needed. Also, to know which clocks will first change their integer value, we only need to know the order between the fractional parts of the clock values. Next we recall the definition of an equivalence relation between valuations based on those two remarks. This equivalence relation partitions the valuations into a finite number of equivalence classes called *regions*. Two states in the same region will behave similarly.

Definition 17 [AD94, AFH94] Two clock valuations η_1, η_2 are *in the same region*, denoted $\eta_1 \approx \eta_2$, for an automaton $A = (L, L_0, \mathcal{P}, \mathbb{C}, E, \mathcal{F})$ iff the following conditions are respected:

- η_1 and η_2 agree on which clocks have the undefined value $\perp$. Those clocks are called *undefined*. The set of clocks undefined in valuation η is denoted $\mathsf{Undefined}(\eta)$. The other clocks are called *active*. The set of clocks active in valuation η is denoted $\mathsf{Active}(\eta)$.
- η_1 and η_2 agree on the integral part of all active clocks that are at most c, where c is the biggest constant appearing in the the real-time constraints decorating the edges of A:
  - $\forall z \in \mathsf{Active}(\eta_1)$, if $\eta_1(z) \le c$ or $\eta_2(z) \le c$ then $\lfloor \eta_1(z) \rfloor = \lfloor \eta_2(z) \rfloor$
- η_1 and η_2 agree on the ordering of the fractional part of all active clocks that are at most c:
  - for a prophecy clock y, let $\langle \eta_1(y) \rangle$ be $\eta_1(y) - \lfloor \eta_1(y) \rfloor$ and for a history variable x let $\langle \eta_1(x) \rangle$ be $\lceil \eta_1(x) \rceil - \eta_1(x)$. For all $z_1, z_2 \in \mathsf{Active}(\eta_1)$ with $\eta_1(z_1) \le c$ and $\eta_1(z_2) \le c$:
    - $\langle \eta_1(z_1) \rangle = 0$ iff $\langle \eta_2(z_1) \rangle = 0$
    - $\langle \eta_1(z_1) \rangle \le \langle \eta_1(z_2) \rangle$ iff $\langle \eta_2(z_1) \rangle \le \langle \eta_2(z_2) \rangle$

A *clock region* is an equivalence class of $\approx$. Two extended states $(l_1, \eta_1), (l_2, \eta_2)$ are region-equivalent if $l_1 = l_2$ and $\eta_1 \approx \eta_2$. Note that $\approx$ is of finite index.

Let us now define when a clock region α_2 is the time successor of another clock region α_1.

Definition 18 A clock region α_2 is a *time successor* of a clock region α_1, denoted $\alpha_2 \in \mathsf{TS}(\alpha_1)$, iff $\forall \eta_1 \in \alpha_1$, $\exists t \in \mathbb{R}^+$ such that $\eta_1 + t \in \alpha_2$.

Next, we define a Büchi automaton with ϵ-moves, called the region automaton of A, denoted $R(A)$ that accepts exactly $\mathsf{Untimed}(L(A))$. The ϵ-moves will be used to model time passing, i.e. transitions between clock regions.

Definition 19 The region automaton of $A = (L, L_0, \mathcal{P}, \mathbb{C}, E, \mathcal{F})$ is the Büchi automaton $R(A) = (L^r, L_0^r, \Sigma^r, E^r, \mathcal{F}^r)$ where:

- L^r is the set of regions, i.e. 3-tuple (l, α, ξ) with $l \in L$, α an equivalence class of clock interpretations and $\xi \in \{t, d\}$. With ξ, locations are partitioned[3];

- L^r_0 is the subset of locations $(l, \alpha, \xi) \in L$ where $l \in L_0$, $\forall x \in \mathbb{H}, \alpha(x) = \perp$, $\xi = t$. Initially all history clocks are undefined.

- $\Sigma^r = 2^{\mathcal{P}} \cup \{\epsilon\}$;

- E^r is the set of triples $((l_1, \alpha_1, \xi_1), (l_2, \alpha_2, \xi_2), s)$ such that

  - if $s \in 2^{\mathcal{P}}$, $\xi_1 = t$ and $\xi_2 = d$ meaning that the last transition of the automaton was a time transition and now the automaton takes a discrete transition, and there is an edge (l_1, l_2, s, ψ) in automaton A and a clock region α_3 such that:

    * $\alpha_1 = \alpha_3[y_p := 0 | p \in s]$ (α_1 agrees with α_3 on all clocks except prophecy clocks associated with propositions that appear in s; those clocks have the value 0 in α_1);
    * $\alpha_2 = \alpha_3[x_p := 0 | p \in s]$ (α_2 agrees with α_3 on all clocks except history clocks associated with propositions that appear in s; those clocks have the value 0 in α_2);
    * $\forall \eta \in \alpha_3, \eta \models \psi$: the value of clocks when crossing the edge are consistent with the real-time constraint ψ.

  - if $s = \epsilon$, $\xi_1 = d$ and $\xi_2 = t$ meaning that the last transition of the automaton was a discrete transition, and now the move is a time move: $\alpha_2 \in \mathsf{TS}(\alpha_1)$ (the region α_2 is a time successor of the region α_1) and $l_1 = l_2$;

- $\mathcal{F}^r = \{F'_1, \ldots, F'_n\} \cup \{F_{x_{p_\rho}} | \lhd_{\sim c}\ \rho \in \overline{\mathsf{Cl}}(\phi)\} \cup \{F_{y_{p_\rho}} | \rhd_{\sim c}\ \rho \in \overline{\mathsf{Cl}}(\phi)\}$; where:

  - for all i, $F'_i = \{(l, \alpha, \xi) | l \in F_i\}$. So each F'_i is a set of regions composed of an accepting location for F_i of A and a clock region α;
  - $F_{x_{p_\rho}} = \{(l, \alpha, \xi) | \eta(x_{p_\rho}) = 0 \vee \eta(x_{p_\rho}) > c \vee \eta(x_{p_\rho}) = \perp, \forall \eta \in \alpha\}$ is the set of regions where the history clock x_{p_ρ} is greater than the maximal constant c, equal to zero or undefined. This ensures that either x_{p_ρ} is reset infinitely often, always undefined or its value goes beyond any bounds. This is imposed by the progress of time requirement of timed traces and the semantics of history clocks.

[3]This partition of the locations allows us to force the region automaton to take infinitely many discrete jumps corresponding to the infinitely many observations of a trace.

– $F_{y_{p_\rho}} = \{(l, \alpha, \xi) | \eta(y_{p_\rho}) = 0 \vee \eta(y_{p_\rho}) = \bot, \forall \eta \in \alpha\}$ is the set of regions where the prophecy clock y_{p_ρ} has the value 0 or is undefined. These sets are necessary to ensure the progress of time. In fact, if a prophecy clock is not undefined, as time always progresses, the clock must inevitably attain the value 0.

The language of $R(A)$ is the set of infinite traces corresponding to accepted runs of $R(A)$. The following theorem states the correctness of the region automaton.

Theorem 3 *[AFH94] The language of $R(A)$ is* $\mathsf{Untimed}(L(A))$.

Corollary 3 *The timed language of A is empty iff the language of $R(A)$ is empty.*

The theorem 2 and corollary 3 give us the possibility to decide the model-checking as well as the satisfiability/validity problems for rEventClockTL.

Theorem 4 *The satisfiability and validity problems for* rEventClockTL *are decidable.*

Proof. The satisfiability of an rEventClockTL formula ϕ can be decided by constructing A_ϕ, the automaton for ϕ and testing if $L(A_\phi) \neq \emptyset$. Similarly the validity of an an rEventClockTL formula ϕ can be decided by constructing $A_{\neg\phi}$, the automaton for the negation of ϕ and testing if $L(A_{\neg\phi}) = \emptyset$. □

The model-checking problem for real-time reactive systems consists in verifying that the timed traces defined by a product of timed automata respect a property expressed in a real-time logic, i.e. $L(A_1 \times \cdots \times A_n) \subseteq L(\phi)$. Note that $L(A_1 \times \cdots \times A_n) \subseteq L(\phi)$ iff $L(A_1 \times \cdots \times A_n) \cap L(\neg\phi) = \emptyset$ iff $L(A_1 \times \cdots \times A_n \times A_{\neg\phi}) = \emptyset$. This gives us a decision procedure for the model-checking problem: compute $A_{\neg\phi}$, the automaton for the negation of ϕ, test if the product of this automaton with the timed automata has a empty timed language. This gives the following theorem.

Theorem 5 *The real-time model checking problem for* rEventClockTL *is decidable.*

The procedure that we propose for deciding rEventClockTL constructs first an EventClockTA which is transformed into an untimed automaton, the region automaton, for checking emptiness. The following lemma and theorem characterize the size of the constructed automata for a given rEventClockTL formula ϕ:

Theorem 6 *[AFH94] The region equivalence $\approx$ defined on the extended states of an* EventClockTA *A is of finite index. The number of locations in* *region automaton of an* EventClockTA A *is* $O(l \cdot 2^{m \cdot \log c \cdot m})$, *where* l *is the number of locations in* A, m *is the number of clocks in* A *and* c *is the largest constant appearing in* A.

The emptiness of the region automaton can be tested without constructing it completely:

Lemma 2 *[SVW85] The nonemptiness problem for Büchi automata is* NLogSpace-Complete.

¿From theorem 6 and lemma 2 we obtain:

Lemma 3 (PSpace-Easiness) *The satisfiability and validity problem for* rEventClockTL *in pointwise semantics are* PSpace-*Easy.*

Proof. First, the size for each formula $\phi \in$ rEventClockTL is defined by the three following elements:

1. the number of subformulas in ϕ (bounded by $|\phi|$);
2. the maximal integer constant K used in a real-time operator within ϕ (bounded by $2^{|\phi|}$);
3. the number of real-time subformulas in ϕ (bounded by $|\phi|$).

By observing how A_ϕ is constructed, it is direct to show that its size is as follows:

- the number of locations in A_ϕ is exponential in the number of subformulas in ϕ;
- the maximal integer constant used by A_ϕ in clock constraints is equal to the maximal integer constant K used by ϕ within real-time operators;
- the number of clocks used by A_ϕ is bounded by the number of real-time subformulas in ϕ.

By lemma 6, we can construct the region automaton R^{A_ϕ} which is a Büchi automaton with a number of locations:

- linear in the number of locations of A_ϕ, and thus singly exponential in the number of subformulas of ϕ;
- singly exponential in the number of clocks used by A_ϕ and thus singly exponential in the number of real-time operators of ϕ;
- singly exponential in the maximal constant used by A_ϕ and thus singly exponential in the maximal constant K used in ϕ.

Using a nondeterministic version for the emptiness of R^{A_ϕ}, this exponential automaton needs not be constructed explicitly and we obtain a PSPACE procedure for the satisfiability and validity problems of rEventClockTL. □

Lemma 4 (PSPACE-Hardness) *The satisfiability and validity problems for* rEventClockTL *in pointwise semantics are* PSPACE-*Hard.*

Proof. The hardness follows directly from the fact that the logic LTL is contained in rEventClockTL and has been shown PSPACE-hard in [CES86]. □

As a consequence, the complexity of the satisfiability problem and the validity problem of rEventClockTL are in PSPACE.

Theorem 7 *The satisfiability and validity problems for* rEventClockTL *are* PSPACE-*complete.*

6 Expressiveness

In this section, we study the expressive power of rEventClockTL in pointwise timed traces. The results differ when the logic is evaluated in continuous timed traces (i.e. timed state sequences), see Appendix. First, we compare its expressive power with respect to MetricIntervalTL; then, with respect to EventClockTA.

6.1 rEventClockTL vs MetricIntervalTL

In this subsection, we compare the expressiveness of the logic rEventClockTL with the expressiveness of the logic MetricIntervalTL. We first recall the definition of the syntax and the semantics of the logic MetricIntervalTL.

Definition 20 (MetricIntervalTL-syntax) A formula of MetricIntervalTL is built from proposition symbols, boolean connectives, and time-bounded "until" and "since" operators:

$$\phi \ ::= \ p \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \phi_1 \widehat{U}_I \, \phi_2 \mid \phi_1 \widehat{S}_I \, \phi_2$$

where p is a proposition and I is a *nonsingular* interval whose finite endpoints are nonnegative integers, and that does not contain 0. □

In the sequel, we will be interested in fragments of MetricIntervalTL:

Definition 21 (MetricIntervalTL$_{0,\infty}$-fragment) The formulas of the fragment MetricIntervalTL$_{0,\infty}$ are defined as above, except that the interval I must either have the left endpoint 0, or be unbounded; in these cases I can be replaced by an expression of the form $\sim c$, for a nonnegative integer constant c and $\sim \in \{<, \leq, \geq, >\}$. □

Definition 22 (MetricIntervalTL^F-fragment) The formulas of the fragment MetricIntervalTL^F are defined as for MetricIntervalTL, except that $\widehat{U}_I$ is the only real-time operator.

We now define the semantics of those logics.

Definition 23 The MetricIntervalTL formula ϕ *holds* in position $i \in \mathbb{N}$ of the timed trace $\theta = (\bar{\sigma}, \bar{\tau})$, denoted $(\theta, i) \models \phi$, according to the following definition:

$(\theta, i) \models p$ iff $p \in \sigma_i$;
$(\theta, i) \models \neg\phi$ iff $(\theta, i) \not\models \phi$;
$(\theta, i) \models \phi_1 \wedge \phi_2$ iff $(\theta, i) \models \phi_1$ and $(\theta, i) \models \phi_2$;
$(\theta, i) \models \phi_1 \widehat{U}_I \phi_2$ iff there exists $j > i$ such that $(\theta, j) \models \phi_2$, $\tau_j - \tau_i \in I$ and for all k with $i < k < j$, we have $(\theta, k) \models \phi_1$ [4];
$(\theta, i) \models \phi_1 \widehat{S}_I \phi_2$ iff there exists j, $0 \leq j < i$, such that $(\theta, j) \models \phi_2$, $\tau_j - \tau_i \in I$ and for all k with $j < k < i$, we have $(\theta, k) \models \phi_1$;

The MetricIntervalTL formula ϕ *defines* the timed ω-language $L(\phi)$ that contains all timed state sequences θ with $(\theta, 0) \models \phi$.

We also use the following classical abbreviations:

- When the real-time constraint is omitted, it is the most permissive: $\phi \widehat{U} \psi \equiv \psi \vee \phi \widehat{U}_{(0,\infty)} \psi$;
- We can use constraints instead of intervals: $\phi \widehat{U}_{\sim c} \psi \equiv \psi \vee \phi \widehat{U}_I \psi$, where $I = \{r \in \mathbb{R} | r > 0 \wedge r \sim c\}$;
- We can extend intervals to include 0, by making $\widehat{U}$ reflexive: $\phi \widehat{U}_{\{0\} \cup I} \psi \equiv \psi \vee \phi \widehat{U}_I \psi$;
- $\widehat{\Diamond}_I \phi \equiv \top \widehat{U}_I \phi$, meaning "eventually within I";
- $\widehat{\Box}_I \phi \equiv \neg \widehat{\Diamond}_I \neg \phi$, meaning "always during I";
- and their past counterparts: $\widehat{\diamondminus}_I \phi \equiv \top \widehat{S}_I \phi$, $\widehat{\boxminus}_I \phi \equiv \neg \widehat{\diamondminus}_I \neg \phi$;
- In lemma 7 we will see that all rEventClockTL operators can be defined as abbreviations.

We now compare the expressive power of the two logics. We will show that they differ on the infinite set of timed traces Θ:

Definition 24 The (infinite) set of timed traces $\Theta = \{\theta^\infty, \theta^1, \theta^2, \ldots, \theta^n, \ldots\}$ contains the following traces defined on the set of propositions $\mathcal{P} = \{p\}$:

[4]Note that the operator $\widehat{U}_I$ is irreflexive.

1. each $\theta^k \in \Theta$ contains the same qualitative information: $\theta^k = (\bar{\sigma}, \bar{\tau}^k)$ is such that for all position $i \in \mathbb{N}$, $\sigma_i = \{p\}$; that is, p is true in every position of every timed trace of Θ.

2. the timed traces of Θ have the following timing information:

   (a) for θ^∞, the timing information $\bar{\tau}^\infty = \tau_0^\infty \tau_1^\infty \ldots \tau_n^\infty \ldots$ is $\tau_i^\infty = i \times 1.5$; that is, an observation each 1.5 time units;

   (b) for θ^k, with $k \in \{1, 2, \ldots, n, \ldots\}$, the timing information $\bar{\tau}^k = \tau_0^k \tau_1^k \ldots \tau_n^k \ldots$ is

   $$\tau_i^k = \begin{cases} i \times 1.5 & \text{if } i \neq k \\ i \times 1.5 - 0.1 & \text{if } i = k \end{cases}$$

   that is, in θ^k, there is an irregular k^{th} observation which is separated from the $k - 1^{th}$ by 1.4 time units and from the $k + 1^{th}$ by 1.6 time units.

   Let us note that for every position $i \in \mathbb{N}$, in a timed trace $\theta = (\bar{\sigma}, \bar{\tau}) \in \Theta$, $\tau_{i+1} - \tau_i \in (1, 2)$. That is, the time difference between two consecutive observations is between 1 and 2 time units, in fact it is either equal to 1.4, 1.5 or 1.6.

□

Example 4 Here are two examples of prefixes of traces from the set Θ:

- a prefix of θ^∞:

  $$(\{p\}, 0)(\{p\}, 1.5)(\{p\}, 3)(\{p\}, 4.5)(\{p\}, 6)(\{p\}, 7.5)\ldots$$

- a prefix of θ^3:

  $$(\{p\}, 0)(\{p\}, 1.5)(\{p\}, 3)(\{p\}, 4.4)(\{p\}, 6)(\{p\}, 7.5)\ldots$$

  so the observation number 3 is at 4.4 instead of 4.5 as it is in θ^∞.

In the next lemma, we show that the future fragment of MetricIntervalTL can distinguish θ^∞ from the other timed traces of Θ. The idea is that the position i is always separated for the position $i + 2$ by 3 time units in θ^∞ while it is not the case in θ^k, where the k^{th} position is separated by 3.1 time units from the position $k + 2$. We now show that a simple MetricIntervalTL^F formula can detect this fact.

Lemma 5 *$\psi \equiv \widehat{\Box}_{(0,\infty)}(p \rightarrow \widehat{\Diamond}_{[2,3]} p) \in$ MetricIntervalTL^F is such that $(\theta^\infty, 0) \models \psi$ and for all $k \geq 1$, $(\theta^k, 0) \not\models \psi$.*

Proof. Every position i in θ^∞ is separated by exactly 3 time units from the position $i+2$. As p is true everywhere, $p \rightarrow \widehat{\Diamond}_{[2,3]}p$ holds in every position of θ^∞ and thus by the semantics of the $\widehat{\Box}_{(0,\infty)}$-operator, $(\theta^\infty, 0) \models \psi$. On the other hand, the k^{th} position of θ^k is not followed by any position in $\tau_k^k + [2,3]$ as the $(k+1)^{th}$ position is at time $\tau_k^k + 1.6$ and the $(k+2)^{th}$ position is at time $\tau_k^k + 3.1$. Thus $p \rightarrow \widehat{\Diamond}_{[2,3]}p$ is false in position k of Θ^k and thus $(\theta^k, 0) \not\models \psi$. □

We now show that the future fragment of rEventClockTL cannot distinguish between timed traces of Θ. This is a consequence of the following stronger lemma:

Lemma 6 *For every formula $\phi \in$ rEventClockTL^F, for every two timed traces $\theta_1, \theta_2 \in \Theta$, for every two positions i, j such that $0 \leq i < j$: $(\theta_1, i) \models \phi$ iff $(\theta_2, i) \models \phi$ iff $(\theta_1, j) \models \phi$ iff $(\theta_2, j) \models \phi$. That is, every formula of* rEventClockTL *is either constantly true in all timed traces of Θ or constantly false in all timed traces of Θ.*

Proof. The proof is by induction on the structure of formula.

- $\phi = p$: as p is true in every position of every timed trace of Θ, the base case is verified.
- $\phi = \rhd_{\sim c} \phi_1$: By induction hypothesis, we know that either:
  1. ϕ_1 is true in all positions of all timed traces of Θ: Thus for every position i, the first following ϕ_1 is in $i+1$ and by definition of Θ, $\tau_{i+1} - \tau_i \in (1,2)$ in the two timed traces. Thus $\rhd_{\sim c} \phi_1$ is constantly true if $(1,2) \subseteq \{v \in \mathbb{R}^+ \mid v \sim c\}$, and constantly false otherwise.
  2. ϕ_1 is false in all positions of all timed traces of Θ: So there does not exists a (first) following ϕ_1 position, and $\rhd_{\sim c} \phi_1$ is constantly false.
- The other cases are left to the reader.

□

A direct consequence of the lemma 6 is that rEventClockTL^F cannot distinguish θ^∞ from other models of Θ:

Corollary 4 *For every formula $\psi \in$ rEventClockTL^F, for every $k \geq 1$, $\theta^\infty \in L(\psi)$ iff $\theta^k \in L(\psi)$.*

And rEventClockTL^F is less expressive than MetricIntervalTL^F:

Theorem 8 MetricIntervalTL$^F \not\subseteq$ rEventClockTL^F.

Proof. By corollary 4, we know that for every formula $\phi \in \mathsf{rEventClockTL}^F$, $L(\phi)$ contains Θ or has an empty intersection with Θ. On the other hand, the formula $\psi \equiv \widehat{\Box}(p \to \widehat{\Diamond}_{[2,3]}p)$ of $\mathsf{MetricIntervalTL}^F$ is satisfied by θ^∞ but by none of the timed traces $\theta^k \in \Theta$. It means that $L(\psi) \cap \Theta \neq \emptyset$ but $\Theta \not\subseteq L(\psi)$. Thus $\mathsf{rEventClockTL}^F$ cannot express the property expressed by ψ. □

Let us now take a look at the other direction of the inclusion: "is every $\mathsf{rEventClockTL}^F$-expressible property also expressible in $\mathsf{MetricIntervalTL}^F$?" To answer this question, we provide a translation ϕ^T, defined by induction:

- $\phi = p$: $\phi^T = p$.
- $\phi = \phi_1 \vee \phi_2$: $\phi^T = \phi_1^T \vee \phi_2^T$.
- $\phi = \neg\phi_1$: $\phi^T = \neg\phi_1^T$.
- $\phi = \bigcirc\phi_1$: $\phi^T = \bot\widehat{U}_{(0,\infty)}\phi_1^T$.
- $\phi = \ominus\phi_1$: $\phi^T = \bot\widehat{S}_{(0,\infty)}\phi_1^T$.
- $\phi = \phi_1 U \phi_2$: $\phi^T = \phi_2^T \vee (\phi_1^T \wedge (\phi_1^T\widehat{U}_{(0,\infty)}\phi_2^T))$.
- $\phi = \phi_1 S \phi_2$: Symmetrically, $\phi^T = \phi_2^T \vee (\phi_1^T \wedge (\phi_1^T\widehat{S}_{(0,\infty)}\phi_2^T))$.
- $\phi = \rhd_I \phi_1$: $\phi^T = \top\widehat{U}_{\downarrow I}\phi_1^T \wedge \neg(\top\widehat{U}_{<I}\phi_1^T)$, where $\downarrow I$ is the real interval $\{t > 0 \mid \exists t' \in I : t \leq t'\}$ and $< I$ is the real interval $\{t > 0 \mid \forall t' \in I : t < t'\}$.
- $\phi = \lhd_I \phi_1$: Symmetrically, $\phi^T = \top\widehat{S}_{\downarrow I}\phi_1^T \wedge \neg(\top\widehat{S}_{<I}\phi_1^T)$.

Lemma 7 *For every formula ϕ of* $\mathsf{rEventClockTL}$*, $\phi^T \in \mathsf{MetricIntervalTL}_{0,\infty}$ has the same meaning: for every timed trace θ, for every position i: $(\theta, i) \models \phi$ iff $(\theta, i) \models \phi^T$. Furthermore, this translation respects future fragments: if $\phi \in \mathsf{rEventClockTL}^F$, $\phi^T \in \mathsf{MetricIntervalTL}^F_{0,\infty}$.*

As a consequence we have the following theorem:

Theorem 9 *The logic* $\mathsf{MetricIntervalTL}_{0,\infty}$ *is at least as expressive as* $\mathsf{rEventClockTL}$*,* $\mathsf{rEventClockTL} \subseteq \mathsf{MetricIntervalTL}_{0,\infty}$ *and thus* $\mathsf{rEventClockTL} \subseteq \mathsf{MetricIntervalTL}$.

In the theorem 8, we have shown that the inclusion $\mathsf{rEventClockTL}^F \subset \mathsf{MetricIntervalTL}^F$ is strict. Is this inclusion also strict for the full $\mathsf{rEventClockTL}$ logic? Before answering this question, let us first note that adding past operators to $\mathsf{rEventClockTL}^F$ adds expressive power. In fact, let us consider the timed trace θ^k, with k even (so that $k \times 1.5$ is a natural number), and the $\mathsf{rEventClockTL}$ formula

$$\psi^k \equiv \underbrace{\bigcirc \ldots \bigcirc}_{k} \lhd_{=k\times 1.5} \neg \ominus \top$$

where $\neg \ominus \top$ is only true at the initial position of θ^k and thus ψ^k expresses, in this initial position, that "the k^{th} position of θ^k has the timing $k \times 1.5$". Which is false by definition of θ^k. On the other hand, this property is true in the initial position of θ^∞ and thus the formula ψ^k can distinguish between θ^k and θ^∞. Thus, adding past operators to $\mathsf{rEventClockTL}^F$ increases the expressive power of the logic:

Theorem 10 *The logic* $\mathsf{rEventClockTL}$ *is strictly more expressive than its future fragment* $\mathsf{rEventClockTL}^F$*:* $\mathsf{rEventClockTL}^F \subset \mathsf{rEventClockTL}$.

Note that this phenomenon is not observed in the temporal logic LTL: adding past operators to LTL only adds convenience but no real expressive power [GPSS80]. For real-time logics, in contrast, past operators add expressive power, for instance [AH92a] proved that $\mathsf{MetricIntervalTL}^F \subset \mathsf{MetricIntervalTL}$, noted there $MITL \subset MITL^P$.

The formula ψ^k above explains why our simple proof that $\mathsf{MetricIntervalTL}^F$ is more expressive than $\mathsf{rEventClockTL}^F$ will not work to show that $\mathsf{MetricIntervalTL}$ is more expressive than $\mathsf{rEventClockTL}$. But this formula does not distinguish θ^∞ from θ^l with $l > k$. For such a l, intuitively, we need a bigger $\mathsf{rEventClockTL}$ formula, such as ψ^l. In the next lemma, we prove that for any given formula ϕ of $\mathsf{rEventClockTL}$, there exists a bound $\mathsf{size}(\phi)$ such that the formula ϕ cannot distinguish between θ^∞ and θ^k for $k > \mathsf{size}(\phi)$. This size, intuitively, measures how far ϕ can look into the past of θ^∞. Formally:

Definition 25 (Size of an $\mathsf{rEventClockTL}$-formula) The size of a formula $\phi \in \mathsf{rEventClockTL}$, denoted $\mathsf{size}(\phi)$, is defined recursively as follows:

- $\mathsf{size}(p) = 0$;
- $\mathsf{size}(\neg\phi_1) = \mathsf{size}(\phi_1)$;
- $\mathsf{size}(\phi_1 \vee \phi_2) = \mathsf{max}(\mathsf{size}(\phi_1), \mathsf{size}(\phi_2))$;
- $\mathsf{size}(\bigcirc\phi_1) = \mathsf{size}(\phi_1)$;
- $\mathsf{size}(\ominus\phi_1) = 1 + \mathsf{size}(\phi_1)$;
- $\mathsf{size}(\phi_1 U \phi_2) = \mathsf{max}(\mathsf{size}(\phi_1), \mathsf{size}(\phi_2))$;
- $\mathsf{size}(\phi_1 S \phi_2) = \mathsf{max}(\mathsf{size}(\phi_1), \mathsf{size}(\phi_2))$;
- $\rhd_{\sim c}\, \phi_1 = \mathsf{size}(\phi_1)$;
- $\lhd_{\sim c}\, \phi_1 = \lceil \frac{c}{1.5} \rceil + \mathsf{size}(\phi_1)$;

For the real-time operator $\lhd$, we use the constant c and divide it by 1.5, because our notion of size is designed for the timed traces of Θ, where observations are separated by 1.5.

For example $\mathsf{size}(\bigcirc \neg \ominus \top) = 1$, $\mathsf{size}(\rhd_{=6}\, p) = 0$ and $\mathsf{size}(\lhd_{=14} \neg \ominus \top) = \lceil \frac{14}{1.5} \rceil + 1 = 11$, $\mathsf{size}(\ominus \ominus \ominus p) = 3$.

Lemma 8 *For every formula $\phi \in$ $\mathsf{rEventClockTL}$, for every model $\theta^k \in \Theta$ with $k > \mathsf{size}(\phi)$ then:*

- $P_1(\phi, \theta^k) = \forall i_1, i_2 \cdot 0 \leq \mathsf{size}(\phi) \leq i_1 < i_2$*:* $(\theta^k, i_1) \models \phi$ *iff* $(\theta^k, i_2) \models \phi$ *iff* $(\theta^\infty, i_1) \models \phi$ *iff* $(\theta^\infty, i_2) \models \phi$*;*
- $P_2(\phi, \theta^k) = \forall i \cdot 0 \leq i < \mathsf{size}(\phi)$*:* $(\theta^k, i) \models \phi$ *iff* $(\theta^\infty, i) \models \phi$*;*

$P_1(\phi, \theta^k)$ *expresses that: for every position i_1, i_2 after* $\mathsf{size}(\phi)$*, the formula ϕ is either constantly true in θ^k and θ^∞ or constantly false. $P_2(\phi, \theta^k)$ expresses that: for every position i before* $\mathsf{size}(\phi)$*, the formula ϕ is evaluated similarly in θ^k and θ^∞ (but its truth value may change from position to position). We note $P_3(\phi, \theta^k)$ the formula $\forall i \geq 0$, $(\theta^k, i) \models \phi$ iff $(\theta^\infty, i) \models \phi$. Note that $P_3(\phi, \theta^k)$ is a consequence of the conjunction of P_1 and P_2.*

Proof. The proof is by induction on the structure of formulas.

- $\phi = p$: as p is constantly true in all timed traces of Θ, then P_1 and P_2 are verified for the base case.
- The boolean cases are trivial.
- $\phi = \bigcirc \phi_1$. Note that $\mathsf{size}(\phi) = \mathsf{size}(\phi_1)$. By induction hypothesis, for all $k > \mathsf{size}(\phi)$, $P_1(\phi_1, \theta^k)$ and $P_2(\phi_1, \theta^k)$ holds and thus $P_3(\phi_1, \theta^k)$.
  1. As, by semantics of the $\bigcirc$-operator, the truth value of $\bigcirc \phi_1$ in position i only depends on the truth value of ϕ_1 in $i+1$ and $P_1(\phi_1, \theta^k)$ holds, we know that $\bigcirc \phi_1$ is constantly either true (if ϕ_1 is constantly true, by P_1 and induction hypothesis) in positions $i \geq \mathsf{size}(\phi)$ or constantly false (if ϕ_1 is constantly false by P_1 and induction hypothesis) in positions $i \geq \mathsf{size}(\phi)$, in both θ^k and θ^∞ and thus $P_1(\phi, \theta^k)$ is verified.
  2. Let us now try to establish $P_2(\phi, \theta^k)$. Again, we know that $P_3(\phi_1, \theta^k)$ is a consequence of the induction hypothesis. That is, ϕ_1 evaluates in the same way in every position of the two timed traces θ^k and θ^∞. By the semantics of the $\bigcirc$-operator $P_3(\phi, \theta^k)$ holds and thus $P_2(\phi, \theta^k)$.
- The U and S operators are treated in the same way.
- $\phi = \ominus \phi_1$. Note that $\mathsf{size}(\phi) = 1 + \mathsf{size}(\phi_1)$.

1. Let us first establish $P_1(\phi,\theta^k)$ for $k > \mathsf{size}(\phi)$. By induction hypothesis, we know that $P_1(\phi_1,\theta^k)$ holds. As a consequence, for all positions $i \geq \mathsf{size}(\phi) - 1$, ϕ_1 has the same constant value in θ^k and θ^∞. Thus in all positions $i \geq \mathsf{size}(\phi)$, $\ominus\phi_1$ has the same constant value in θ^k and θ^∞. And thus $P_1(\phi,\theta^k)$ is established.
2. Let us now turn to $P_2(\phi,\theta^k)$. By induction hypothesis $P_3(\phi_1,\theta^k)$ holds. That is, ϕ_1 has the same truth value in θ^k and θ^∞, for every position i. As the truth value of $\ominus\phi_1$ in all positions $i \geq 1$ only depends on the truth value of ϕ_1 and a $\ominus$-formula is always false in $i = 0$, $P_3(\phi,\theta^k)$ holds and thus $P_2(\phi,\theta^k)$ holds.

- $\phi = \rhd_{\sim c} \phi_1$. We know that $\mathsf{size}(\phi) = \mathsf{size}(\phi_1)$.
  1. We first establish $P_1(\phi,\theta^k)$ for $k > \mathsf{size}(\phi)$. By induction hypothesis, we know that either:
     (a) for all position $i \geq \mathsf{size}(\phi)$ that $(\theta^k,i) \models \phi_1$ and $(\theta^\infty,i) \models \phi_1$: In this case, for all $i \geq \mathsf{size}(\phi)$, the following ϕ_1 is at a distance of $d \in (1,2)$ and thus $\rhd_{\sim c} \phi_1$ is constantly true if $(1,2) \subseteq \{v | v \sim c\}$ and constantly false otherwise, in both timed traces θ^k and θ^∞.
     (b) for all position $i \geq \mathsf{size}(\phi)$ that $(\theta^k,i) \not\models \phi_1$ and $(\theta^\infty,i) \not\models \phi_1$: In that case, for all $i \geq \mathsf{size}(\phi)$, there is no following ϕ_1 position and thus $\rhd_{\sim c} \phi_1$ is constantly false in both timed traces θ^k and θ^∞.

     This establishes $P_1(\phi,\theta^k)$.
  2. Let us now turn to $P_2(\phi,\theta^k)$. First we know that for all positions $0 \leq i < k$, $\tau_i^k = \tau_i^\infty$, that is, the timing of the two timed traces agree. We also now that ϕ_1 has the same constant value in the two traces after position $i = \mathsf{size}(\phi) < k$. Let us consider any position l such that $0 \leq l < \mathsf{size}(\phi) < k$, the following ϕ_1 must be true in a location m, $l < m \leq \mathsf{size}(\phi)$ or it will be false for ever. In the last case $\rhd_{\sim c} \phi_1$ is false in the two timed traces. In the case that ϕ_1 is true in a position m, $l < m \leq \mathsf{size}(\phi) < k$, the formula $\rhd_{\sim c} \phi_1$ evaluates similarly in the two timed traces as their timing information is the same for all positions i, $0 \leq i < k$. And thus property $P_2(\phi,\theta^k)$ holds.

- $\phi = \lhd_{\sim c} \phi_1$. First, note that if $c = 0$ then $\lhd_{\sim c} \phi$ is equivalent to false as the $\lhd$ operator is irreflexive and time is strictly monotone. Let us consider the case where $c > 0$. Let $d = \lceil \frac{c}{1.5} \rceil$. Note that $d \geq 1$. We know that $\mathsf{size}(\phi) = d + \mathsf{size}(\phi_1)$.
  1. We first establish P_1. By induction hypothesis, we know that either:

(a) $\forall i : \mathsf{size}(\phi) - d \leq i$: $(\theta^k, i) \models \phi_1$ and $(\theta^\infty, i) \models \phi_1$. For every position j such that $\mathsf{size}(\phi) \leq j$, the last ϕ_1 position is $j-1$ at a distance $d_a \in (1,2)$. As a consequence, for all j such that $\mathsf{size}(\phi) \leq j$, $(\theta^k, j) \models \rhd_{\sim c} \phi_1$ iff $(\theta^\infty, j) \models \rhd_{\sim c} \phi_1$ iff $(1,2) \subseteq \{v | v \sim c\}$; and thus ϕ is either true in all $j \geq \mathsf{size}(\phi)$ in both θ^k and θ^∞ or it is false in all $j \geq \mathsf{size}(\phi)$ in both θ^k and θ^∞.

(b) $\forall i : \mathsf{size}(\phi) - d \leq i$: $(\theta^k, i) \not\models \phi_1$ and $(\theta^\infty, i) \not\models \phi_1$. For every position j such that $j \geq \mathsf{size}(\phi)$, the last ϕ_1 observation is, if it exists, in a position i with $0 \leq i < \mathsf{size}(\phi) - d$. Thus at a distance $d_b > d \times 1.5 \geq c$ and thus $\lhd_{\sim c} \phi_1$ is verified in all positions $j \geq \mathsf{size}(\phi)$, both in θ^k and θ^∞ if "$\sim$" = "$>$" or "$\geq$" and the ϕ_1-position exists. In all other cases, $\lhd_{\sim c} \phi_1$ is false in all positions $j \geq \mathsf{size}(\phi)$, both in θ^k and θ^∞.

And thus $P_1(\phi, \theta^k)$ holds.

2. Let us now turn to the property P_2. So, we want to establish that $\forall i \cdot 0 \leq i < \mathsf{size}(\phi)$: $(\theta^k, i) \models \phi$ iff $(\theta^\infty, i) \models \phi$. Let us first note that the value of $\lhd_{\sim c} \phi_1$ in the positions i, $0 \leq i < \mathsf{size}(\phi)$, only depends on the truth value of ϕ_1 in $0 \leq i < \mathsf{size}(\phi)$ and the timing information for θ^k and θ^∞. The value of ϕ_1 is similar in those positions for the two models by induction hypothesis. Also the timing information in that interval of positions is identical as $k > \mathsf{size}(\phi)$ and thus the value of $\lhd_{\sim c} \phi_1$ is exactly the same for each position i, $0 \leq i < \mathsf{size}(\phi) < k$, in both θ^k and θ^∞. And thus $P_2(\phi, \theta^k)$ holds.

□

Theorem 11 MetricIntervalTL *is strictly more expressive than* rEventClockTL*:* MetricIntervalTL $\supset$ rEventClockTL*.*

Proof. By lemma 8, we know that for every formula $\phi \in$ rEventClockTL, there exists a bound l such that for all θ^k with $k > l$, $\theta^k \in L(\phi)$ iff $\theta^\infty \in L(\phi)$. On the other hand, the formula $\psi \equiv \widehat{\Box}_{(0,\infty)}(p \rightarrow \widehat{\Diamond}_{[2,3]} p)$ of MetricIntervalTL is satisfied by θ^∞ but by none of the timed traces $\theta^k \in \Theta$. Thus rEventClockTL cannot express the property expressed by ψ. □

Let us now show that every $\mathsf{MetricIntervalTL}_{0,\infty}$-property can be expressed by an rEventClockTL-formula. This is a consequence of the following stronger lemma:

Lemma 9 *For every formula* $\phi \in \mathsf{MetricIntervalTL}_{0,\infty}$*, there exists a formula* $\phi^T \in$ rEventClockTL *such that, for every timed trace* θ*, for every position* i*:* $(\theta, i) \models \phi$ *iff* $(\theta, i) \models \phi^T$*.*

Proof. We reason by induction on the structure of formulas. The boolean cases are trivial. We only treat the $\widehat{U}_I$: the similar $\widehat{S}_I$ is left to the reader. First note that the following rewritings within $\mathsf{MetricIntervalTL}_{0,\infty}$ are valid:

- $\phi_1 \widehat{U}_{<c} \phi_2 = (\phi_1 \widehat{U}_{(0,\infty)} \phi_2) \wedge \widehat{\Diamond}_{<c} \phi_2$;
- $\phi_1 \widehat{U}_{\leq c} \phi_2 = (\phi_1 \widehat{U}_{(0,\infty)} \phi_2) \wedge \widehat{\Diamond}_{\leq c} \phi_2$;

Now it is easy to show that: $\widehat{\Diamond}_{<c}\phi = \rhd_{<c} \phi^T$ and $\widehat{\Diamond}_{\leq c}\phi = \rhd_{\leq c} \phi^T$. By definition of the $\widehat{\Box}$-operator, we also have: $\widehat{\Box}_{<c}\phi = \neg \rhd_{<c} \neg\phi^T$ and $\widehat{\Box}_{\leq c}\phi = \neg \rhd_{\leq c} \neg\phi^T$. Also, we have that $\phi_1 \widehat{U}_{(0,\infty)} \phi_2 = \bigcirc(\phi_1^T U \phi_2^T)$ and thus every $\widehat{U}_{<,\leq c}$ formula can be expressed in $\mathsf{rEventClockTL}$. Let us now turn to the $\widehat{U}_{>,\geq}$ cases. Here are the translations (we use $\widehat{\Box}_{<,\leq}$ in $\mathsf{rEventClockTL}$ formulas, since we have shown just above that it can be translated in "plain" $\mathsf{rEventClockTL}$):

- $\phi_1 \widehat{U}_{>c} \phi_2 = \widehat{\Box}_{\leq c}(\phi_1^T \wedge \bigcirc(\phi_1^T U \phi_2^T)) \wedge \bigcirc(\phi_1^T U \phi_2^T)$;
- $\phi_1 \widehat{U}_{\geq c} \phi_2 = \widehat{\Box}_{<c}(\phi_1^T \wedge \bigcirc(\phi_1^T U \phi_2^T)) \wedge \bigcirc(\phi_1^T U \phi_2^T)$;

We justify the right to left implication for $\widehat{U}_{>c}$. Thus we must show that if $(\theta, i) \models \widehat{\Box}_{\leq c}(\phi_1^T \wedge \bigcirc(\phi_1^T U \phi_2^T)) \wedge \bigcirc(\phi_1^T U \phi_2^T)$ then $(\theta, i) \models \phi_1 \widehat{U}_{>c} \phi_2$. Let $J = \{j | \tau_i < \tau_j \leq \tau_i + c\}$, that is, J is the set of positions after position i that are at a time distance less or equal to c from i. We consider two disjoint situations:

- (a) $J = \emptyset$. There is no position $J > i$ such that $\tau_j \leq \tau_i + c$ then verifying $(\theta, i) \models \bigcirc(\phi_1^T U \phi_2^T)$ is sufficient because the first ϕ_2-position will be at a distance $d > c$ from i and between this ϕ_2 position and after i, ϕ_1 is verified;
- (b) $J \neq \emptyset$. There is some position in the interval $(\tau_i, \tau_i + c]$, the formula $\widehat{\Box}_{\leq c}(\phi_1^T \wedge \bigcirc(\phi_1^T U \phi_2^T))$ imposes that ϕ_1 is constantly true in the interval $(\tau_i, \tau_i + c]$ and also that in the last position of that interval, let say k, that $\bigcirc(\phi_1^T U \phi_2^T)$ is true and thus $\phi_1^T U \phi_2^T$ is true in position $k+1$ and ensures that ϕ_1 will stay true until a ϕ_2 position is encountered at a distance $d > c$ from position i.

□

The lemma 9 and theorem 9 together give:

Theorem 12 *The logics* $\mathsf{MetricIntervalTL}_{0,\infty}$ *and* $\mathsf{rEventClockTL}$ *are equally expressive, i.e.* $\mathsf{MetricIntervalTL}_{0,\infty} = \mathsf{rEventClockTL}$.

Note that the translation between formulas of one logic to the other does not change the size of the maximal constant used, generates only a linear number of subformulas and thus the number of real-time operators stays also linear. Note also that the last theorem also apply for the future fragment of the two logics as future formulas of one logic are always translated by future formula of the other logic.

Theorem 13 *The two logics* $\mathsf{MetricIntervalTL}^{F}_{0,\infty}$ *and* $\mathsf{rEventClockTL}^{F}$ *are equally expressive, i.e.* $\mathsf{MetricIntervalTL}^{F}_{0,\infty} = \mathsf{rEventClockTL}^{F}$.

Corollary 5 MetricIntervalTL *is more expressive than* $\mathsf{MetricIntervalTL}_{0,\infty}$*:* $\mathsf{MetricIntervalTL} \supset \mathsf{MetricIntervalTL}_{0,\infty}$.

Remark. In [HRS98], it is proved that the expressive powers of MetricIntervalTL and rEventClockTL agree when evaluated continuously, i.e. on timed state sequences, see appendix for the definition of rEventClockTL in continuous models. We refer the interested reader to [HRS98] for details about this interesting phenomenon.

6.2 rEventClockTL vs EventClockTA

It is well known that LTL cannot express some counting properties that are expressible by Büchi automata. For example, there does not exist any LTL formula that expresses the $even - p$ property: "p is true in all even positions of the trace", while this property is easily expressed by an automaton. Similarly, rEventClockTL cannot express counting properties. As EventClockTA are an extension of Büchi automata, and thus more expressive, we have the following theorem:

Lemma 10 *There exist* EventClockTA*-properties that are not expressible into* rEventClockTL*, i.e.* $\mathsf{EventClockTA} \not\subseteq \mathsf{rEventClockTL}$.

Let us take a look at the other direction. Without real-time, we know that every LTL property is expressible by Büchi automaton. Similarly, is every rEventClockTL-property expressible by an event clock automaton? Surprisingly, the answer is negative:

Lemma 11 *There exist* rEventClockTL*-properties that are not expressible into* EventClockTA*, i.e.* $\mathsf{rEventClockTL} \not\subseteq \mathsf{EventClockTA}$.

Proof. To show that not all rEventClockTL-properties are expressible with EventClockTA, we consider the two timed traces $\theta^1 = (\bar{\sigma}, \bar{\tau}^1)$ and $\theta^2 = (\bar{\sigma}, \bar{\tau}^2)$ on the set of propositions $\mathcal{P} = \{p\}$. θ^1 and θ^2 share the same qualitative information $\bar{\sigma} = \{\}\{p\}\{\}\{p\}\{p\}...\{p\}...$ that is, p is true everywhere except in position 0 and 2. The timing information $\bar{\tau}^1$ of θ^1 is as follows: $\tau^1_i = i \times 1.4$, i.e. all positions are separated by 1.4 time units: $\bar{\tau}^1 = 0, 1.4, 2.8, 4.2, 5.6, 7, \ldots$. On the other hand the timing information $\bar{\tau}^2$ of θ^2 is defined by:

$$\tau_i^2 = \begin{cases} i \times 1.4 & \text{if } i \neq 3 \\ i \times 1.4 - 0.3 & \text{if } i = 3 \end{cases}$$

Thus, $\bar{\tau}^2 = 0, 1.4, 2.8, 3.9, 5.6, 7, \ldots$. It is easy to show that for every position $i \geq 0$, for every constraint $x_p \sim c$, $(\theta^1, i) \models x_p \sim c$ iff $(\theta^2, i) \models x_p \sim c$. Similarly, it is easy to show that for every position $i \geq 0$, for every constraint $y_p \sim c$, $(\theta^1, i) \models y_p \sim c$ iff $(\theta^2, i) \models y_p \sim c$. That is, the constraints about history and prophecy clocks associated with p evaluate similarly in both models, in all positions. This is because clock constraints can only use integer constants and, for the prophecy clock y_p:

- in position 0, the distance to the following p-position, and thus the value of y_p, is, in the two models, equal to $1.4 \in (1, 2)$;
- in position 1, this distance is equal to $2.8 \in (2, 3)$ in θ^1 and to $2.5 \in (2, 3)$ in θ^2. Even if the distances are different, it cannot be seen using integer constants;
- in position 2, this distance is equal to $1.4 \in (1, 2)$ in θ^1 and is equal to $1.1 \in (1, 2)$ in θ^2. Again even if the distances are different, it cannot be seen using integer constants;
- in position 3, this distance is equal to $1.4 \in (1, 2)$ in θ^1 and is equal to $1.9 \in (1, 2)$ in θ^2. Again even if the distances are different, it cannot be seen using integer constants;
- after position 3, the distance to the following p position is always, in both θ^1, θ^2, equal to $1.4 \in (1, 2)$.

A same reasoning applies for the history clock x_p. So every event clock automaton A accepts θ^1 if and only if it accepts θ^2. On the other hand the recursive rEventClockTL-formula $\psi = \rhd_{\leq 4} \Box p$ is true in the initial position of θ^2 but false in the initial position of θ^1. And thus rEventClockTL can differentiate between the two models. $\Box$

¿From lemma 10 and lemma 11, we obtain the following theorem:

Theorem 14 *The expressive power of* rEventClockTL *and* EventClockTA *are incomparable.*

This surprising theorem can be explained as follows: when moving from EventClockTA to rEventClockTL, we have automatically added recursion, or uniform substitution in logician parlance: any formula can replace a proposition symbol. If we remove this possibility, we obtain the expected result from corollary 2:

Corollary 6 *The models of a non-recursive formula is the language of an* EventClockTA*:* EventClockTL $\subset$ EventClockTA

We can also obtain an inclusion by introducing recursion [5] in EventClockTA, as described in [HRS98].

7 Conclusion

In this paper, we have presented a new real-time logic called rEventClockTL. This logic extends LTL with real-time operators $\rhd_{\sim c}\ \rho$ read "the next ρ is at a distance d that respects $d \sim c$", and symmetrically $\lhd_{\sim c}\ \rho$ expressing that "ρ was last true at a distance d such that $d \sim c$". These two modal operators introduce the clean and powerful concept of event clock, from timed automata [AFH94], in the domain of real-time logics. The natural expressive power of those two operators has been illustrated by showing that most important real-time requirements [Koy92] can be straightforwardly and naturally expressed in rEventClockTL.

We have shown that the problems of satisfiability, validity and model-checking are decidable for this logic, more precisely PSPACE-complete, as for LTL. We provided a simple decision procedure based on EventClockTA, a determinizable class of timed automata. Our decision procedure is far less complicated than the decision procedure of [AFH96] for MetricIntervalTL, the only real-time logic that was previously known to be decidable. Our decision procedure is obtained by extending the decision procedure for LTL in a natural way: we use the close connection that exists between the two real-time operators of rEventClockTL and the prophecy and history clocks of EventClockTA. This naturalness helps in axiomatizing this logic, see [RSH98]. Corresponding monadic logics are built in [HRS98].

References

[ACD90] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. In *Proceedings of the 5th Symposium on Logic in Computer Science*, pages 414–425, Philadelphia, June 1990.

[AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994. Preliminary version appears in Proc. 17th ICALP, 1990, LNCS 443.

[AFH94] R. Alur, L. Fix, and T.A. Henzinger. A determinizable class of timed automata. In *Proceedings of the Sixth Conference on*

[5] Simple generalizations as the association of boolean formulas with clocks is not sufficient

Computer-Aided Verification, Lecture Notes in Computer Science 818, pages 1–13. Springer-Verlag, 1994.

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

[AH92a] R. Alur and T.A. Henzinger. Back to the future: towards a theory of timed regular languages. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 177–186. IEEE Computer Society Press, 1992.

[AH92b] R. Alur and T.A. Henzinger. Logics and models of real time: a survey. In J.W. de Bakker, K. Huizing, W.-P. de Roever, and G. Rozenberg, editors, *Real Time: Theory in Practice*, Lecture Notes in Computer Science 600, pages 74–106. Springer-Verlag, 1992.

[AH93] R. Alur and T.A. Henzinger. Real-time logics: complexity and expressiveness. *Information and Computation*, 104(1):35–77, 1993. Preliminary version appears in the Proc. of 5th LICS, 1990.

[AH94] R. Alur and T.A. Henzinger. A really temporal logic. *Journal of the ACM*, 41(1):181–204, 1994. Preliminary version appears in Proc. 30th FOCS, 1989.

[BCM+90] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic model checking: 10^{20} states and beyond. In *Proceedings of the 5th Symposium on Logic in Computer Science*, pages 428–439, Philadelphia, June 1990.

[BKP86] H. Barringer, R. Kuiper, and A. Pnueli. A really abstract concurrent model and its temporal logic. In *Proceedings of the 13th Annual Symposium on Principles of Programming Languages*, pages 173–183. ACM Press, 1986.

[CES86] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, January 1986.

[Eme90] E.A. Emerson. *Handbook in Theoretical Computer Science, Formal Models and Semantics*, chapter Temporal and Modal Logic, pages 995–1072. Elsevier, 1990.

[GPSS80] Dov Gabbay, Amir Pnueli, Saharon Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proceedings of the Seventh ACM*

Symposium on Principles of Programming Languages, pages 163–173. ACM, 1980.

[GPVW95] R. Gerth, D. Peled, M. Y. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Proc. 15th Work. Protocol Specification, Testing, and Verification*, Warsaw, June 1995. North-Holland.

[Hen96] T.A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, pages 278–292. IEEE Computer Society Press, 1996.

[HMP92] T.A. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In W. Kuich, editor, *ICALP 92: Automata, Languages, and Programming*, Lecture Notes in Computer Science 623, pages 545–558. Springer-Verlag, 1992.

[HRS98] T.A. Henzinger, J.-F. Raskin, and P.-Y. Schobbens. The regular real-time languages. In K.G. Larsen, S. Skyum, and G. Winskel, editors, *ICALP'98: Automata, Languages, and Programming*, Lecture Notes in Computer Science 1443, pages 580–591. Springer-Verlag, 1998.

[Koy92] Ron Koymans. *Specifying message passing and time-critical systems with temporal logic.* LNCS 651, Springer-Verlag, 1992.

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

[MP95] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems : Safety.* Springer-Verlag, Berlin, January 1995.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symposium on Foundation of Computer Science*, pages 46–57, 1977.

[RS97a] J.-F. Raskin and P.-Y. Schobbens. Real-time logics: Fictitious clock as an abstraction of dense time. In *Proc. Third International Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS97)*, volume 1217 of *Lecture Notes in Computer Science (LNCS)*, pages 165–182. Springer-Verlag, 1997.

[RS97b] J.-F. Raskin and P.-Y. Schobbens. State clock logic: a decidable real-time logic. In *Proc. Hart'97 : Hybrid and Real-Time Systems, LNCSs 1201*, pages 15–30, Grenoble, France, 1997. Springer.

[RSH98] J.-F. Raskin, P.-Y. Schobbens, and T. Henzinger. Axioms for real-time logics. In D. Sangiorgi and R. de Simone, editors, *Proceedings of CONCUR'98: 9th International Conference on Concurrency Theory*, volume 1466 of *Lecture Notes in Computer Science (LNCS)*. Springer, 1998.

[Sti87] C. Stirling. Comparing linear and branching time temporal logics. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Temporal Logic in Specification*, volume 398, pages 1–20. Lecture Notes in Computer Science, Springer-Verlag, 1987.

[SVW85] A.P. Sistla, M.Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. In *Proc. 10th Int. Colloquium on Automata, Languages and Programming*, volume 194, pages 465–474, Nafplion, July 1985. LNCS, Springer-Verlag.

[Wol85] P. Wolper. The tableau method for temporal logic: An overview. *Logique et Analyse*, (110–111):119–136, 1985.

A Continuous Interpretation

An *interval* $I \subseteq \mathbb{R}^+$ is a convex nonempty subset of the nonnegative reals. Given $t \in \mathbb{R}^+$, we freely use notation such as $t+I$ for the interval $\{t' \mid$ exists $t'' \in I$ with $t' = t + t''\}$, and $t > I$ for the constraint "$t > t'$ for all $t' \in I$." Two intervals I and J are *adjacent* if the right endpoint of I is equal to the left endpoint of J, and either I is right-open and J is left-closed or I is right-closed and J is left-open. An *interval sequence* $\bar{I} = I_0, I_1, \ldots$ is a finite or infinite sequence of bounded intervals so that for all $i \geq 0$, the intervals I_i and I_{i+1} are adjacent. We say that the interval sequence $\bar{I}$ *covers* the interval $\bigcup_{i\geq 0} I_i$. If $\bar{I}$ covers $[0, \infty)$, then $\bar{I}$ partitions the nonnegative real line so that every bounded subset of $\mathbb{R}^+$ is contained within a finite union of elements from the partition.

Let $\mathcal{P}$ be a finite set of proposition symbols. A *state* $s \subseteq \mathcal{P}$ is a set of propositions. A *timed state sequence* $\kappa = (\bar{s}, \bar{I})$ is a pair that consists of an infinite sequence $\bar{s}$ of states and an infinite interval sequence $\bar{I}$ that covers $[0, \infty)$. Equivalently, the timed state sequence κ can be viewed as a function from $\mathbb{R}^+$ to $2^{\mathcal{P}}$, indicating for each time $t \in \mathbb{R}^+$ a state $\kappa(t) \subseteq \mathcal{P}$.

The formulas of $\mathsf{rEventClockTL}$ [RS97b] are built from propositional symbols, boolean connectives, the temporal "until" and "since" operators, and two real-time operators: at any time t, the *history operator* $\lhd_I\ \phi$ asserts that ϕ was true last time in the interval $t - I$, and the *prophecy operator* $\rhd_I\ \phi$ asserts that ϕ will be true next time in the interval $t+I$. The formulas of $\mathsf{rEventClockTL}$ are generated by the following grammar:

$$\phi \ ::= \ p \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \phi_1 U \phi_2 \mid \phi_1 S \phi_2 \mid \lhd_I\ \phi \mid \rhd_I\ \phi$$

where p is a proposition and I is an interval whose finite endpoints are nonnegative integers. Let ϕ be an $\mathsf{rEventClockTL}$ formula and let κ be a timed state sequence whose proposition symbols contain all proposition symbols that occur in ϕ. The formula ϕ *holds* at time $t \in \mathbb{R}^+$ of κ, denoted $(\kappa, t) \models \phi$, according to the following definition:

$(\kappa, t) \models p$ iff $p \in \kappa(t)$
$(\kappa, t) \models \phi_1 \wedge \phi_2$ iff $(\kappa, t) \models \phi_1$ and $(\kappa, t) \models \phi_2$
$(\kappa, t) \models \neg\phi$ iff not $(\kappa, t) \models \phi$
$(\kappa, t) \models \phi_1 U \phi_2$ iff exists a real $t' > t$ with $(\kappa, t') \models \phi_2$, and for all reals $t'' \in (t, t')$, we have $(\kappa, t'') \models \phi_1 \vee \phi_2$
$(\kappa, t) \models \phi_1 S \phi_2$ iff exists a real $t' < t$ with $(\kappa, t') \models \phi_2$, and for all reals $t'' \in (t', t)$, we have $(\kappa, t'') \models \phi_1 \vee \phi_2$
$(\kappa, t) \models \lhd_I\ \phi$ iff exists a real $t' < t$ with $t' \in (t-I)$ and $(\kappa, t') \models \phi$, and for all reals $t'' < t$ with $t'' > (t - I)$, not $(\kappa, t'') \models \phi$
$(\kappa, t) \models \rhd_I\ \phi$ iff exists a real $t' > t$ with $t' \in (t+I)$ and $(\kappa, t') \models \phi$, and for all reals $t'' > t$ with $t'' < (t + I)$, not $(\kappa, t'') \models \phi$

Note that the temporal and real-time operators are defined in a strict manner; that is, they do not constrain the current state. Non-strict operators are easily defined from their strict counterparts.

Theorem 15 *[RS97b] The satisfiability and validity problems for* rEventClockTL *in timed state sequences (continuous interpretation) are decidable in* PSPACE.

I N F O R M A T I K

Below you find a list of the most recent technical reports of the research group *Logic of Programming* at the Max-Planck-Institut für Informatik. They are available by anonymous ftp from our ftp server `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Birgit Hofmann
Im Stadtwald
D-66123 Saarbrücken
GERMANY
e-mail: `library@mpi-sb.mpg.de`

MPI-I-98-2-011	A. Degtyarev, A. Voronkov	Equality Reasoning in Sequent-Based Calculi
MPI-I-98-2-010	S. Ramangalahy	Strategies for Conformance Testing
MPI-I-98-2-009	S. Vorobyov	The Undecidability of the First-Order Theories of One Step Rewriting in Linear Canonical Systems
MPI-I-98-2-008	S. Vorobyov	AE-Equational theory of context unification is Co-RE-Hard
MPI-I-98-2-007	S. Vorobyov	The Most Nonelementary Theory (A Direct Lower Bound Proof)
MPI-I-98-2-006	P. Blackburn, M. Tzakova	Hybrid Languages and Temporal Logic
MPI-I-98-2-005	M. Veanes	The Relation Between Second-Order Unification and Simultaneous Rigid *E*-Unification
MPI-I-98-2-004	S. Vorobyov	Satisfiability of Functional+Record Subtype Constraints is NP-Hard
MPI-I-98-2-003	R.A. Schmidt	E-Unification for Subsystems of S4
MPI-I-97-2-012	L. Bachmair, H. Ganzinger, A. Voronkov	Elimination of Equality via Transformation with Ordering Constraints
MPI-I-97-2-011	L. Bachmair, H. Ganzinger	Strict Basic Superposition and Chaining
MPI-I-97-2-010	S. Vorobyov, A. Voronkov	Complexity of Nonrecursive Logic Programs with Complex Values
MPI-I-97-2-009	A. Bockmayr, F. Eisenbrand	On the Chvátal Rank of Polytopes in the 0/1 Cube
MPI-I-97-2-008	A. Bockmayr, T. Kasper	A Unifying Framework for Integer and Finite Domain Constraint Programming
MPI-I-97-2-007	P. Blackburn, M. Tzakova	Two Hybrid Logics
MPI-I-97-2-006	S. Vorobyov	Third-order matching in $\lambda \rightarrow$-Curry is undecidable
MPI-I-97-2-005	L. Bachmair, H. Ganzinger	A Theory of Resolution
MPI-I-97-2-004	W. Charatonik, A. Podelski	Solving set constraints for greatest models
MPI-I-97-2-003	U. Hustadt, R.A. Schmidt	On evaluating decision procedures for modal logic
MPI-I-97-2-002	R.A. Schmidt	Resolution is a decision procedure for many propositional modal logics