Real Time Temporal Logic: Past, Present, Future*

Oded Maler[1], Dejan Nickovic[1], and Amir Pnueli[2,3]

[1] Verimag, 2 Av. de Vignate, 38610 Gières, France
{Dejan.Nickovic, Oded.Maler}@imag.fr
[2] Weizmann Institute of Science, Rehovot 76100, Israel
[3] New York University, 251 Mercer St. New York, NY 10012, USA
Amir.Pnueli@cs.nyu.edu

Abstract. This paper attempts to improve our understanding of timed languages and their relation to timed automata. We start by giving a constructive proof of the folk theorem stating that timed languages specified by the past fragment of MITL, can be accepted by deterministic timed automata. On the other hand we provide a proof that certain languages expressed in the future fragment of MITL are not deterministic,[1] and analyze the reason for this asymmetry.

1 Introduction

In this paper we compare the past and future fragments of the real-time temporal logic MITL [AFH96] with respect to the recognizability of their models by *deterministic* timed automata. To put our work in context we first discuss past and future in untimed temporal logic, the question of online and offline monitoring as well as some related work on real-time logics and timed languages.

1.1 Past and Future in LTL

Propositional linear-time temporal logic (LTL) is a commonly-accepted formalism for specifying properties of finite-state discrete systems [MP95b]. The semantic models for LTL are typically sequences which are *infinite* toward the *future* and *finite* toward the *past*.[2] On this semantic domain there is a "typing" asymmetry between models of properties expressed in the *past* fragment of LTL, which are star-free[3] regular *languages*, and models for formulae written using the *future* fragment which are star-free regular ω*-languages*. To facilitate a closer comparison of the expressive power of the two

* This work was partially supported by the European Community project IST-2003-507219 PROSYD (Property-based System Design).

[1] As far as we know, no systematic techniques for proving such facts have been developed for timed automata since their introduction 15 years ago until recently.

[2] In other words the "carrier set" is isomorphic to $\mathbb{N}$, not $\mathbb{Z}$. Languages over bi-infinite sequences have been studied in [NP86].

[3] The word *star-free* comes from the characterization of these languages as those definable using a special class of regular expressions the do not use the Kleene star but allow intersection and complementation, see [MNP71].

P. Pettersson and W. Yi (Eds.): FORMATS 2005, LNCS 3829, pp. 2–16, 2005.

formalisms, one can unify their semantic domains by interpreting future LTL over finite sequences. This can be done, for example, by extending LTL semantics to include "truncated" (finite) paths as in [EFH$^+$03]. Getting rid of the ω-dimension we can focus on the differences between the two formalisms which are related to the *direction* of the temporal modalities.

When an automaton reads a sequence, the current state of the automaton represents (the equivalence class of) the prefix read so far. Past LTL fits naturally this point of view as satisfaction is determined *now* by what happened from time zero *until now*. Future LTL, on the other hand, states at time zero what it expects to see or not to see in the future. As time progresses, some of those "obligations" are fulfilled (or violated) and some new ones are generated. Satisfaction is established if all obligations are met at the end of the sequence. The translation from LTL formulae to automata that accept their models is one of the cornerstones of formal verification [VW86], and most of the work on the topic focused on future properties and ω-automata. From a future LTL formula one can construct naturally an alternating or a non-deterministic automaton that accepts the language. Such an automaton can be determinized either by the non-trivial construction of Safra for ω-automata [Saf88], or by the simpler subset construction if we take the finitary interpretation. The translation from past LTL to automata is more folklore, but it is not hard to see that it translates *naturally* to deterministic automata, a fact the also explains the simplicity of the online monitoring procedure in [HR02]. So the bottom line for LTL is that both the past and future fragments can be eventually translated into deterministic automata.[4]

1.2 Deterministic Automata and Online Monitors

Monitoring is the process of testing whether a given behavior ξ satisfies a property φ (or, equivalently, belongs to the corresponding language L). This process can be performed in two different fashions. Offline monitoring starts *after* the whole sequence is given. Online monitoring is interleaved with the process of reading the sequence and is similar to the way the sequence is read by an automaton. Online monitors can detect violation or satisfaction *as soon as they happen*, which can be after a small prefix of the sequence.[5] This is advantageous for two reasons: for monitoring *real* systems (rather than simulated ones) offline monitoring is a post-factum analysis and can be too late to be useful. Even for simulated systems, where monitoring is used as a lightweight alternative to formal verification, early detection may reduce simulation time significantly. In analog circuits, the application domain that triggered this work, simulations can be *very* long.

[4] We mention the results of [MP90] which show how to go from counter-free automata to past LTL formulae and from counter-free ω-automata to mixed past-future formulae which are Boolean combinations of formulae of the form $\Box \Diamond \varphi$ where φ is a past formula. An alternative proof of the fact that all LTL formulae can be brought to this normal form appears in [LPZ85].

[5] To be more precise, violation or satisfaction of a property based on a prefix can be declared when all possible continuations of the prefix are equivalent with respect to the formula. Such a prefix is called "definitive" in [EFH$^+$03]. If the corresponding automaton is minimal, this fact can be easily detected by entering a "sink"state, either rejecting (for violation of safety) or accepting (satisfaction of eventuality). For non-minimal automata the analysis is a bit more involved.

In [MN04] we have developed an offline monitoring procedure for the real-time logic MITL$_{[a,b]}$. This procedure, which was used to monitor properties of real-valued signals, scans the signal from the end backwards and propagates truth values from sub-formulae "upward" and from the present to the past. In order to have an online version of this procedure, we somehow need to produce an automaton-like mechanism that reads Boolean signals, and whose state during reading is sufficiently detailed to detect acceptance or rejection as they occur. To follow the same recipe as in the untimed case, one would like to transform a formula to a timed automaton to be used as a monitoring procedure. However, the natural translation of MITL yields non-deterministic or alternating timed automata which, in the general case, do not determinize [AD94]. There are several remedies for this problem:

1. Use the important observation of Tripakis [Tri02, KT04] that on-the-fly determinization with respect to a *given* non-Zeno signal is possible for any timed automaton. The reason for non-determinizability of certain automata is the need to memorize the times of *all* events that have occurred within a bounded time window, without any a-priori bound on their number. In monitoring, we observe a signal with a *fixed* number of events, which can generate only a finite number of non-deterministic choices and hence the restriction of the automaton to this signal is amenable to subset construction.
2. Develop a piecewise-backward version of the procedure in [MN04] which after every new event or sampling point, restarts the propagation of truth values backwards (in most cases the propagation need not go back too far).
3. Use specification formalisms that correspond to deterministic timed automata.

This work is the result of attempting to follow the third approach.

1.3 Related Work

The study of real-time specification formalisms started in the eighties and generated numerous logics, results and papers. The reader is advised to look at surveys and discussions of these logics [AH92a, Hen98, HR04], of timed automata [Alu99] and timed languages in general [Asa04]. Without purporting to be exhaustive, we mention some relevant results.

The real-time logic MITL was introduced in [AFH96] as a restriction of the more general logic MTL (metric temporal logic) of [Koy90]. The restriction of time modalities to positive-length intervals was intended to guarantee decidability but recent results [OW05, LW05] show that this restriction is not necessary for deciding MTL over finitary event-based semantics. The original version of MITL contained only future temporal operators and [AFH96] give a procedure for translating an MITL formula into a non-deterministic timed automaton with the satisfiability and model-checking problems being EXPSPACE-complete. The non-determinizable nature of MITL is hinted in the paper.

Event-recording automata, where only the time of the *last* occurrence of every input letter can be remembered by a clock, have been shown to be determinizable in [AFH99]. Event-clock automata, introduced in the same paper, constitute a generalization of the latter which allow also "event-predicting" clocks. Event-clock logic is another decid-

able real-time logic which is equally expressive as MITL [RSH98] and which can be naturally translated into determinizable event-clock automata. However those become non-deterministic when expressed as classical Alur-Dill automata.[6]

An investigation of past and future versions of MITL was carried out in [AH92b] where the "prediction" feature of event-clock automata was replaced by the ability of the automaton to change the direction of reading. The authors describe a strict hierarchy of timed languages based on the number of direction reversals needed to recognize them (which corresponds roughly to the nesting depth of past and future operators). The deterministic nature of the past fragment of MITL is mentioned as a corollary of that hierarchy but no explicit proof is given.

Real-time monitoring tools often rely on temporal logics as their property specification language, but typically under a *discrete-time* interpretation. For example, [KPA03] use LTL_t, standard LTL augmented with *freeze quantifiers*, while in [TR04] the monitoring procedure uses MTL. In [Gei02] the dense semantics is preserved but additional restrictions on MITL are imposed in order to guarantee determinizability. These include the restriction of timed modalities to intervals of the form $[0, d]$ and disallowing arbitrary nesting of temporal operators.

In [MP04] we started focusing on deterministic timed automata because of the belief that some fundamental concepts of automata theory are better studied in a deterministic framework. We have defined there a notion of *recognizability* and have shown that is coincides with acceptance by deterministic timed automata. The current paper is part of the quest for a matching specification formalism.

The rest of the paper is organized as follows. In Section 2 we describe signals along with the logic MITL. In Section 3 we define the variant of timed automata that we use as signal acceptors. The proof of determinizability of the past fragment of MITL is given in Section 4 followed, in Section 5, by the proof of non-determinizability of the future fragment and a discussion of the reasons. Further contemplations close the paper.

2 Signals and Their Temporal Logic

Two basic semantic domains can be used to describe timed behaviors. *Time-event sequences* consist of instantaneous events separated by time durations while discrete-valued *signals* are functions from time to some discrete domain. The reader may consult the introduction to [ACM02] or [Asa04] for more details on the algebraic characterization of these domains. In this work we use Boolean signals as the semantic domain, but the extension of the results to time-event sequences (which are equivalent to the timed traces used by Alur and Dill [AD94]) need not be a difficult exercise.

Let the time domain $\mathbb{T}$ be the set $\mathbb{R}_{\geq 0}$ of non-negative real numbers. A finite length Boolean signal ξ is a partial function $\xi : \mathbb{T} \to \mathbb{B}^n$ whose domain of definition is an interval $I = [0, r)$, $r \in \mathbb{N}$. We say that the length of the signal is r and denote this

[6] One may argue that deterministic event-clock automata preserve one essential feature of determinism, namely, a unique run for every input signal, but this comes at the expense of losing the *causality* of the runs due to the prediction feature which amounts to going back and forth in time.

fact by $|\xi| = r$. We use $\xi[t]$ for the value of the signal at time t and the notation $\sigma_1^{t_1} \cdot \sigma_2^{t_2} \cdots \sigma_k^{t_k}$ for a signal of length $t_1 + \cdots + t_k$ whose value is σ_1 at the interval $[0, t_1)$, σ_2 in the interval $[t_1, t_1 + t_2)$, etc. We use $t \oplus [a, b]$ to denote $[t + a, t + b] \cap [0, r)$ and $t \ominus [a, b]$ for $[t - b, t - a) \cap [0, r)$, that is, the Minkowski sum (difference) of $\{t\}$ and $[a, b]$ restricted to the domain of definition of the signal in question. We call these operations, respectively, forward and backward shifting.

We define the logic MITL$_{[a,b]}$ as a bounded version of the real-time temporal logic MITL [AFH96], such that all temporal modalities are restricted to intervals of the form $[a, b]$ with $0 \leq a < b$ and $a, b \in \mathbb{N}$. The use of bounded temporal properties is one way to interpret temporal logic over finite-duration traces. The basic formulae of MITL$_{[a,b]}$ are defined by the grammar

$$\varphi := p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \mathcal{S}_{[a,b]} \varphi_2 \mid \varphi_1 \mathcal{U}_{[a,b]} \varphi_2$$

where p belongs to a set $P = \{p_1, \ldots, p_n\}$ of propositions corresponding naturally to the coordinates of the n-dimensional Boolean signal considered. The future and past fragments of MITL use only the $\mathcal{U}$ and $\mathcal{S}$ modalities, respectively. The satisfaction relation $(\xi, t) \models \varphi$, indicating that signal ξ satisfies φ at position t, is defined inductively below. We use $p[t]$ to denote the projection of $\xi[t]$ on the dimension that corresponds to variable p.

$$\begin{array}{ll}
(\xi, t) \models p & \leftrightarrow t \in [0, r) \wedge p[t] = \text{T} \\
(\xi, t) \models \neg\varphi & \leftrightarrow (\xi, t) \not\models \varphi \\
(\xi, t) \models \varphi_1 \vee \varphi_2 & \leftrightarrow (\xi, t) \models \varphi_1 \text{ or } (\xi, t) \models \varphi_2
\end{array}$$

$$(\xi, t) \models \varphi_1 \mathcal{S}_{[a,b]} \varphi_2 \leftrightarrow \exists t' \in t \ominus [a, b]\ (\xi, t') \models \varphi_2 \text{ and } \forall t'' \in [t', t], (\xi, t'') \models \varphi_1$$

$$(\xi, t) \models \varphi_1 \mathcal{U}_{[a,b]} \varphi_2 \leftrightarrow \exists\, t' \in t \oplus [a, b]\ (\xi, t') \models \varphi_2 \text{ and } \forall t'' \in [t, t'], (\xi, t'') \models \varphi_1$$

The satisfaction of a formula φ by the whole signal ξ is defined differently for the past and future fragments. For the past it is defined backwards as[7] $(\xi, |\xi|) \models \varphi$, and for the future as $(\xi, 0) \models \varphi$.

From basic MITL$_{[a,b]}$ operators one can derive other standard Boolean and temporal operators, in particular the time-constrained *sometime in the past*, *always in the past*, *eventually in the future* and *always in the future* operators whose semantics is defined as

$$\begin{array}{l}
(\xi, t) \models \diamondminus_{[a,b]} \varphi \leftrightarrow \exists t' \in t \ominus [a, b]\ (\xi, t') \models \varphi \\
(\xi, t) \models \boxminus_{[a,b]} \varphi \leftrightarrow \forall t' \in t \ominus [a, b]\ (\xi, t') \models \varphi \\
(\xi, t) \models \diamondplus_{[a,b]} \varphi \leftrightarrow \exists t' \in t \oplus [a, b]\ (\xi, t') \models \varphi \\
(\xi, t) \models \boxplus_{[a,b]} \varphi \leftrightarrow \forall t' \in t \oplus [a, b]\ (\xi, t') \models \varphi
\end{array}$$

Note that our definition of the semantics of the time-bounded *since* and *until* operators differs slightly from their conventional definition in discrete time as it requires a time instant t' where *both* $(\xi, t') \models \varphi_2$ and $(\xi, t') \models \varphi_1$.

[7] To be more precise, it is the right limit of $(\xi, t) \models \varphi$ at $t \to r$.

3 Timed Automata

We use a variant of TA which differs from the classical definitions [AD94, HNSY94] by the following features:

1. It reads multi-dimensional *dense* Boolean signals, hence the alphabet letters are associated with *states* rather than with *transitions*.
2. Acceptance conditions are more refined and may include constraints on clock values.
3. Clock values may include the special symbol $\perp$ indicating that the clock is currently *inactive*.
4. Transitions can be labeled by the usual resets of the form $x := 0$ or $x := \perp$ as well as by *copy assignments* of the form $x_i := x_j$.

The last three features do not change the expressive power of timed automata, see [SV96], but allow us to treat clocks in a more "dynamic" fashion. Note that clock inactivity in a state can be encoded implicitly by the fact that in all paths emanating from the state, the clock is reset to zero before being tested [DY96]. The use of signals is motivated by our application domain and replicating our results to event-based semantics is left as an exercise.

The set of valuations of a set $\mathcal{C} = \{x_1, \ldots, x_n\}$ of clock variables, each denoted as $v = (v_1, \ldots, v_n)$, defines the clock space $\mathcal{H} = (\mathbb{R}_{\geq 0} \cup \{\perp\})^n$. A *configuration* of a timed automaton is a pair of the form (q, v) with q being a discrete state. For a clock valuation $v = (v_1, \ldots, v_n)$, $v + t$ is the valuation $(v'_1, \ldots, v'_n)$ such that $v'_i = v_i$ if $v_i = \perp$ and $v'_i = v_i + t$ otherwise. A *clock constraint* is a Boolean combination of conditions of the forms $x \geq d$ or $x > d$ for some integer d.

Definition 1 (Timed Automaton). *A timed automaton over signals is a tuple $\mathcal{A} = (\Sigma, Q, \mathcal{C}, \lambda, I, \Delta, q_0, F)$ where Σ is the input alphabet ($\mathbb{B}^n$ in this paper), Q is a finite set of discrete states and $\mathcal{C}$ is a set of clock variables. The labeling function $\lambda : Q \to \Sigma$ associates a letter of the alphabet to every state while the staying condition (invariant) I assigns to every state q a subset I_q of $\mathcal{H}$ defined by a conjunction of inequalities of the form $x \leq d$, for some clock x and integer d. The transition relation Δ consists of elements of the form (q, g, ρ, q') where q and q' are discrete states, the transition guard g is a subset of $\mathcal{H}$ defined by a clock constraint and ρ is the update function, a transformation of $\mathcal{H}$ defined by a set of copy assignments and resets on $\mathcal{C}$. Finally q_0 is the initial state and F is the acceptance condition, a subset of $Q \times \mathcal{H}$ defined for each state by a clock constraint.*

The behavior of the automaton as it reads a signal ξ consists of an alternation between time progress periods where the automaton stays in a state q as long as $\xi[t] = \lambda(q)$ and I_q holds , and discrete instantaneous transitions guarded by clock conditions. Formally, a *step* of the automaton is one of the following:

- A time step: $(q, v) \xrightarrow{\sigma^t} (q, v + t)$, $t \in \mathbb{R}_+$ such that $\lambda(q) = \sigma$ and $v + t$ satisfies I_q (due to the structure of I_q this holds as well for every t', $0 \leq t' < t$).
- A discrete step: $(q, v) \xrightarrow{\delta} (q', v')$, for some transition $\delta = (q, g, \rho, q') \in \Delta$, such that v satisfies g and $v' = \rho(v)$.

A *run* of the automaton starting from a configuration (q_0, v_0) is a finite sequence of alternating time and discrete steps of the form

$$\xi: \quad (q_0, v_0) \xrightarrow{\sigma_1^{t_1}} (q_0, v_0 + t_1) \xrightarrow{\delta_1} (q_1, v_1) \xrightarrow{\sigma_2^{t_2}} (q_1, v_1 + t_2) \xrightarrow{\delta_2} \cdots \xrightarrow{\sigma_n^{t_n}} (q_f, v_f)$$

A run is accepting if the last configuration $(q_f, v_f) \in F$. The signal carried by the run is $\sigma_1^{t_1} \cdot \sigma_2^{t_2} \cdots \sigma_n^{t_n}$. The language of the automaton consists of all signals carried by accepting runs.

A timed automaton is *input-deterministic* if every input signal admits a unique run, a property guaranteed by the following two conditions:

1. Transition determinism: for every two transitions (q, g_1, ρ_1, q_1) and (q, g_2, ρ_2, q_2), $\lambda(q_1) = \lambda(q_2)$ implies $g_1 \cap g_2 = \emptyset$.
2. Time determinism: for every state q and transition (q, g, ρ, q'), if $\lambda(q) = \lambda(q')$ then the interior of $I_q \cap g$ is empty.

These two conditions imply that while reading a given signal, the automaton cannot be in two or more configurations simultaneously for any positive-length duration.

4 From Past MITL$_{[a,b]}$ to Deterministic Timed Automata

In this section we show how to build a deterministic timed automaton for any past MITL$_{[a,b]}$ formula. The construction follows the same lines as the compositional construction of [Pnu03] for untimed future temporal logic, where an automaton for a formula observes the states of the automata that correspond to its sub-formulae. This construction is particularly attractive for past temporal logic where the correspondence between states in the automaton and satisfaction of a sub formula is more direct.

We illustrate the idea underlying the proof on the formula $\diamondsuit_{[a,b]}\, \varphi$ for some past formula φ. Intuitively, an automaton that accepts such a language should monitor the truth value of φ and memorize, using clocks, the times when this value has changed. Memorizing all such changes may require an unbounded number of clocks, but as we shall see, only a finite number of those is sufficient since not all occurrence times of these changes need to be remembered.

Consider signal φ of Figure 1-(a), a clock x_i reset to zero at the i^{th} time φ becomes true and a clock y_i reset when φ becomes false. For this example $\diamondsuit_{[a,b]}\, \varphi$ is true exactly when $(x_1 \geq a \wedge y_1 \leq b) \vee (x_2 \geq a \wedge y_2 \leq b)$. Due to the monotonicity of the clock dynamics, whenever y_1 goes beyond b, its value becomes irrelevant for the satisfaction of the acceptance condition, it can be discarded together with x_1. By itself, this fact does not guarantee finiteness of the number of clocks because we assume no a-priori bound on the variability of φ.

Consider now Figure 1-(b), where the second rise of φ is less than $b-a$ time after the preceding fall. In this case, condition $(x_1 \geq a \wedge y_1 \leq b) \vee (x_2 \geq a \wedge y_2 \leq b)$ becomes equivalent to $x_1 \geq a \wedge y_2 \leq b$. Since the values of y_1 and x_2 do not matter anymore we may disactivate them and forget this short episode of $\neg\varphi$. When φ falls again we may re-use clock y_1 to record the occurrence time and let the acceptance condition be $x_1 \geq a \wedge y_1 \leq b$. Hence the maximal number of events to be remembered before the

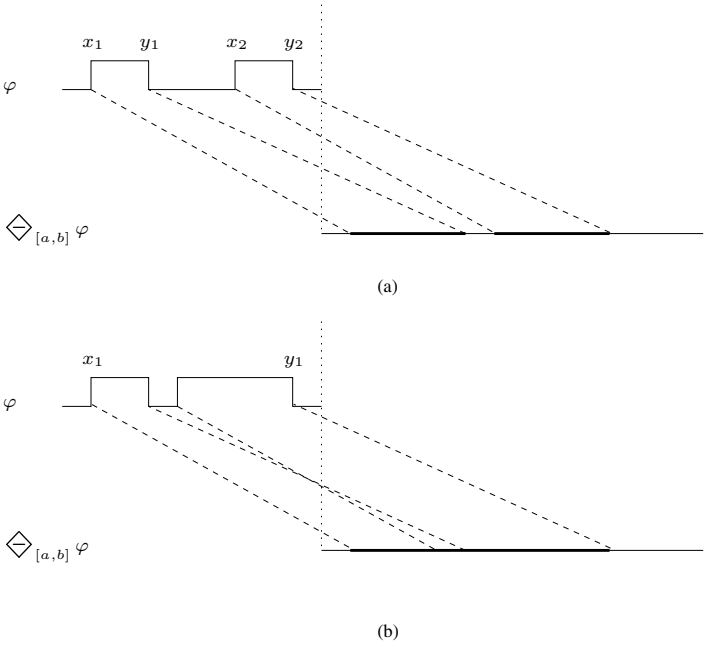

Fig. 1. Memorizing changes in the truth value of φ: (a) $x_2 - y_1 \geq b - a$; (b) $x_2 - y_1 < b - a$

oldest among them expires is $m = b/(b-a) - 1$ and at most $2m$ clocks are sufficient for monitoring such a formula. Note that for a "punctual" modality where $a = b$, m goes to infinity.

The automaton depicted in Figure 2, is a kind of an *"event recorder"* for accepting signals satisfying $\Diamond_{[a,b]}\,\varphi$. Its set of discrete states Q is partitioned into

$$Q_{\neg\varphi} = \{(01)^i 0\}_{i=0..m} \text{ and } Q_{\varphi} = \{(01)^i\}_{i=1..m},$$

with the intended meaning that the Boolean sequences that encode states correspond to the qualitative histories that they memorize, that is, the patterns of remembered rising and falling of φ that have occurred less than b time ago. The clocks of the automaton are $\{x_1, y_1, \ldots, x_m, y_m\}$, each measuring the time since its corresponding event. Naturally, clock x_i is active only at states $(01)^j$ and $(01)^j 0$ for $j \geq i$ and clock y_i at $(01)^j 0$ $(01)^{j+1}$ for $j \geq i$.

When φ first occurs the automaton moves from 0 to 01 and resets x_1. When φ becomes false it moves to 010 while resetting y_1. From there the following three continuations are possible:

Table 1. The effect of the clock shifting operation while taking a transition from $(01)^i$ to $(01)^{i-1}$

c	x_1	y_1	$\cdots$	x_{i-1}	y_{i-1}	x_i	y_i	$\cdots$	x_m	y_m
v	u_1	v_1	$\cdots$	u_{i-1}	v_{i-1}	u_i	$\perp$	$\cdots$	$\perp$	$\perp$
$s(v)$	u_2	v_2	$\cdots$	u_i	$\perp$	$\perp$	$\perp$	$\cdots$	$\perp$	$\perp$

1. If φ remains false for more than b time, the true episode of φ can be forgotten and the automaton moves to 0.
2. If φ becomes true within less than $b-a$ time, the false episode is forgotten and the automaton returns to 01.
3. If φ becomes true after more than $b-a$ time the automaton resets x_2 and moves to 0101.

Transitions of type 1 may happen in all states that record 2 changes or more. They occur when the first falling of φ is more than b time old and hence the values of clocks x_1 and y_1 can be forgotten. In order to keep the number of clocks bounded, this transition is accompanied by "shifting" the clocks values, that is, applying the operations $x_i := x_{i+1}$ and $y_i := y_{i+1}$ for all i as well as $x_m := y_m := \bot$. The effect of this shifting operation when a transition from $(01)^i$ to $(01)^{i-1}$ is taken is illustrated in Table 1.

Lemma 1. *The event recorder automaton, running in parallel with the automaton $\mathcal{A}_\varphi$, accepts the signals satisfying $\diamondsuit\!\!\!-_{[a,b]}\,\varphi$ whenever x_1 is active and satisfies $x_1 \geq a$.*

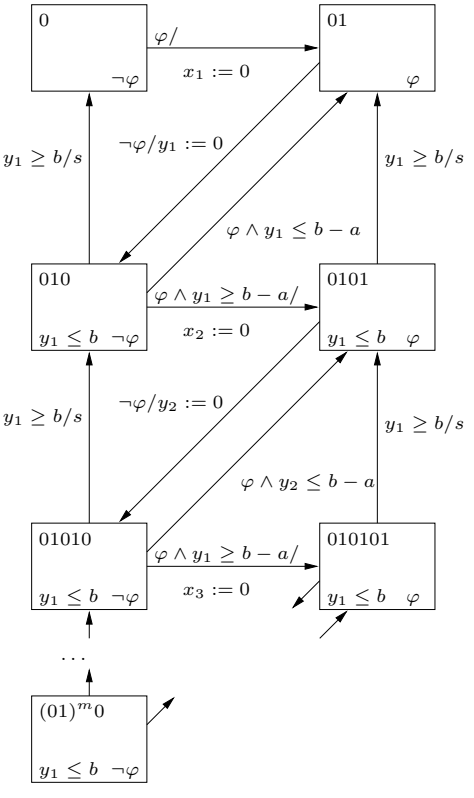

Fig. 2. An $[a,b]$ event recorder. The input labels and staying conditions are written on the bottom of each state. Transitions are decorated by the input labels of the target states and by clock resets. The clock shift operator is denoted by the symbol s.

Sketch of Proof: We need to show that in every state of the form $(01)^i 0$ there have been i risings and fallings of φ that have occurred less than b time ago such that each falling has lasted for more than $b - a$ time, and that the corresponding clocks represent the times elapsed since they have occurred. When this is the case and since $y_1 \leq b$ by construction, $x_1 \geq a$ at time t iff there was a time $t' \in t \ominus [a, b]$ in which φ was true. The proof is by induction on the length of the run. The claim is trivially true at the initial state. The inductive step starts with a configuration of the automaton satisfying the above, and proceeds by showing that it is preserved under time passage and transitions. The proof for states of the form $(01)^i$ is similar. ⌟

Lemma 2. *Given deterministic timed automata $\mathcal{A}_\varphi$ and $\mathcal{A}_\psi$ accepting $[\![\varphi]\!]$ and $[\![\psi]\!]$, respectively, one can construct a deterministic timed automaton accepting $\varphi \mathcal{S}_{[a,b]} \psi$.*

Proof: Observe first that $\varphi \mathcal{S} \psi$ can be seen as a restriction of $\diamondsuit\!\!\!\!-\ \psi$ to periods where φ holds *continuously*. In other words, the automaton need not measure times of changes in ψ after which φ became false. Hence the $\mathcal{S}$-automaton (Figure 3) consists of an event recorder for ψ augmented with an additional initial state $\neg\varphi$. Whenever φ becomes true the automaton moves to the initial state of the event recorder and whenever φ becomes false it moves (from any state) back to $\neg\varphi$ while forgetting all the past history of ψ. ⌟

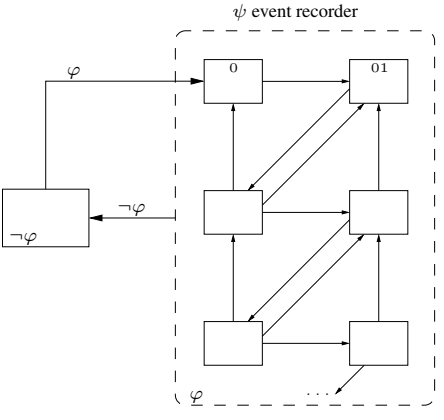

Fig. 3. The automaton for $\varphi \mathcal{S}_{[a,b]} \psi$

Theorem 1 (Past MITL is Deterministic). *Given a past* $\text{MITL}_{[a,b]}$ *formula φ, one can construct a deterministic timed automaton $\mathcal{A}$ accepting $[\![\varphi]\!]$.*

Proof: By induction on the structure of the formula. For a proposition p we build the deterministic two-state automaton $\mathcal{A}_p$ which moves to and from the accepting state according to the current value of p. For $\neg\varphi$ we take the automaton $\mathcal{A}_\varphi$ and complement its acceptance condition while for $\varphi \vee \psi$ we do a Cartesian product of $\mathcal{A}_\varphi$ and $\mathcal{A}_\psi$. Combining this with the previous lemma the result is established. ⌟

5 Future MITL is Non-deterministic

In this section we demonstrate the existence of a timed language L, definable in future MITL, which cannot be accepted by any deterministic automaton. Consider the formula

$$\Box_{[0,a]}(p \Rightarrow \Diamond_{[a,b]} q). \tag{1}$$

and the language L consisting of all signals of length $a+b$ that satisfy it. Models of this formula are two-dimensional Boolean signals that satisfy some relation between the times p is true in the interval $[0, a]$ and times when q holds in $[a, a + b]$ (see Figure 4). An automaton for L reads first the p part and memorizes what is required to memorize in order to determine whether the q part is accepted.

The syntactic (Nerode) right-congruence $\sim$ associated with a language L is defined as:

$$u \sim v \text{ iff } \forall w \; u \cdot w \in L \Leftrightarrow v \cdot w \in L.$$

For untimed languages acceptance by a finite (deterministic) automaton is equivalent to $\sim$ having a finite number of equivalence classes. In [MP04] we have shown that for timed languages, finiteness can be replaced by some special kind of *boundedness* which, among other things, implies:

Proposition 1 (MP04). *If a language is accepted by a deterministic timed automaton then there is some n such that all signals with n changes are Nerode equivalent to signals with less than n changes.*[8]

We now show that this is not the case for L, for which only signals which are identical are equivalent.

Claim. Let u and v be two p-signals of length a. Then, $u \neq v$ implies $u \not\sim v$ with respect to L.

Proof: Let t be the first time when u and v differ. Assume that p is true on $[t, t + \varepsilon]$ in u and false on that interval in v. We can then construct a distinguishing signal w such that $uw \notin L$ and $vw \in L$. Let w be the q-signal $1^t \cdot 0^{b-a+\varepsilon} \cdot 1^{a-t-\varepsilon}$, i.e. a signal which is true throughout $[a, a + b]$ except for the interval $[t + a, t + b + \varepsilon]$ (see Figure 5). Clearly uw will be rejected due to unfulfilled eventuality in the interval while vw will be accepted because v generates no obligations for this interval which are not fulfilled by the true values of w on both sides of the interval. ▟

Hence, while reading the p-part the automaton should memorize the exact form of the signal, and since its variability is not bounded, an unbounded number of clocks is needed to memorize the times when p changes.

Corollary 1 (Future MITL is not Deterministic). *There are languages expressible in future* MITL *which cannot be recognized by any deterministic timed automaton.*

[8] Note that the converse is not true: consider, for instance, the language consisting of all signals where p holds continuously, and whose duration is a prime number. All signals with one or more changes in p value are rejected and hence equivalent, yet the language cannot be accepted by a deterministic (or non-deterministic) timed automaton.

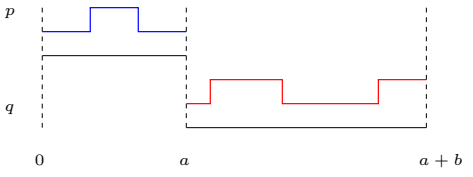

Fig. 4. A candidate signal for satisfying the formula; values of p and q are specified only in the relevant intervals, $[0, a]$ for p and $[a, a + b]$ for q

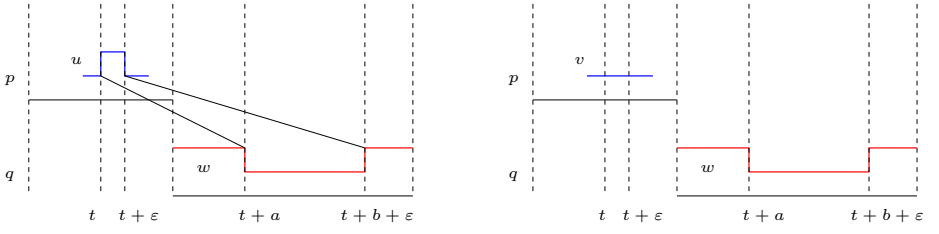

Fig. 5. Signal u, v and w such that $u \cdot w \notin L$ and $v \cdot w \in L$

This raises an intriguing question: why for specifications expressed in past MITL, the automaton *can* forget certain small changes in p that persist less than $b - a$ time? Below we give an answer to this question.

Consider first a "punctual" version of the future formula (1), where q should follow *exactly* b time after p:

$$\Box_{[0,a]}(p \Rightarrow \Diamond_b q)$$

This formula admits a "dual" past formula

$$\boxminus_{[0,a]}(\neg q \Rightarrow \boxminus_b \neg p)$$

which is semantically equivalent on signals of length $a + b$. In other words, the first-order interpretations

$$\forall t \in [0, a]\ p[t] \Rightarrow q[t + b]$$

and

$$\forall t' \in [b, b + a]\ \neg q[t'] \Rightarrow \neg p[t' - b]$$

are equivalent (see Figure 6).

However, when we relax punctuality and use interval time modalities, the symmetry between past and future is broken and the automaton for the corresponding past formula

$$\boxminus_{[0,a]}(\neg q \Rightarrow \diamondminus_{[a,b]} \neg p) \tag{2}$$

can ignore short episodes. The reason is due to the inter-relationship between the direction of the implication and the Minkowski sum. In a future interval modality, an event that happens at t may create an obligation for something to hold somewhere or throughout the future interval $t \oplus [a, b] = [t + a, t + b]$. In the past modality a future

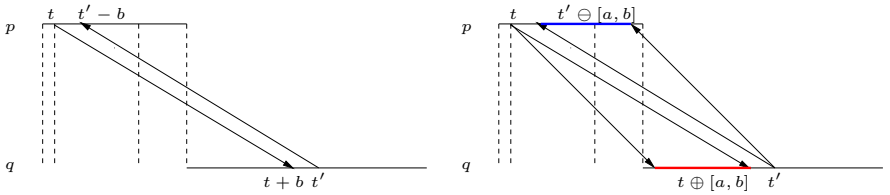

Fig. 6. Punctual modalities are past-future symmetric; interval modalities are not

event at time t' is implied by something that could/should have happened at the interval $t' \ominus [a,b] = [t'-b, t'-a]$. Anything that lasts less then $b-a$ does not generate its own specific obligations (obligations that are not already generated by neighboring segments of the signal). Logically speaking, (2) translates into the the first-order formula

$$\forall t' \in [b, a+b]\ (\neg q[t'] \Rightarrow \exists t \in t' \ominus [a,b]\ \neg p[t])$$

or equivalently

$$\forall t' \in [b, a+b]\ ((\forall t \in t' \ominus [a,b]\ p[t]) \Rightarrow q[t']).$$

Consider now two p-signals $p_1 = \xi \cdot 0^{t_1} 1^{t_2} 0^{t_3}$ and $p_2 = \xi \cdot 0^{t_1+t_2+t_3}$ that differ only by the true episode of length $t_2 < b-a$ in p_1. It is not hard to see that for any $t' \in [b, a+b]$

$$\forall t \in t' \ominus [a,b]\ p_1[t] \iff \forall t \in t' \ominus [a,b]\ p_2[t]$$

because any $t' \ominus [a,b]$ that intersects the true segment 1^{t_2} in p_2 also intersects at least one of its neighboring false segments. Hence the same obligations for t' are generated by p_1 and p_2 and they are Nerode equivalent.

So in conclusion, the difference between past and future in real-time temporal logic turns out to be due to a syntactic artifact that generates some bounded variability "filtering" for past interval modalities, but not for the future ones.

6 Discussion

It seems that the current paper does not conclude the search for a specification formalism which is natural, powerful and yet determinizable. Past MITL lacks the first property and future MITL is not deterministic. As another candidate we have explored a star-free version of the timed regular expressions presented in [ACM02]. The concatenation operator is more symmetric then the *since* and *until* operators and it could be interesting to see how it behaves. However it turns out that variations on both the future (1) and past (2) formulae can be defined by expressions such as

$$\neg(U \cdot (p \cdot U \wedge \neg(\langle U \cdot q \rangle_{[a,b]} \cdot U)))$$

and

$$\neg(U \cdot (U \cdot p \wedge \neg(U \cdot \langle q \cdot U \rangle_{[a,b]})) \cdot U),$$

respectively, where U is a special symbol denoting the universal timed language. This shows the star-free expressions are not deterministic either.

It looks as if determinizability can be obtained by enforcing some minimal duration for sub-formulae that imply something toward the future, for example p in (1). In past MITL this is automatically guaranteed by the structure of the formulae. We are currently contemplating sufficient syntactic conditions that will guarantee a similar property for the future fragment. In this context it is worth mentioning the *inertial bi-bounded delay* operator used for expressing delays in abstract models of digital circuits, which was formalized using timed automata in [MP95a]. This operator allows one to relate two signals by an inclusion of the form

$$\xi' \in D^c_{[a,b]}(\xi)$$

where $c \leq a$, meaning that every change in ξ that *persists for at least* c *time*, is propagated to ξ' within $t \in [a, b]$ time. Observe that this type of persistence constraint can be expressed within MITL. For example, the future formula (1) can be transformed into

$$\Box_{[0,a]}(\Box_{[0,b-a]}p \Rightarrow \Diamond_{[a,b]}q)$$

which is determinizable. Further investigations of these issues belong to the future.

Acknowledgments. This work benefited from discussions with E. Asarin, S. Tripakis and Y. Lakhnech and from comments made by D. Fisman, T. Henzinger and anonymous referees.

References

[Alu99] R. Alur, Timed Automata, *Proc. CAV'99*, LNCS 1633, 8-22, Springer, 1999.

[AD94] R. Alur and D.L. Dill, A Theory of Timed Automata, *Theoretical Computer Science* 126, 183–235, 1994.

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

[AFH99] R. Alur, L. Fix, and T.A. Henzinger, Event-Clock Automata: A Determinizable Class of Timed Automata, *Theoretical Computer Science* 211, 253-273, 1999.

[AH92a] R. Alur and T.A. Henzinger. Logics and Models of Real-Time: A Survey. *Proc. REX Workshop, Real-time: Theory in Practice*, pages 74–106. LNCS 600, Springer, 1992.

[AH92b] R. Alur and T.A. Henzinger, Back to the Future: Towards a Theory of Timed Regular Languages, *Proc. FOCS'92*, 177-186, 1992.

[Asa04] E. Asarin, Challenges in Timed Languages, *Bulletin of EATCS* 83, 2004.

[ACM02] E. Asarin, P. Caspi and O. Maler, Timed Regular Expressions *The Journal of the ACM* 49, 172-206, 2002.

[DY96] C. Daws and S. Yovine, Reducing the Number of Clock Variables of Timed Automata, *Proc. RTSS'96*, 73-81, IEEE, 1996.

[EFH+03] C. Eisner, D. Fisman, J. Havlicek, Y. Lustig, A. McIsaac, and D. Van Campenhout, Reasoning with Temporal Logic on Truncated Paths, *Proc. CAV'03*, 27-39. LNCS 2725, Springer, 2003.

[Gei02] M.C.W. Geilen. *Formal Techniques for Verification of Complex Real-time Systems*. PhD thesis, Eindhoven University of Technology, 2002.

[Hen98] T.A. Henzinger. It's about Time: Real-time Logics Reviewed. In *Proc. CONCUR'98*, pages 439–454. LNCS 1466, Springer, 1998.

[HR02] K. Havelund and G. Rosu. Synthesizing Monitors for Safety Properties. In *Proc. TACAS'02*, pages 342–356. LNCS 2280, Springer, 2002.

[HNSY94] T. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine, Symbolic Model-checking for Real-time Systems, *Information and Computation* 111, 193–244, 1994.

[HR04] Y. Hirshfeld and A. Rabinovich Logics for Real Time: Decidability and Complexity, *Fundamenta Informaticae* 62, 1-28, 2004.

[Koy90] R. Koymans, Specifying Real-time Properties with with Metric Temporal Logic, *Real-time Systems*, pages 255-299. 1990.

[KPA03] K.J. Kristoffersen, C. Pedersen, and H.R. Andersen. Runtime Verification of Timed LTL using Disjunctive Normalized Equation Systems. In *Proc. RV'03*. ENTCS 89(2), 2003.

[KT04] M. Krichen and S. Tripakis. Black-box Conformance Testing for Real-time Systems. In *Proc. SPIN'04*, pages 109–126. LNCS 2989, Springer, 2004.

[LPZ85] O. Lichtenstein, A. Pnueli and L.D. Zuck, The Glory of the Past, *Proc. Conf. on Logic of Programs*, 196-218, LNCS, 1985.

[LW05] S. Lasota and L. Walukiewicz, Alternating Timed Automata, In *Proc. FOSSACS'05*, pages 250–265. LNCS 3441, Springer.

[MN04] O. Maler and D. Nickovic, Monitoring Temporal Properties of Continuous Signals, *proc. FORMATS/FTRTFT'04*, LNCS 3253, 2004.

[MP90] O. Maler and A. Pnueli, Tight Bounds on the Complexity of Cascaded Decomposition of Automata. *proc. FOCS'90*, 672-682, 1990.

[MP95a] O. Maler and A. Pnueli, Timing Analysis of Asynchronous Circuits using Timed Automata, in *Proc. CHARME'95*, LNCS 987, 189-205, Springer, 1995.

[MP04] O. Maler and A. Pnueli. On Recognizable Timed Languages. In *Proc. FOSSACS'04*, pages 348–362. LNCS 2987, Springer, 2004.

[MP95b] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer, 1995.

[MNP71] R. McNaughton and S. Papert, *Counter Free Automata*, MIT Press, Cambridge, MA, 1971.

[NP86] M. Nivat and D. Perrin, Ensembles Reconnaissables de Mots Bi-infinis, *Canadian J. of Mathematics* 38, 513-537, 1986.

[OW05] J. Ouaknine and J. Worrell, On the Decidability of Metric Temporal Logic, In *Proc. LICS'05*, (to appear), 2005.

[Pnu03] A. Pnueli. Verification of Reactive Systems. Lecture Notes, NYU, 2003. http://cs.nyu.edu/courses/fall03/G22.3033-007/lecture4.pdf.

[RSH98] J.-F. Raskin, P.Y. Schobbens and T.A. Henzinger, Axioms for Real-Time Logics, *Proc. Concur'98*, 219-236, 1998.

[Saf88] S. Safra, On the Complexity of ω-automata, *Proc. FOCS'88*, 319-327, 1998.

[SV96] J.G. Springintveld and F.W. Vaandrager, Minimizable Timed Automata, *Proc. FTRTFT'96*, 130-147, LNCS 1135, Springer, 1996.

[TR04] P. Thati and G. Rosu. Monitoring Algorithms for Metric Temporal Logic Specifications. In *Proc. of RV'04*, 2004.

[Tri02] S. Tripakis. Fault Diagnosis for Timed Automata. In *Proc. FTRTFT'02*, pages 205–224. LNCS 2469, Springer, 2002.

[VW86] M.Y. Vardi and P. Wolper. An Automata-theoretic Approach to Automatic Program Verification. In *Proc. LICS'86*, pages 322–331. IEEE, 1986.