From MITL to Timed Automata*

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. We show how to transform formulae written in the real-time temporal logic MITL into timed automata that recognize their satisfying models. This compositional construction is much simpler than previously known and can be easily implemented.

Prediction is very difficult, especially about the future.
Niels Bohr

1 Introduction

Decision procedures for model-checking of temporal logic formulae [MP91,MP95] play a central role in algorithmic verification [CGP99]. Such procedures are based, in the linear-time context, on deriving from a formula φ an automaton-like device that accepts exactly sequences of states or events that satisfy it [VW86]. For discrete-time models, used for functional verification of software or synchronous hardware, the logical situation is rather mature. Logics like LTL (linear-time temporal logic) or CTL (computation-tree logic) are commonly accepted and incorporated into verification tools. LTL admits a variety of efficient algorithms for translating a formula into an equivalent automaton [GPVW95,SB00,GO01,KP05] and it even underlies the industrial standard PSL [KCV04,HFE04].

When considering *timed* models and specification formalisms whose semantics involves the time domain $\mathbb{R}_+$ rather than $\mathbb{N}$, the situation is somewhat less satisfactory [A04]. Many variants of real-time logics [Koy90,AH92a,Hen98,HR04] as well as timed regular expressions [ACM02] have been proposed but the correspondence between simply-defined logics and variants of timed automata (automata with auxiliary clock variables [AD94]) is not as simple and canonical as for the untimed case, partly, of course, due to the additional complexity of the timed model. Consequently, existing verification tools for timed automata rarely use temporal properties other than safety. One of the most popular dense-time extensions of LTL is the logic MITL introduced in [AFH96] as a restriction of the logic MTL [Koy90]. The principal modality of MITL is the *timed until* $\mathcal{U}_I$ where I is some non-singular interval. A formula $p\,\mathcal{U}_{[a,b]}\,q$ is

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

satisfied by a model at any time instant t that admits q at some $t' \in [t+a, t+b]$, and where p holds continuously from t to t'. The decidability of MITL was established in [AFH96] and it was, together with MTL, subject to further investigations [AH92b,RSH98,HRS98,OW05]. However, the automaton construction in [AFH96] is very complicated (11 pages) and, to the best of our knowledge, has never been implemented. The only logic that has been integrated into a real-time model-checking tool was the timed version of CTL, TCTL [HNSY94], used in the tool Kronos [Y97].[4]

In this paper we remedy this situation somewhat by presenting a simple construction of timed automata that accept exactly models of MITL formulae. The construction is based on two ideas, the first being the *modular construction* of property testers for untimed formulae [KP05] and the other is the observation, similar to the one already made in [AFH96], that the evolution over time of the satisfiability of a formula of the form $p\,\mathcal{U}_{[a,b]}\,q$ is of bounded variability, regardless of the variability of p and q.

The rest of the paper is organized as follows. In Section 2 we illustrate the modular construction of testers for (untimed) LTL formulae. The logic MITL is presented in Section 3 together with its semantic domain (Boolean signals) and timed automata. The main result, the construction of property testers for MITL, is presented in Section 4, followed by a brief discussion of the differences between the version of MITL that we use and the one presented in [AFH96].

2 Temporal Testers for LTL

In this section we discuss some of the problems associated with the construction of automata that accept models of LTL formulae, and describe the modular procedure of [KP05] which we later extend for MITL. We feel that, independently of its timed generalization, this construction, based on composition of *acausal sequential transducers*, improves our understanding of temporal logic and can serve as a unifying framework for both verification and monitoring. We assume familiarity with basic notions of LTL, whose syntax is defined according to the grammar

$$\varphi := p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \bigcirc \varphi \mid \varphi_1 \mathcal{U} \varphi_2$$

where p belongs to a set $P = \{p_1, \ldots, p_n\}$ of propositions. LTL is interpreted over n-dimensional Boolean ω-sequences of the form $\xi : \mathbb{N} \to \mathbb{B}^n$. We abuse p to denote the projection of the sequence ξ on p. The semantics of LTL formulae is typically given via a recursive definition of the relation $(\xi, t) \models \varphi$ indicating that the sequence ξ satisfies φ at position t. The satisfaction of a compound formula $op(\varphi_1, \varphi_2)$, where op is a temporal or propositional operator, by a sequence ξ is an op-dependent function of the satisfaction of the sub formulae φ_1 and φ_2 by ξ. Functionally speaking, the satisfaction of φ by arbitrary sequences can be viewed as a *characteristic function* χ^φ which maps sequences over $\mathbb{B}^n$ into Boolean sequences such that $\beta = \chi^\varphi(\xi)$ means that for every i, $\beta[i] = 1$ iff $(\xi, i) \models \varphi$. The semantics of LTL can thus be formulated[5] as a recursive

[4] An efficient emptiness checking algorithm for timed Büchi automata has been proposed and implemented in [TYB05] but without an upstream translation from a logical formalism.

[5] Throughout the paper we use a variant of *until* in which $p\,\mathcal{U}\,q$ requires that p holds also at the same time instant when q becomes true, which can be expressed as $p\mathcal{U}(p \wedge q)$ using the

definition of χ^{φ}:

$$\begin{array}{lcl} \chi^{p}[t] & = & p[t] \\ \chi^{\neg\varphi}[t] & = & \neg\chi^{\varphi}[t] \\ \chi^{\varphi_1 \vee \varphi_2}[t] & = & \chi^{\varphi_1}[t] \vee \chi^{\varphi_2}[t] \\ \chi^{\bigcirc\varphi}[t] & = & \chi^{\varphi}[t+1] \\ \chi^{\varphi_1 \mathcal{U} \varphi_2}[t] & = & \bigvee_{t' \geq t} (\chi^{\varphi_2}[t'] \wedge \bigwedge_{t'' \in [t,t']} \chi^{\varphi_1}[t'']) \end{array} \tag{1}$$

Given a formula φ, its characteristic function is defined as a composition of sequential functions, following the pattern of its parse tree, as illustrated in Figure 1. So what remains to be done is to build an automaton that realizes the appropriate sequential function for each LTL operator and use these building blocks, that we call *temporal testers*, to construct a mechanism that computes the characteristic function for arbitrary formulae, but some problem related to causality should be settled first.

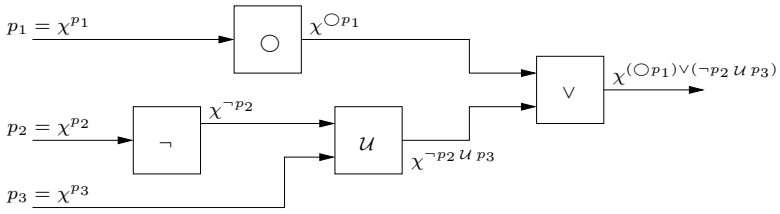

Fig. 1. Computing the characteristic function of $(\bigcirc p_1) \vee (\neg p_2 \,\mathcal{U}\, p_3)$.

A sequential function $f : A^{\omega} \to B^{\omega}$ is said to be *causal* if for every $u \in A^*$, $v, v' \in B^*$ such that $|u| = |v| = |v'|$ and every $\alpha \in A^{\omega}$ and $\beta \in B^{\omega}$

$$f(u \cdot \alpha) = v \cdot \beta \text{ and } f(u \cdot \alpha') = v' \cdot \beta' \text{ implies } v = v'.$$

In other words, the value of $f(\alpha)$ at time t may depend only on the values $\{\alpha[t'] : t' \leq t\}$. Causal functions are realized naturally by deterministic automata with output (sequential synchronous transducers). However, unlike the semantics of the *past* fragment of LTL which is causal (see [HR02]), the semantics of future LTL is not. Looking closely at (1) we can see that while the propositional operators define causal sequential functions, the future temporal operators are acausal. The output of the *next* operator at time t depends on the input at $t + 1$ and, even worse, the output of the *until* operator at t may depend on input values at some t' which may be arbitrarily further in the future.

One can think of two ways to realize acausal sequential functions. The first approach, which works well for operators with a *bounded* level of acausality, such as the *next* operator or several nestings of which (denoted by $\bigcirc^d$) is to dissociate the time scales of the input and the output. That is, let the automaton ignore the first d input

standard interpretation. For LTL this variation just simplifies the corresponding tester while for MITL it avoids certain anomalies.

symbols, and then let $\beta[t] = \alpha[t+d]$. Unfortunately this does not always work for unbounded acausality.

Using the second approach the transducer responds to the input *synchronously* but since at time t the information might not be sufficient for determining the output, it will have to "guess" the output non-deterministically and split the computation into two runs, one that predicts 0 and one that predicts 1. Each of the runs needs to remember its predictions until sufficient input information is accumulated to abort all but one run. The automaton for operators with acausality of depth d, may need to memorize up to 2^d predictions. The first approach is more intuitive[6] but since we do not know how to extend it to unbounded acausality, we use the second one. The automaton that computes the characteristic function of $\bigcirc$ is depicted in Figure 2 along with a sample run.[7] State s_0 indicates that the prediction made in the previous step was 0, hence at this state, observing p contradicts the prediction and the run is aborted. Input $\overline{p}$ confirms the prediction and the run splits by generating two predictions for the next value and so on. For every infinite input sequence ξ, only *one* infinite run survives and its output is $\chi^{\bigcirc p}(\xi)$. The automaton for $\bigcirc^d p$ follows the same pattern and is nothing but an output-driven shift register of depth d (see Figure 2). States s_{00} and s_{01} of this automaton represent predictions for unsatisfiability at $t-2$ and hence admit only $\overline{p}$-labeled transitions. It is worth mentioning that these automata are output-deterministic and can be obtained by reversing the transitions in the ordinary input-driven shift registers that correspond to the past temporal operator *previously*, also known as the delay operator z^{-1}.

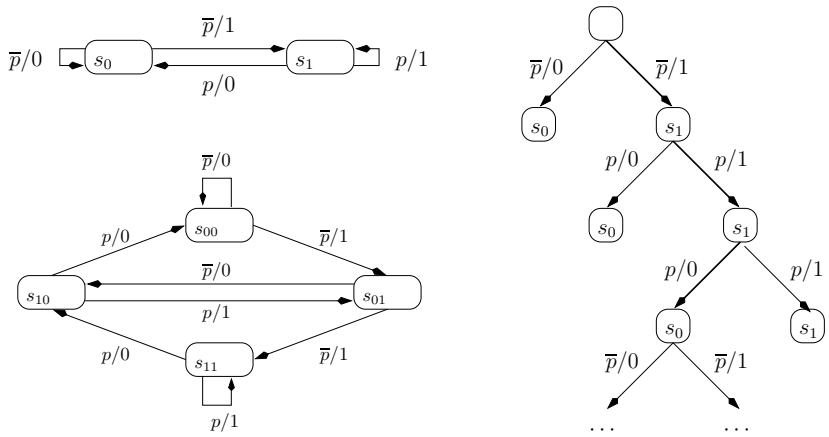

Fig. 2. The testers for $\bigcirc p$ and for $\bigcirc(\bigcirc p)$; An initial fragment of the behavior for the $\bigcirc p$-tester while computing $\chi^{\bigcirc}(\overline{p}pp\overline{p}\cdots) = 110\cdots$.

[6] After all, it is more natural to reply "I don't know yet" rather than saying "the answer now could be either 0 or 1, but only in the next step I will tell you which of them is actually true".

[7] To ease readability we use $\{p, \overline{p}\}$ instead of $\{0, 1\}$ as the input alphabet for unary operators and $\{\overline{pq}, \overline{p}q, p\overline{q}, pq\}$ for the binary ones. Moreover, we use p as a shorthand for $\{p\overline{q}, pq\}$.

The situation with $p\,\mathcal{U}\,q$ is more involved because a priori, due to the unbounded horizon, one might need to generate and memorize 2^ω predictions. However the semantics of *until* implies that at most *two* confirmable predictions may co-exist simultaneously.

Lemma 1. *Let $\beta = \chi^{p\,\mathcal{U}\,q}(\xi)$. Then for every t such that $\xi[t] = \xi[t+1] = p\overline{q}$, $\beta[t] = \beta[t+1]$.*

Proof. There are three possibilities: 1) The earliest $t' > t+1$ such that $\beta[t'] \neq p\overline{q}$ satisfies $\beta[t'] = pq$. In that case, the property is satisfied at t and $t+1$; 2) The same t' satisfies $\beta[t'] = \overline{p}$ and the property is falsified at both t and $t+1$; 3) $\beta[t'] = p\overline{q}$ for every $t' > t+1$ and the property is falsified from both time points.[8] ⌟

This fact is reflected by the tester of Figure 3. At time instants where $\overline{p}$ is observed, the value of the output is determined to be 0. Likewise when pq is observed the output is determined to be 1. The only situation that calls for non-determinism is when $p\overline{q}$ occurs and we do not know in which of the three cases of Lemma 1 we will eventually find ourselves. Hence we split the run into positive and negative predictions (states $s_{p\overline{q}}$ and $\overline{s}_{p\overline{q}}$, respectively). The only input sequences that will lead to *two* infinite runs are those ending with an infinite suffix of $p\overline{q}$'s. To choose the correct one among the two we add a Büchi condition requiring infinitely many visits in the set $\{s_{\overline{p}}, \overline{s}_{p\overline{q}}, s_{pq}\}$ which amounts to rejecting runs that stay forever in $s_{p\overline{q}}$. With these two testers (and simple testers for the Boolean operators) one can build testers for arbitrary LTL formulae. Note that in both the *next* and *until* testers, all transitions entering a state are decorated by the same output symbols so, in fact, one might view outputs as associated with states rather than with transitions, as is done in [KP05].

The notion of compositional temporal testers, as introduced in [KP05], is based on several key ideas. One of them is the allocation of an individual Boolean variable to each sub-formula of an LTL formula. This idea, to which we refer in [KP05] as *statification* of the sub-formula, has been considered first in [BCM$^+$92] in the context of symbolic implementation of a *tableau* construction. The observation that such a Boolean variable can replace the sub-formula itself in the context of model checking has been considered in [CGH94]. It is worth mentioning the translation of LTL to *alternating automata* [Var96] which, like the temporal testers approach, works inductively on the structure of the formula but not in a compositional manner.

3 Signals, their Temporal Logic and Timed Automata

Extending the construction of temporal testers from discrete to dense time requires adaptations of the semantic domain, the logic and the automata. The interaction between discrete transitions and dense time may give rise to certain anomalies and complications that we avoid by deviating slightly from the original definitions of [AFH96].

[8] This is the "strong" interpretation of *until*. For the weak version both t and $t+1$ will satisfy the property.

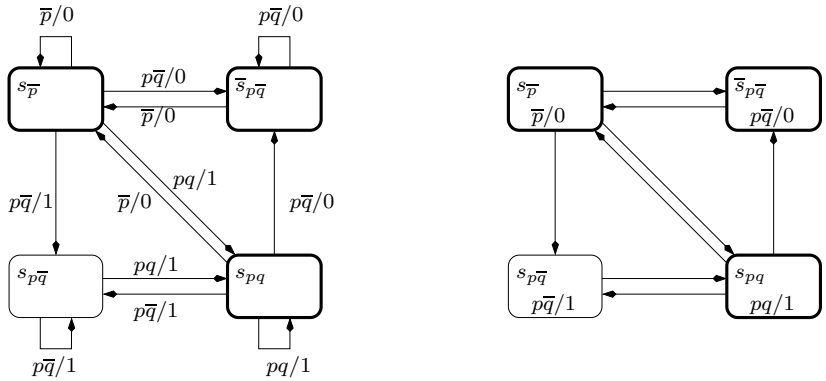

Fig. 3. The tester for $p\,\mathcal{U}\,q$ for sequences (left) and for signals (right). All states except $s_{p\overline{q}}$ are accepting. All transitions in the signal tester should be decorated by $z > 0/z := 0$, with z being an auxiliary clock, to force signals to maintain each value for a non-punctual duration.

3.1 Signals

Two basic semantic domains for describing timed behaviors have been introduced in an algebraic form in [ACM02,A04]. The first semantic objects are the *time-event sequences*, elements of the free product (shuffle) of the monoids Σ^* and $\mathbb{R}_+$. Time-event sequences consist of instantaneous events (or finite sequences of events) separated by numbers that represent time durations. The other semantic domain is that of *signals*, elements of the free product of several copies of the monoid $\mathbb{R}_+$, a copy for each alphabet symbol. Signals are thus viewed as concatenations of "typed" real numbers, where $\sigma_1^5 \cdot \sigma_2^3$ represents a σ_1 period of duration 5 followed by a σ_2 period of duration 3. Zero is the identity element of each of the monoids and it is absorbed when signals are transformed to a canonical form, that is $\sigma_1^{t_1} \cdot \sigma^0 \cdot \sigma_2^{t_2} = \sigma_1^{t_1} \cdot \sigma_2^{t_2}$. A *signal-event monoid* containing both can be defined as well [ACM02].

The transformation of these objects into the form of functions from the time domain to the alphabet is not painless. For time-event sequences a "super-dense" time [MMP92] had to be invented, a subset of the product of $\mathbb{R}_+$ and $\mathbb{N}$, where a sequence of discrete events, all taking place at time t, is mapped to the sequence $(t, 1), (t, 2), \ldots$ of generalized time instants. The timed traces of [AD94] constitute another representation of the same objects. Signals, which are the natural domain for MITL, are more well-behaving in this respect and can be mapped bijectively to functions from $\mathbb{R}_+$ to the alphabet if one accepts some restrictions. The one we adopt is to view a signal of the form $\sigma_1^{t_1} \cdot \sigma_2^{t_2}$ as a function whose value is σ_1 in the *left-closed right-open interval* $[0, t_1)$ and σ_2 in the interval $[t_1, t_1 + t_2)$. Since such intervals cannot be punctual without being empty, we exclude from the discussion signals that may have a distinct value in some isolated point.[9] Since our construction is based on characteristic functions and signal transduc-

[9] Other possibilities are left-open right-closed intervals with the exclusion of time zero, or a non-bijective representation which may map the algebraic object into two time functions that differ on t_1.

ers, we need this property of signals to be preserved by the temporal operators, hence we deviate from [AFH96] by restricting the quantitative temporal modalities to closed intervals.

Definition 1 (Signals). *Let $\mathbb{R}_+$ be the set of non-negative real numbers. A Boolean signal is a function $\xi : \mathbb{R}_+ \to \mathbb{B}^n$ such that for every $\sigma \in \mathbb{B}^n$, $\xi^{-1}(\sigma)$ is a union of* left-closed right-open *intervals.*

A partial signal is a restriction of a signal to some interval $[a, b)$. We will sometimes refer to a partial signal of the form $\sigma_1^{t_1} \cdot \sigma_2^{t_2} \cdots \sigma_k^{t_k}$ as a $\sigma_1 \cdot \sigma_2 \cdots \sigma_k$-segment of the signal.

3.2 Real-time Temporal Logic

The syntax of MITL is defined by the grammar

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

where p belongs to a set $P = \{p_1, \ldots, p_n\}$ of propositions and $b > a \geq 0$ are rational numbers (in fact, using normalization, it is sufficient to consider integers). From basic MITL operators one can derive other standard Boolean and temporal operators, in particular the time-constrained *eventually* and *always* operators

$$\Diamond_{[a,b]} \varphi = \mathrm{T}\, \mathcal{U}_{[a,b]} \varphi \quad \text{and} \quad \Box_{[a,b]} \varphi = \neg \Diamond_{[a,b]} \neg \varphi.$$

We interpret MITL over n-dimensional Boolean signals and define the satisfiability relation via characteristic functions, which for the propositional operators and the untimed *until* are defined exactly as for LTL. The semantics of timed *until* is given by

$$\chi^{\varphi_1 \mathcal{U}_{[a,b]} \varphi_2}[t] \;=\; \bigvee_{t' \in [t+a, t+b]} (\chi^{\varphi_2}[t'] \wedge \bigwedge_{t'' \in [t,t']} \chi^{\varphi_1}[t''])$$

The difference with respect to untimed *until* is that t' ranges over the bounded (but dense) interval $[t+a, t+b]$ rather than over the unbounded set $\{t, t+1, \ldots\}$ of integers. Our interpretation of the timed *until* slightly deviates from [AFH96] by requiring φ_1 to hold also at the moment t' when φ_2 becomes true, and not only in the open interval (t, t'). A signal ξ satisfies the formula φ iff $\chi^{\varphi}(\xi)[0] = 1$.

The following lemma, proved also in [DT04], shows that timed *until* can be expressed by a combination of untimed *until* and timed *eventually* whose characteristic function is

$$\chi^{\Diamond_{[a,b]} \varphi} \;=\; \bigvee_{t' \in [t+a, t+b]} \chi^{\varphi}[t'].$$

Lemma 2 (Timed Until is Redundant). *For every signal ξ,*

$$\xi \models p \mathcal{U}_{[a,b]} q \;\leftrightarrow\; \xi \models \Box_{[0,a]} (p \,\mathcal{U}\, q) \wedge \Diamond_{[a,b]} q.$$

Proof: One direction of the equivalence follows directly from the semantics of timed *until*, so we will consider only the other direction which is proved via the following observations:

1. If $\xi \models \Box_{[0,a]} p \,\mathcal{U}\, q$ then p is continuously true throughout $[0, a]$.
2. If $\xi \models \Box_{[0,a]} p \,\mathcal{U}\, q$, then $p \,\mathcal{U}\, q$ has to hold at a and hence there exists $t' \geq a$ such that q is true at t' and p holds during $[a, t']$.
3. Formula $\Diamond_{[a,b]} q$ requires the existence of $t' \in [a, b]$ such that q holds at t'

Combining these observations we can see that $\xi \models \Box_{[0,a]}(p \,\mathcal{U}\, q) \wedge \Diamond_{[a,b]} q$ implies that there exists $t' \in [a, b]$ such that q is true at t' and p holds continuously during $[0, t']$, which is exactly the semantic definition of $p \,\mathcal{U}_{[a,b]}\, q$. ⌟

For the sake of completeness, let us mention that the special case $p \,\mathcal{U}_{[0,b]}\, q$ can be written as $p \,\mathcal{U}\, q \wedge \Diamond_{[0,b]} q$, and that $p \,\mathcal{U}_{[a,\infty)}\, q$, which is not allowed by our syntax, can be written as $\Box_{[0,a]}(p \,\mathcal{U}\, q)$.

3.3 Timed Automata

We use a variant of timed automata which differs slightly from the classical definitions [AD94,HNSY94,Alu99] as it reads multi-dimensional *dense-time* Boolean signals, and outputs Boolean signals. Hence the input and output alphabet letters are associated with *states* rather than with *transitions*. We also extend the domain of clock values to include the special symbol $\perp$ indicating that the clock is currently *inactive*[10] and extend the order relation on $\mathbb{R}_+$ accordingly by letting $\perp < v$ for every $v \in \mathbb{R}_+$. For a set $A \subseteq \mathbb{R}^n$ we use $cl(A)$ to denote its closure (in the topological sense).

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 $\mathcal{H} = (\mathbb{R}_+ \cup \{\perp\})^n$. 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 conjunction of conditions of the form $x \leq d$, $x < d$, $x \geq d$ or $x > d$ for some integer d.

Definition 2 (Timed Signal Transducer). *A timed signal transducer is a tuple $\mathcal{A} = (\Sigma, Q, \Gamma, \mathcal{C}, \lambda, \gamma, I, \Delta, q_0, F)$ where Σ is the input alphabet, Q is a finite set of discrete states, Γ is the output alphabet and $\mathcal{C}$ is a set of clock variables. The input labeling function $\lambda : Q \to \Sigma$ associates an input letter to every state while the output function $\gamma : Q \to \Gamma$ assigns output letters. 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 $x \leq d$ or $x < d$, 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 an assignment of the form $x := 0$ or $x := \perp$. Finally q_0 is the initial state and $\mathcal{F} \subseteq 2^Q$ is a generalized Büchi acceptance condition.*

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:

[10] This is syntactic sugar since 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].

– A time step: $(q, v) \stackrel{\sigma^t/\tau^t}{\longrightarrow} (q, v+t)$, $t \in \mathbb{R}_+$ such that $\sigma = \lambda(q)$, $\tau = \gamma(q)$, and[11] $v + t \in cl(I_q)$.
– A discrete step: $(q, v) \stackrel{\delta}{\longrightarrow} (q', v')$, for some transition $\delta = (q, g, \rho, q') \in \Delta$, such that $v \in g$ and $v' = \rho(v)$

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

$$\xi : \quad (q_0, v_0) \stackrel{\sigma_1^{t_1}/\tau_1^{t_1}}{\longrightarrow} (q_0, v_0 + t_1) \stackrel{\delta_1}{\longrightarrow} (q_1, v_1) \stackrel{\sigma_2^{t_2}/\tau_2^{t_2}}{\longrightarrow} (q_1, v_1 + t_2) \stackrel{\delta_2}{\longrightarrow} \cdots ,$$

such that $\sum t_i$ diverges. A run is accepting if for every $F \in \mathcal{F}$, the set of time instants in which it visits states in F is unbounded. The input signal carried by the run is $\sigma_1^{t_1} \cdot \sigma_2^{t_2} \cdots$ and the output is $\tau_1^{t_1} \cdot \tau_2^{t_2} \cdots$ The automaton realizes a sequential relation $f_{\mathcal{A}}$ over its input and output alphabets defined by $f_{\mathcal{A}}(\xi) = \xi'$ iff the accepting run induced by the input signal ξ, outputs the signal ξ'.

4 Main Result

In this section we show how to build for every MITL formula φ a property tester, a timed signal transducer $\mathcal{A}_\varphi$ whose associated sequential function coincides with the characteristic function of φ, that is, $f_{\mathcal{A}_\varphi} = \chi^\varphi$. The construction follows the pattern of the untimed one by composing testers corresponding to operators that appear in the formula. The tester for the untimed *until* can be easily adapted to signals by associating input and output symbols with states and removing self loops (see Figure 3). The only missing link is the following proposition.

Proposition 1. *One can construct a timed signal transducer that realizes* $\chi^{\Diamond_{[a,b]}p}$.

The construction used to prove this proposition follows the lines of the untimed one, based on generating output predictions non-deterministically and aborting them when they are contradicted by actual values of p in ξ. Dense time, however poses new problems because the set of potential predictions of bounded duration includes signals with an arbitrary number of switchings between true and false, and such predictions cannot be memorized by a finite-state timed device. We first show in the following lemma, also used in [AFH96], that predictions that switch too frequently cannot be true for any sub formula of type $\Diamond_{[a,b]}p$. A similar property was used in [MNP05] to show that past MITL is deterministic. We use the auxiliary variable u for the output of the tester.

Lemma 3. *Let* β *be a Boolean signal satisfying* $\beta = \chi^{\Diamond_{[a,b]}p}(\xi)$ *for* $0 \leq a < b$ *and some arbitrary* ξ. *Then for any factorization* $\beta = v \cdot 0^{r_1} \cdot 1^{r_2} \cdot 0^{r_3} \cdot w$ *we have* $r_2 \geq b - a$.

[11] Note that we have chosen I_q to be "backward-closed" so that $v+t \in cl(I_q)$ implies $v+t' \in I_q$ for every t', $0 \leq t' < t$.

Proof: The following observation concerning the constraints on the values of u and p at every t follows from the definitions: if p holds at $t + b$, u must hold throughout the interval $[t, t + b - a]$. Let $[t_1, t_2)$ be the corresponding interval for 1^{r_2}. Since t_1 is the first instant where u holds, p must start holding at $t_1 + b$ and not before that, because otherwise this would imply that u holds before t_1, contrary to our assumptions. Following the observation, u has to hold during the entire interval $[t_1, t_1 + b - a]$. Consequently, $t_2 \geq t_1 + b - a$ and we are done. ∎

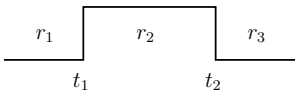

Fig. 4. A signal $\chi^{\Diamond_{[a,b]}p}(\xi)$ for an arbitrary ξ.

The importance of this property is that it bounds the variability of any realizable prediction and constrains the relation between the number of changes and the duration of candidate signals. Let $d = b - a$ and $m = \lceil b/d \rceil + 1$. Since every[12] 10 or 01 segment of β has at least d duration, any acceptable prediction of the form $(01)^m$ or $(10)^m$ has a duration beyond b and, hence, its initial segment can be forgotten. Consequently, $2m$ clocks suffice to memorize relevant predictions.

Let us first explain our construction in discrete time where $\Diamond_{[a,b]}$ is just syntactic sugar:

$$\Diamond_{[a,b]}p \equiv \bigvee_{i \in [a,b]} \bigcirc^i p$$

As this operator is bounded by $d = b - a$, a tester will need to remember at most 2^d predictions, which can be encoded explicitly as states that correspond to elements of $\mathbb{B}^d$. For obvious reasons, this approach will not extend to dense time. Instead, we can encode such sequences by the length (duration) of their 0 and 1 segments as is done in data compression. For example, the sequence 00011011 can be encoded as $0^3 1^2 0^1 1^2$. This information can be memorized by an automaton augmented with additional discrete clocks (counters), each reset upon a change between 0 and 1, and incremented at each step. For this example, assuming x_1 is reset at the first 0, y_1 reset at the subsequent occurrence of 1, and so on, we reach the end of the sequence with $x_1 = 8$, $y_1 = 5$, $x_2 = 3$ and $y_2 = 2$. With such a "symbolic" representation, the relevant past predictions at any time instant are identified via conditions on the clocks, which are used to admit or reject actual values of the input.

Our construction does exactly that in dense time. Due to the symbolic encoding of states, it is simpler to decompose the tester into two parts, as depicted in Figure 5: a *prediction generator* (Figure 6) which generates output signals non-deterministically and the *checker* (Figure 7) which checks whether actual inputs confirm these predictions.

[12] To be more precise, the first 10 segment can be arbitrarily small.

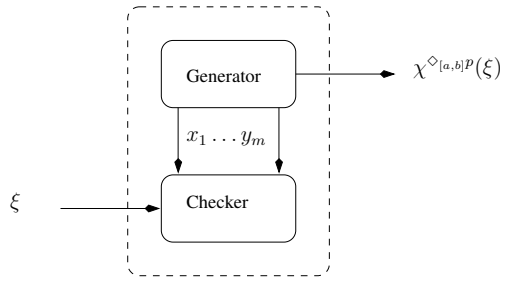

Fig. 5. The architecture of the tester for $\Diamond_{[a,b]}p$.

The generator simply generates arbitrary Boolean signals subject to the sole restriction of bounded variability, according to Lemma 3. This condition is imposed via the guards of the form $y_i \geq b-a$ on all transitions from 1 to 0. The values of the $2m$ clocks $x_1, \ldots x_m$ and $y_1 \ldots y_m$ represent at time t the form of the prediction segment at the time window $[t-b, t]$. Clocks whose value cross b become inactive and can be re-used to memorize new events. This way the denotation of the clocks shifts circularly, and the value of the active clocks always represents the boundaries of the relevant prediction segments.

To see that the checker indeed aborts all but correct predictions observe a prefix of its behavior as it moves through states $\{s_1, s_2, s_3\}$ trying to match prediction to input (see Figure 8). The negative prediction segment at $[t_0, t_1)$ forbids p anywhere in the interval $t_0 + a, t_1 + b)$ and this is guaranteed by forcing the checker to be at the $\overline{p}$-state s_1 from the time $x_1 \geq a$ until the time $y_1 = b$. At time $t_1 + b$, p must be observed and a transition to s_2 is taken. The positive prediction interval requires p to hold during the interval $[t_1 + b, t_2 + a)$ (which corresponds to the clock conditions $y_1 \geq b$ and $x_2 < a$) except, possibly, for short episodes of $\overline{p}$ that last less than $b - a$ time. This is reflected in the transition from s_2 to s_3 which resets the clock z. If $z = b - a$ and we are still in s_3, the run is aborted. When $x_2 \geq a$ we arrive to a new negative prediction segment and move to the next identical segment of the automaton. This bounded operator requires no Büchi condition, and the proof of Proposition 1 is concluded. ▟

To complete the construction for MITL we just need to compose the testers for the propositional, untimed and timed operators according to the structure of the formula. The parallel composition of transducers is fairly standard and we give only the definition of an input-output composition of signal transducers $\mathcal{A}^1 \triangleright \mathcal{A}^2$ where the output of $\mathcal{A}^1$ is the input of $\mathcal{A}^2$. Note that the need for a generalized Büchi condition comes from such a composition of testers for unbounded operators as we need to identify accepting runs of $\mathcal{A}^2$ triggered by outputs of accepting runs of $\mathcal{A}^1$.

Definition 3 (I/O Composition). *Let* $\mathcal{A}^1 = (\Sigma^1, Q^1, \Gamma^1, \mathcal{C}^1, \lambda^1, \gamma^1, I^1, \Delta^1, q_0^1, F^1)$ *and* $\mathcal{A}^2 = (\Sigma^2, Q^2, \Gamma^2, \mathcal{C}^2, \lambda^2, \gamma^2, I^2, \Delta^2, q_0^2, F^2)$ *be timed signal transducers such that* $\Gamma^1 = \Sigma^2$. *Their I/O composition is the transducer*

$$\mathcal{A} = \mathcal{A}^1 \triangleright \mathcal{A}^2 = (\Sigma^1, Q, \Gamma^2, \mathcal{C}, \lambda, \gamma, I, \Delta, q_0, F)$$

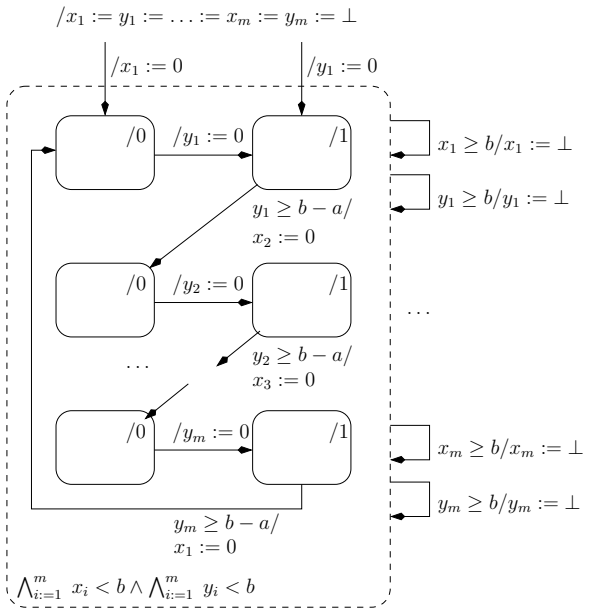

Fig. 6. The prediction generator. We use a StateChart-like notation when several states admit the same transition or staying condition.

where

$$Q = \{(q^1, q^2) \in Q^1 \times Q^2 \textit{ s.t. } \gamma^1(q^1) = \lambda^2(q^2)\},$$

$\mathcal{C} = \mathcal{C}^1 \cup \mathcal{C}^2$, $\lambda(q^1, q^2) = \lambda^1(q^1)$, $\gamma(q^1, q^2) = \gamma^2(q^2)$, $I_{(q^1,q^2)} = I^1_{q^1} \cap I^2_{q^2}$, $q_0 = (q^1_0, q^2_0)$ *and* $F = F^1 \cup F^2$. *The transition relation* Δ *is the restriction to* Q *of the set of all transitions of either of the following forms*

$$((q^1, q^2), g^1 \cap g^2, \rho^1 || \rho^2, (q'^1, q'^2))$$
$$((q^1, q^2), g^1 \cap I_{q^2}, \rho^1, (q'^1, q^2))$$
$$((q^1, q^2), I_{q^1} \cap g^2, \rho^2, (q^1, q'^2))$$

such that $(q^1, g^1, \rho^1, q'^1) \in \Delta^1$ *and* $(q^2, g^2, \rho^2, q'^2) \in \Delta^2$.

It is not hard to see that $A^1 \triangleright A^2$ realizes the sequential function obtained by composing the sequential functions realized by $\mathcal{A}^1$ and $\mathcal{A}^2$.

Corollary 1 (Main Result). *MITL formulae can be transformed into timed automata using a simple procedure.*

This construction provides a decision procedure for MITL by emptiness checking. Following the results in [TYB05,Tri06], timed Büchi emptiness checking can be reduced to the search of accepting cycles in the zone-closed simulation graph, which is generated in practice by timed verification tools such as KRONOS citekronos, UPPAAL [LPY97]

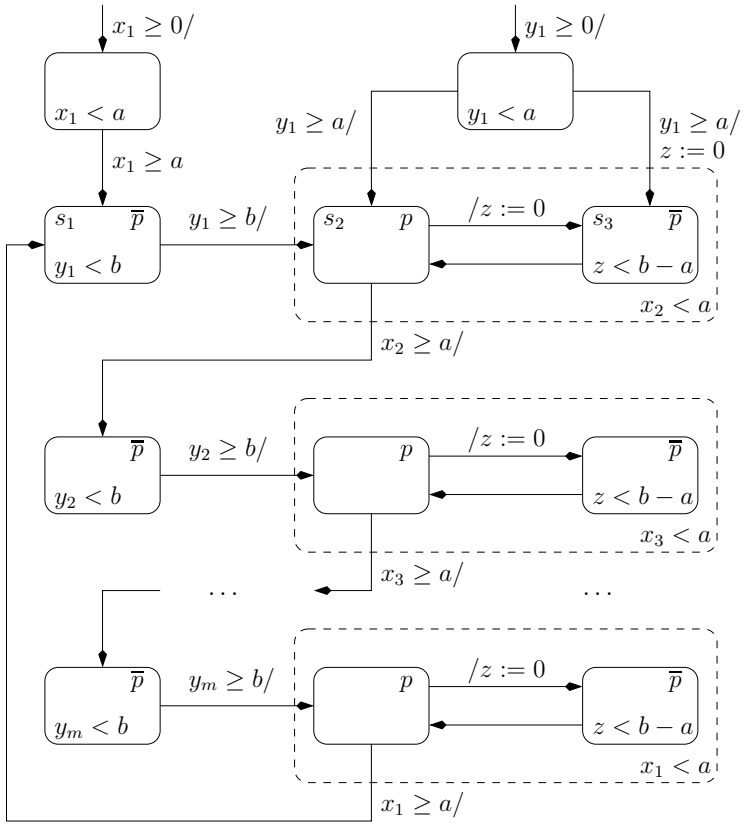

Fig. 7. The automaton for checking predictions.

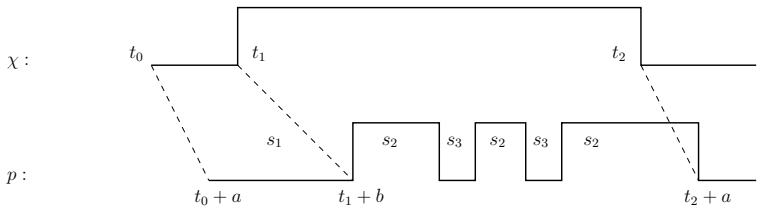

Fig. 8. A behavior of the checker on a prediction-input pair.

or IF [BGM02]. The test for emptiness can be conducted efficiently on-the-fly using standard maximal SCC or double DFS algorithms [CVWY92]. Due to the generalized Büchi conditions, our implementation is based on the nested DFS algorithm of [Tau06].

Given a real-time system S modeled by a timed automaton $\mathcal{A}_S$ and an MITL formula φ, the model checking problem, that is, the language inclusion $L(\mathcal{A}_S) \subseteq L(\varphi)$ between the possible behaviors of $\mathcal{A}_S$ and the behaviors satisfying φ, reduces to the timed Büchi emptiness check on the product automaton $\mathcal{A}_S \times \mathcal{A}_{\neg\varphi}$, where $\mathcal{A}_{\neg\varphi}$ is obtained from the negated MITL formula φ using the modular construction described in this paper.

5 Discussion

Our definitions deviate from those of [AFH96] in the following respects:

1. We disallow signals that admit punctuality;
2. We restrict the temporal modalities to closed intervals;
3. We modify the semantics of $p\,\mathcal{U}\,q$ to require a "handshake" moment where both p and q hold.

The restriction to non-punctual signals is very reasonable from a semantic point of view and constitutes the natural choice for signals. The two other modifications are consequences of this choice as we want the output of the testers to be valid signals as well. The main limitation of this logic is the inability to specify *events* (or the *rising* and *falling* of a signal) which prevents, for example, expressing properties such as *bounded variability*. The extension of the logic to express subsets of the more general signal-event monoid [ACM02] is a topic for future research.

Let us also remark that giving preference to results proved with respect to the most general existing definitions is a mathematicians attitude that should not be adopted without a critical examination. Such an attitude can be counter-productive in young domains where "classical" results and definitions are only decade old, and the most appropriate formalization has not yet stabilized.

Acknowledgment We thank anonymous referees of several conferences for their remarks and suggestions.

References

[Alu99] R. Alur, Timed Automata, *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**, 116–146, 1996 (first published in *PODC'91*).

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

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

[A04] 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.

[BGM02] M. Bozga, S. Graf and L. Mounier, IF-2.0: A Validation Environment for Component-Based Real-Time Systems, *CAV'02*, 343–348, LNCS 2404, 2002.

[BCM+92] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill and L.J. Hwang, Symbolic Model Checking: 10^{20} States and Beyond, *Information and Computation* **98**, 140–170, 1992.

[CGH94] E.M. Clarke, O. Grumberg and K. Hamaguchi, Another look at LTL Model Checking, *CAV'94*, 415–427, LNCS 818, 1994.

[CGP99] E.M. Clarke, O. Grumberg, and D.A. Peled, *Model Checking*. The MIT Press, 1999.

[CVWY92] C. Courcoubetis, M.Y. Vardi, P. Wolper and M. Yannakakis, Memory-Efficient Algorithms for the Verification of Temporal Properties, *Formal Methods in System Design* **1**, 275–288, 1992.

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

[DT04] D. D'Souza and N. Tabareau, On Timed Automata with Input-determined Guards, *FORMATS/FTRTFT'04*, 68–83, LNCS 3253, 2004.

[GO01] P. Gastin and D. Oddoux, Fast LTL to Büchi Automata Translation, *CAV'01*, 53–65, LNCS 2102, 2001.

[GPVW95] R. Gerth, D.A. Peled, M.Y. Vardi and P. Wolper, Simple On-the-fly Automatic Verification of Linear Temporal Logic, *PSTV*, 3–18, 1995.

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

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

[HFE04] J. Havlicek, D. Fisman and C. Eisner, Basic results on the semantics of Accellera PSL 1.1 foundation language, *Technical Report 2004.02*, Accelera, 2004.

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

[HRS98] T.A. Henzinger, J.-F. Raskin, and P.-Y. Schobbens, The Regular Real-time Languages, *ICALP'98*, 580–591, LNCS 1343, 1998.

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

[KP05] Y. Kesten and A. Pnueli, A Compositional Approach to CTL* Verification, *Theoretical Computer Science* **331**, 397–428, 2005.

[Koy90] R. Koymans, Specifying Real-time Properties with Metric Temporal Logic, *Real-time Systems* **2**, 255–299, 1990.

[KCV04] A. Kumari B. Cohen and S. Venkataramanan, *Using PSL/Sugar for Formal and Dynamic Verification*, VhdlCohen Publishing, 2004.

[LPY97] K.G. Larsen, P. Pettersson and W. Yi, Uppaal in a Nutshell, *International Journal of Software Tools for Technology Transfer* **1** 134–152, 1997.

[MMP92] O. Maler, Z. Manna, and A. Pnueli, From Timed to Hybrid Systems, *Real-Time: Theory in Practice*, 447–484, LNCS 600, 1992.

[MN04] O. Maler and D. Nickovic, Monitoring Temporal Properties of Continuous Signals, *FORMATS/FTRTFT'04*, 152–166, LNCS 3253, 2004.

[MNP05] O. Maler, D. Nickovic and A. Pnueli, Real Time Temporal Logic: Past, Present, Future, *FORMATS'05*, 2–16, LNCS 3829, 2005.

[MP91] Z. Manna and A. Pnueli, *Temporal Verification of Reactive Systems: Specification*, Springer, 1991.

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

[OW05] J. Ouaknine and J. Worrell, On the Decidability of Metric Temporal Logic, *LICS'05*, 188–197, 2005.

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

[SB00] F. Somenzi and R. Bloem, Efficient Büchi automata from LTL formulae, *CAV'00*, 248–263, LNCS 1855, 2000. 1855.

[Tau06] H. Tauriainen, Nested Emptiness Search for Generalized Bchi automata *Fundamenta Informaticae*, **70**, 127–154, 2006.

[Tri06] S. Tripakis, Checking Timed Büchi Automata Emptiness on Simulation Graphs, *Technical Report 2006-1*, Verimag, 2006.

[TYB05] S. Tripakis, S. Yovine and A. Bouajjani, Checking Timed Büchi Automata Emptiness Efficiently, *Formal Methods in System Design* **26**, 267-292, 200.

[Var96] M.Y. Vardi, Alternating Automata and Program Verification, *Current Trends in Computer Science*, 267–278, LNCS 1000, 1996.

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

[Y97] S. Yovine, Kronos: A Verification Tool for Real-time Systems, *International Journal of Software Tools for Technology Transfer* **1**, 123–133, 1997.