Model Checking for Extended Timed Temporal Logics

Ahmed Bouajjani[1] Yassine Lakhnech[2] Sergio Yovine[1]

[1] VERIMAG, Miniparc-Zirst, Rue Lavoisier, 38330 Montbonnot St-Martin, France.
email: Ahmed.Bouajjani@imag.fr Sergio.Yovine@imag.fr

[2] Institut für Informatik und Praktische Mathematik Christian-Albrechts-Universität zu Kiel, Preußerstr. 1-9, D-24105 Kiel, Germany.
email: yl@informatik.uni-kiel.d400.de

Abstract

We introduce the real-time temporal logic BTATL_p which is obtained by extending CTL* with both past operators and timed automata constraints. Considering together past operators and automata constraints allows expressing timing requirements in a simple and natural way. Model-checking for full BTATL_p is undecidable. Fortunately, there exist significant (both linear-time and branching-time) fragments for which the verification problem is decidable. In particular, we identify a sublogic of BTATL_p, which is more expressive than TCTL, and for which model-checking can be effectively done. The practical interest of this logic is illustrated through the example of the Philips audio protocol.

1 Introduction

One of the major approaches for specifying and verifying (reactive) systems consists in using ω-automata and/or temporal logics as specification formalisms with model checking algorithms for automatic verification. This framework has been extended to the case of real-time systems by adapting the existing specification formalisms and verification techniques in order to take into account hard timing constraints [4, 1, 3, 10].

Timed ω-automata and timed temporal logics have been defined by adding the ability of resetting and testing clocks, which allows to express upper and lower bounds on the time distances between events. For instance, this extension allows to express in the logic TPTL (timed PTL) [4] properties like "*whenever P occurs, eventually Q will be observed within 5 time units*".

Unfortunately, this natural extension of the usual specification formalisms does not preserve in general the decidability of their verification problem. Indeed, this problem is undecidable for (dense-time) TPTL [4]. Moreover, while the emptiness problem of timed ω-automata is decidable, their universality problem is undecidable [2], and hence, verifying a property expressed by timed ω-automata is in general undecidable.

Nevertheless, model checking algorithms have been proposed for timed logics like the branching-time logic TCTL [1, 10] and the linear-time logic MITL [3]. However, in many practical cases we have considered, specifications are either not expressible in these logics or expressible in a cumbersome and not natural way. It is indeed possible the make the following observations. First, as it is well known, many properties are more simply expressed using past operators. It has also been shown in [5] that adding past operators enhances the expressiveness of timed logics like MITL. Second, it is often convenient and sometimes necessary to express parts of the specification using automata, especially those involving complex timing dependencies between events. Third, it is often the case that we need to express branching-time properties (i.e., requiring nested alternation of universal and existential path quantifiers) involving nontrivial path constraints (i.e., linear-time properties).

The aim of this paper is to propose a formalism which is powerful enough to express all the kinds of properties mentioned above and for which model checking can be effectively done.

We start by defining as a general specification framework the logic BTATL_p, which is a branching-time temporal logic with past operators where timed automata are used as constraints. Automata constraints allow to say that the computation from some designated position to the current one is accepted by some given automaton on finite timed sequences. In particular, this automaton can simply check that the time length of the considered computation segment satisfies some upper or lower bound constraint, and hence, the use of automata as constraints generalizes the way of expressing constraints in timed logics like TPTL.

Clearly, the verification problem for BTATL_p is undecidable. Indeed, the language of its path formulas, called TATL_p, subsumes the logic TPTL. Therefore, we introduce a sublogic called BTATL_p^* for which we provide a model checking algorithm. To define the logic BTATL_p^* we consider two different fragments of TATL_p for describing path formulas, called TATL_p^+ and TATL_p^-. These fragments are dual in the sense that negations of TATL_p^+ formulas are equivalent to TATL_p^- formulas and conversely. BTATL_p^* is defined by allowing TATL_p^+ formulas under universal path quantification and TATL_p^- formulas under existential path quantification.

BTATL_p^* allows to express all the useful properties like invariants and response properties. In particular, we use it to describe properties of the Philips audio protocol [6]. This experience shows that BTATL_p^* allows a natural expression of properties that are in general not expressible in the other existing logics for which the verification problem is decidable.

Our model checking algorithm for BTATL_p^* is based on the fact that we can translate each path formula in TATL_p^- into a timed Muller ω-automaton. Hence, by the fact that existentially quantified path formulas are in TATL_p^- and universally quantified ones are in the dual fragment TATL_p^+, the model checking algorithm consists in, roughly speaking, solving nested emptiness problems for timed ω-automata (which is decidable).

To prove that TATL_p^- formulas can be characterized by timed ω-automata, we give a construction which associates with every formula a 2-way timed ω-automaton with a bounded number of reversals, and then use the fact that these kind of 2-way automata are convertible into 1-way automata [5]. Moreover, we show that direct constructions of 1-way automata, without resorting to 2-way automata, are possible for significant fragments of TATL_p^-, in particular for TPTL_p^- which consists of formulas that are in both TATL_p^- and TPTL_p.

The remainder of this paper is organized as follows. In Section 2, we define 2-way timed automata and timed systems. The logic BTATL_p is introduced in Section 3, and its sublogic BTATL_p^* is defined in Section 4. In Section 5, we use BTATL_p^* to specify the Philips audio protocol. In Section 6, we give the constructions of timed ω-automata for TATL_p^- and TPTL_p^- formulas. Then, in Sections 7 and 8, we present the model checking procedures for the linear-time logic TATL_p^+ and the branching-time logic BTATL_p^*. Concluding remarks are given in Section 9.

2 Timed automata and timed systems

We give hereafter the definitions of 2-way timed graphs and (ω-)automata, and introduce the notion of timed systems. These definitions are inspired from those introduced in [5] and [2].

2.1 Two-way timed graphs

Let Σ be a finite alphabet. We call *timed sequence* over Σ any finite or infinite sequence $\theta = \langle\sigma_1, \tau_1\rangle \cdots \langle\sigma_i, \tau_i\rangle \cdots$ where the σ_i's are symbols in Σ and the τ_i's are positive real numbers such that $\forall i \geq 1.\ \tau_i < \tau_{i+1}$ and $\lim_{i\to\infty} \tau_i = \infty$.

Two-way timed graphs are finite control machines running back and forth over timed sequences. They consist of finite transition systems supplied with a finite set of variables called *clocks*. Transitions of these graphs correspond to moves on the input timed sequence that can be either to the left or to the right of the current position. These transitions are conditioned by constraints on the time distances between the current position and some other designated positions on the input timed sequence. A special variable T is used to represent permanently the time in the current position. The other clocks are used to memorize the time values at different positions in the sequence.

Let us give the formal definitions. Given a set of clocks, a *clock constraint* is a boolean combination of atomic formulas of the form $|T - x| \prec c$ where x is a clock, $\prec$ stands for either $\leq$ or $<$, and c is a nonnegative integer constant. A *clock valuation* is a mapping which assigns to each clock a positive real value. Given a clock constraint f and a clock valuation ν, we denote by $\nu \models f$ the fact that f evaluates to true under ν (ν satisfies f). Given a valuation ν and a set of clocks X, we denote by $\nu[X \mapsto t]$ the valuation which associates with each clock in X the value t, and coincides with ν on all the other clocks. Then, a *2-way timed graph* $\mathcal{G}$ over Σ consists of the following components:

- $\mathcal{Q}$, a finite set of control locations,
- $\Pi \in [\mathcal{Q} \to \Sigma]$, a labelling function,
- $\mathcal{X}$, a finite set of clocks containing a special variable T,

– $\mathcal{E}$, a set of edges. Each edge is a tuple $e = (q, f, X, \delta, q')$ where $q, q' \in \mathcal{Q}$ are the source and target locations, f is a clock constraint (enabling guard), $X \subseteq \mathcal{X} \setminus \{T\}$ is a set of reset clocks, and $\delta \in \{+, -\}$ is a direction indicator.

A *configuration* of $\mathcal{G}$ is a triplet $\langle q, \nu, t\rangle$ where q is a control location, ν is a valuation of the clocks in $\mathcal{X} \setminus \{T\}$, and t is the time value associated with T.

Let $\theta = \langle\sigma_1, \tau_1\rangle \cdots \langle\sigma_i, \tau_i\rangle \cdots$ be a finite or infinite timed sequence over Σ. A *situation* of $\mathcal{G}$ w.r.t. θ is a tuple $\langle q, \nu, t, \ell\rangle$ where $\langle q, \nu, t\rangle$ is a configuration of $\mathcal{G}$ and ℓ is a positive integer such that $1 \leq \ell \leq |\theta|$ which indicates the position of $\mathcal{G}$ on θ.

Let $\langle q, \nu, t, \ell\rangle$, $\langle q', \nu', t', \ell'\rangle$ be situations of $\mathcal{G}$ w.r.t. θ, and $e = (q, f, X, \delta, q') \in \mathcal{E}$. Then, we write $\langle q, \nu, t, \ell\rangle \xrightarrow{e} \langle q', \nu', t', \ell'\rangle$ iff $(\nu, t') \models f$ and $\nu' = \nu[X \mapsto t']$ and, either $\delta =$ "$+$" and $\ell < |\theta|$ and $\ell' = \ell + 1$, or $\delta =$ "$-$" and $\ell > 1$ and $\ell' = \ell - 1$.

Let $\kappa = \langle q, \nu, t\rangle$ be a configuration of $\mathcal{G}$. Then, a *run* of $\mathcal{G}$ over θ starting from κ is a sequence of situations $\rho = \langle q_1, \nu_1, t_1, \ell_1\rangle \cdots \langle q_i, \nu_i, t_i, \ell_i\rangle \cdots$ such that

– $\langle q_1, \nu_1, t_1\rangle = \kappa$ and $\ell_1 = 1$,
– $\forall i.\ 1 \leq i \leq |\rho|.\ \Pi(q_i) = \sigma_{\ell_i}$ and $t_i = \tau_{\ell_i}$,
– $\forall i.\ 1 \leq i < |\rho|.\ \exists e \in \mathcal{E}.\ \langle q_i, \nu_i, t_i, \ell_i\rangle \xrightarrow{e} \langle q_{i+1}, \nu_{i+1}, t_{i+1}, \ell_{i+1}\rangle$,
– $\forall i.\ 1 \leq i \leq |\theta|.\ \exists j.\ 1 \leq j \leq |\rho|.\ \ell_j = i$.

For every nonnegative number k, the 2-way timed graph $\mathcal{G}$ is *k-bounded* if for every timed sequence θ, every run ρ of $\mathcal{G}$ over θ, and every $i \geq 1$, the number of position indices ℓ_j's that are equal to i is less than or equal to $2k + 1$.

When for every edge of $\mathcal{G}$, the direction indicator δ is "$+$" (i.e., $\mathcal{G}$ is 0-bounded), we say that $\mathcal{G}$ is a *1-way timed graph*, or simply a *timed graph*. To simplify the description of timed graphs, we omit sometimes direction indicators in their edges, and write their clock constraints as $T - x \prec c$ instead of $|T - x| \prec c$. We also write $X := T$ for the set of reset clocks X.

Given a timed graph $\mathcal{G}$ over Σ, a run ρ of $\mathcal{G}$ over a timed sequence θ can be considered as a sequence of configurations instead of situations since the indices ℓ_i's are not relevant. In this case, the timed sequence θ coincides with the sequence $\langle\Pi(q_1), t_1\rangle \cdots \langle\Pi(q_i), t_i\rangle \cdots$ obtained from the run ρ and denoted by $Trail(\rho)$. We can also interpret timed graphs as *generators* of timed sequences, and hence, we say that ρ *generates* the timed sequence $Trail(\rho)$.

2.2 Two-way timed automata on finite sequences

A 2-way timed automaton on finite timed sequences over the alphabet Σ is a triplet $A = (\mathcal{G}, \mathcal{I}, \mathcal{F})$ where $\mathcal{G} = (\mathcal{Q}, \Pi, \mathcal{X}, \mathcal{E})$ is a 2-way timed graph over Σ, $\mathcal{I} \subseteq \mathcal{Q}$ is a set of *initial* control locations, and $\mathcal{F} \subseteq \mathcal{Q}$ is a set of *final* control locations. We say that A is a 1-way timed automaton if $\mathcal{G}$ is a 1-way timed graph.

Let $\theta = \langle\sigma_1, \tau_1\rangle \cdots \langle\sigma_n, \tau_n\rangle$ be a finite timed sequence over Σ. We say that A *accepts* θ if there exists a run $\rho = \langle q_1, \nu_1, t_1, \ell_1\rangle \cdots \langle q_m, \nu_m, t_m, \ell_m\rangle$ of $\mathcal{G}$ over θ such that $q_1 \in \mathcal{I}$, $\nu_1(x) = t_1$ for every $x \in \mathcal{X} \setminus \{T\}$, and $q_m \in \mathcal{F}$. We denote by $L(A)$ the set of finite timed sequences accepted by A.

Deterministic (2-way) timed automata can be defined as in [5]. They have at most one run over every finite timed sequence and they are closed under all boolean operations whereas nondeterministic timed automata are not closed under complementation [2]. We omit their formal definition since it is not necessary for the understanding of the paper.

2.3 Two-way timed ω-automata on infinite sequences

A 2-way timed Büchi (resp. Muller) ω-automaton over Σ, is a triplet $\mathcal{A} = (\mathcal{G}, \mathcal{I}, \Gamma)$ where $\mathcal{G} = (\mathcal{Q}, \Pi, \mathcal{X}, \mathcal{E})$ is a 2-way timed graph over Σ, $\mathcal{I} \subseteq \mathcal{Q}$ is a set of *initial* control locations, and Γ is a set $\mathcal{F} \subseteq \mathcal{Q}$ (resp. $\Delta \subseteq 2^{\mathcal{Q}}$) called Büchi (resp. Muller) acceptance condition. We say that $\mathcal{A}$ is a 1-way timed Büchi (resp. Muller) ω-automaton, or simply a timed Büchi (resp. Muller) ω-automaton, if $\mathcal{G}$ is a 1-way timed graph.

Let $\theta = \langle \sigma_1, \tau_1 \rangle \cdots \langle \sigma_i, \tau_i \rangle \cdots$ be an infinite timed sequence over Σ. Then, given a run $\rho = \langle q_1, \nu_1, t_1, \ell_1 \rangle \cdots \langle q_i, \nu_i, t_i, \ell_i \rangle \cdots$ of $\mathcal{G}$ over θ, we denote by $Inf(\rho)$ the set of control locations q such that $\overset{\infty}{\exists}\, i \geq 1.\ q_i = q$. Then, we say that ρ is an *accepting run* of the timed Büchi (resp. Muller) ω-automaton $\mathcal{A} = (\mathcal{G}, \mathcal{I}, \Gamma)$ over the timed sequence θ if $q_1 \in \mathcal{I}$, $\nu_1(x) = t_1$ for every $x \in \mathcal{X} \setminus \{T\}$, and $Inf(\rho) \cap \Gamma \neq \emptyset$ (resp. $Inf(\rho) \in \Gamma$). We denote by $L(\mathcal{A})$ the set of infinite timed sequences having accepting runs of $\mathcal{A}$.

As in the case of automata on finite timed sequences, deterministic (2-way) timed ω-automata can be defined. We recall that deterministic timed Muller ω-automata are closed under all boolean operations, whereas deterministic and nondeterministic Büchi and nondeterministic Muller timed ω-automata are not closed under complementation [2].

2.4 Timed systems

We call *timed system* a timed graph with self-loops on all the control locations and a fairness constraint on the edges. Self-loops allow to make several successive observations at the same location (with strictly increasing times) whereas the fairness constraint allows to prevent staying at the same location forever. Formally, a *timed system* over Σ is a pair $\mathcal{T} = (\mathcal{G}, \Omega)$ where $\mathcal{G} = (\mathcal{Q}, \Pi, \mathcal{X}, \mathcal{E})$ is a timed graph over Σ, such that $\forall q \in \mathcal{Q}$, $(q, true, \emptyset, q) \in \mathcal{E}$, and $\Omega \subseteq \mathcal{E}$ is a set of *fair edges*.

Given a configuration κ of $\mathcal{T}$, a sequence of configurations $\rho = \kappa_1 \cdots \kappa_i \cdots$ is a *computation* of $\mathcal{T}$ starting from κ, if ρ is an infinite run of $\mathcal{G}$ starting from κ on the timed sequence $Trail(\rho)$ such that $\exists e \in \Omega.\ \overset{\infty}{\exists}\, i \geq 1.\ \kappa_i \xrightarrow{e} \kappa_{i+1}$. We denote by $\mathcal{C}(\mathcal{T}, \kappa)$ the set of computations of $\mathcal{T}$ starting from κ.

3 Branching-time TATL$_p$

Let $\mathcal{P}$ be a finite set of atomic propositions, and let $\Sigma = 2^{\mathcal{P}}$. We use letters $P, Q, \ldots$ to range over elements of $\mathcal{P}$. Let us introduce a set $\mathcal{W}$ of *position variables* and use letters $u, v, \ldots$ to range over $\mathcal{W}$. We use letters $A, B, \ldots$ to range over (1-way) timed automata on finite timed sequences.

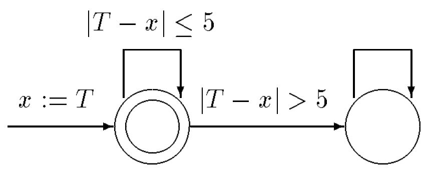
The set of *state* (resp. *path*) BTATL$_p$ *formulas* is the set of formulas φ (resp. ψ) defined by the following grammar:

$$\begin{aligned} \varphi &::= P \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists\psi \\ \psi &::= \varphi \mid u.\, \psi \mid A^u \mid \neg\psi \mid \psi \vee \psi \mid \psi \mathcal{U} \psi \mid \psi \mathcal{S} \psi \end{aligned}$$

In addition, we introduce abbreviations as the usual boolean connectives $\wedge$ and $\Rightarrow$, and the temporal operators $\Diamond\psi = true\mathcal{U}\psi$, $\Diamond_p\psi = true\mathcal{S}\psi$, $\Box\psi = \neg\Diamond\neg\psi$, and $\Box_p\psi = \neg\Diamond_p\neg\psi$, and universal path quantification $\forall\psi = \neg\exists\neg\psi$.

Given a timed system $\mathcal{T}$ over Σ, state formulas are interpreted as sets of configurations of $\mathcal{T}$ whereas path formulas are interpreted as sets of computations of $\mathcal{T}$. The operators $\mathcal{U}$, $\mathcal{S}$, $\Diamond$, $\Diamond_p$, $\Box$, and $\Box_p$ are the classical *until*, *since*, *eventually in the future*, *sometime in the past*, *always in the future*, and *always in the past* operators of the linear-time propositional temporal logic PTL$_p$ [12]. The state formula $\exists\psi$ means that there exists a computation starting from the current configuration which satisfies the path formula ψ. Hence, when interpreting ψ the past cannot go beyond the current configuration. This way of introducing past modalities in branching-time logics corresponds to the interpretation adopted in [9].

The construction "$u.$" associates the current position (index) on the computation with the position variable u. Then, u can be used as a label allowing to refer to the position associated with it. The formula A^u when interpreted at some position i means that the automaton A accepts the timed sequence generated by the finite subcomputation starting at u and ending at i, by running either forward or backward on the computation according to the fact that i is greater or smaller than the position of u. We call formulas of the form A^u *automata constraints*. For example, let $T_{\leq 5}$ be the automaton represented by the following picture:



Then, the property "*whenever P holds, Q will be eventually true within* 5 *time units*" is expressed by the formula

$$\Box\,(P \Rightarrow u.\, \Diamond(Q \wedge T^u_{\leq 5})) \tag{1}$$

and the property "*whenever Q holds, P was necessarily true no longer than* 5 *time units ago*" is expressed by the formula

$$\Box\,(Q \Rightarrow u.\, \Diamond_p(P \wedge T^u_{\leq 5})) \tag{2}$$

The formulas (1) and (2) above are examples of *time-constrained response properties*. Another kind of properties are *time-constrained invariants* as for instance "*whenever P holds, Q is continuously true for at least* 5 *time units*", which is expressed by the formula

$$\Box\,(P \Rightarrow u.\,\Box(T^u_{\leq 5} \Rightarrow Q)) \tag{3}$$

or "*whenever P holds, Q was true during the last* 5 *time units*", expressed by:

$$\Box\,(P \Rightarrow u.\,\Box_p(T^u_{\leq 5} \Rightarrow Q)) \tag{4}$$

In the formula $u.\,\psi$, the construction "$u.$" binds the position variable u in ψ. We call the construction "$u.$" *position quantification*. We suppose without loss of generality that in every formula, each position variable is bound at most once. Then, every variable appearing in some formula is either *bound* or *free*. A formula is *closed* if all the variables occurring in it are bound, otherwise it is *open*.

For the sake of simplicity, we suppose from now on that all state formulas are closed, i.e., each automata constraint refer to some position introduced within the scope of the inner-most path quantifier under which the constraint occurs. However, path formulas may be open.

The formal semantics of BTATL_p is given by means of two satisfaction relations, namely $\models_{\mathcal{T}}$, defined between configurations and state formulas, and $\models\!\!\!\equiv_{\mathcal{T}}$, defined between computations and path formulas. The relation $\models\!\!\!\equiv_{\mathcal{T}}$ is parameterized by a *position association* E that associates with each position variable the position (index) on the computation where it has been introduced. We denote the parametrized satisfaction relation by $\models\!\!\!\equiv_{\mathcal{T},E}$. The definition of $\models\!\!\!\equiv_{\mathcal{T},E}$ depends on the current position on the computation. Hence, we write $(\rho, i) \models\!\!\!\equiv_{\mathcal{T},E} \psi$ to say that the computation ρ satisfies ψ at position i w.r.t. E. In the definition of $\models\!\!\!\equiv_{\mathcal{T},E}$, the position association E is updated due to the introduction of new position variables. We denote by $\emptyset$ the position association whose domain is empty, and denote by $E[u \mapsto i]$ the position association whose domain is the extension of the domain of E by u, and which associates with u the position i and coincides with E on all the other variables. We denote simply $\models\!\!\!\equiv_{\mathcal{T}}$ the relation $\models\!\!\!\equiv_{\mathcal{T},\emptyset}$, and write $\rho \models\!\!\!\equiv_{\mathcal{T}} \psi$ for $(\rho, 1) \models\!\!\!\equiv_{\mathcal{T}} \psi$.

Given a timed system $\mathcal{T}$, for every configuration $\langle q, \nu, t\rangle$ and every state formula φ, the meaning of $\langle q, \nu, t\rangle \models_{\mathcal{T}} \varphi$ is inductively defined by:

$$\begin{array}{ll}
\langle q,\nu,t\rangle \models_{\mathcal{T}} P & \text{iff } P \in \Pi(q) \\
\langle q,\nu,t\rangle \models_{\mathcal{T}} \neg\varphi & \text{iff } \langle q,\nu,t\rangle \not\models_{\mathcal{T}} \varphi \\
\langle q,\nu,t\rangle \models_{\mathcal{T}} \varphi_1 \vee \varphi_2 & \text{iff } \langle q,\nu,t\rangle \models_{\mathcal{T}} \varphi_1 \text{ or } \langle q,\nu,t\rangle \models_{\mathcal{T}} \varphi_2 \\
\langle q,\nu,t\rangle \models_{\mathcal{T}} \exists\psi & \text{iff } \exists\rho \in \mathcal{C}(\mathcal{T}, \langle q,\nu,t\rangle).\ \rho \models\!\!\!\equiv_{\mathcal{T}} \psi
\end{array}$$

where, for every computation ρ of $\mathcal{T}$, every $i \geq 1$, and every position association E, the meaning of $(\rho, i) \models\!\!\!\equiv_{\mathcal{T},E} \psi$ is inductively defined by:

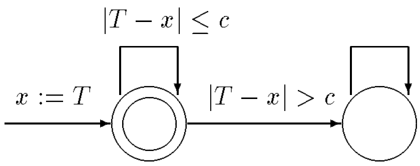
$$
\begin{array}{lll}
(\rho,i) \models_{\mathcal{T},E} \varphi & \text{iff} & \rho[i,i] \models_{\mathcal{T}} \varphi \\
(\rho,i) \models_{\mathcal{T},E} u.\psi & \text{iff} & (\rho,i) \models_{\mathcal{T},E[u \mapsto i]} \psi \\
(\rho,i) \models_{\mathcal{T},E} A^u & \text{iff} & Trail(\rho[E(u),i]) \in L(A) \\
(\rho,i) \models_{\mathcal{T},E} \neg\psi & \text{iff} & (\rho,i) \not\models_{\mathcal{T},E} \psi \\
(\rho,i) \models_{\mathcal{T},E} \psi_1 \vee \psi_2 & \text{iff} & (\rho,i) \models_{\mathcal{T},E} \psi_1 \text{ or } (\rho,i) \models_{\mathcal{T},E} \psi_2 \\
(\rho,i) \models_{\mathcal{T},E} \psi_1 \mathcal{U} \psi_2 & \text{iff} & \exists j \geq i.\ (\rho,j) \models_{\mathcal{T},E} \psi_2 \text{ and } \forall k.\ i \leq k < j.\ (\rho,k) \models_{\mathcal{T},E} \psi_1 \\
(\rho,i) \models_{\mathcal{T},E} \psi_1 \mathcal{S} \psi_2 & \text{iff} & \exists j.\ 1 \leq j \leq i.\ (\rho,j) \models_{\mathcal{T},E} \psi_2 \text{ and} \\
 & & \quad \forall k.\ j < k \leq i.\ (\rho,k) \models_{\mathcal{T},E} \psi_1
\end{array}
$$

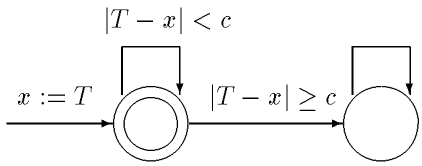
where for every computation $\rho = \kappa_0 \cdots \kappa_i \cdots$, $\rho[j,i]$ denotes the finite subcomputation $\kappa_{j_0} \cdots \kappa_{j_n}$ where $n = |j-i|$ and for all $0 \leq k \leq n$, κ_{j_k} is κ_{j+k} if $j \leq i$, or κ_{j-k} otherwise.

Given a closed state formula φ, $[\![\varphi]\!]_{\mathcal{T}}$ is the set of configurations κ such that $\kappa \models_{\mathcal{T}} \varphi$.

Several sublogics of BTATL$_p$ can be considered. We define the linear-time logic TATL$_p$ as the set of BTATL$_p$ path formulas without any occurrence of path quantifiers (ψ formulas where the φ's can only be atomic propositions). The semantics of TATL$_p$ formulas can actually be given in terms of sets of timed sequences instead of computations of a fixed timed system $\mathcal{T}$. Given a closed TATL$_p$ formula ψ, we denote by $[\![\psi]\!]$ the set of timed sequences satisfying ψ.

Then, the logics TATL [7] and BTATL are the "*future*" fragments of TATL$_p$ and BTATL$_p$, i.e., they correspond respectively to the sets of TATL$_p$ and BTATL$_p$ formulas without occurrences of past operators. The linear-time temporal logic TPTL [4] is the sublogic of TATL where automata constraints can only constrain time distances. Indeed, for every integer constant c, let $T_{\leq c}$ and $T_{<c}$ be the timed automata represented respectively by the following pictures:





Notice that the automata above are deterministic, and hence, the complements of their languages are recognized respectively by the automata $T_{>c}$ and $T_{\geq c}$ obtained by inversing final and nonfinal locations. Then, the logics TPTL [4] and TPTL$_p$ are defined as the respective sublogics of TATL and TATL$_p$ where automata constraints are only expressed by means of $T_{\leq c}$ and $T_{<c}$ (or $T_{>c}$ and $T_{\geq c}$), for any integer constant c. The same restriction applied on BTATL and BTATL$_p$ leads to the definitions of the branching-time logics TCTL* [1] and TCTL$^*_p$, respectively. Finally, the branching-time logic TCTL [1] is the fragment of TCTL* such that path formulas can only be of the forms $u.\, \phi_1 \mathcal{U} \phi_2$ or $\neg u.\, \phi_1 \mathcal{U} \phi_2$, where the ϕ_i's are boolean combinations of automata constraints (e.g., A^u) and state formulas.

Given a timed system $\mathcal{T}$ and a configuration κ, the *verification* (*model-checking*) *problem* for a state formula φ consists in deciding whether $\kappa \in [\![\varphi]\!]_{\mathcal{T}}$, whereas for a path formula ψ, it consists in deciding whether $Trail(\mathcal{C}(\mathcal{T},\kappa)) \subseteq [\![\psi]\!]$, or equivalently, whether $\kappa \in [\![\forall\psi]\!]_{\mathcal{T}}$.

It has been shown that the verification problem of TCTL is decidable [1]. This problem is however undecidable for TPTL [4]. Hence, the verification problem is also undecidable for $\mathrm{TATL}_{(p)}$, $\mathrm{TCTL}^*_{(p)}$, and $\mathrm{BTATL}_{(p)}$, where the subscript (p) means with or without past operators.

4 The logic BTATL^*_p

BTATL^*_p is the sublogic of BTATL_p whose set of state (resp. path) formulas φ (resp. ψ) is defined by the following grammar:

$$\begin{aligned}
\varphi &::= P \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists\psi \\
\psi &::= \phi \mid u.\,\psi \mid \psi \vee \psi \mid \psi \wedge \psi \mid \phi\mathcal{U}\psi \mid \phi\mathcal{S}\psi \\
\phi &::= \varphi \mid A^u, A \text{ is deterministic } \mid \neg\phi \mid \phi \vee \phi \mid \phi\mathcal{U}\phi \mid \phi\mathcal{S}\phi
\end{aligned}$$

Notice that, while the set of BTATL^*_p state formulas (φ) is closed under negation, the set of path formulas (ψ) is not. It is convenient, however, to dispose of universal path quantification. Then, we extend the set of BTATL^*_p formulas by state formulas of the form $\forall\psi'$ and path formulas ψ' defined by:

$$\psi' \ ::= \ \phi \mid u.\,\psi' \mid \psi' \vee \psi' \mid \psi' \wedge \psi' \mid \Box\psi' \mid \Box_p\psi' \mid \psi'\mathcal{U}\phi \mid \psi'\overline{\mathcal{U}}\phi \mid \psi'\mathcal{S}\phi \mid \psi'\overline{\mathcal{S}}\phi$$

where $\psi_1\overline{\mathcal{U}}\psi_2 = \psi_1\mathcal{U}(\psi_1 \wedge \psi_2)$ and $\psi_1\overline{\mathcal{S}}\psi_2 = \psi_1\mathcal{S}(\psi_1 \wedge \psi_2)$. We show that this extension does not enrich the expressiveness of the logic because ψ formulas are equivalent to negations of ψ' formulas and conversely. However, this extension allows to consider two different languages of path formulas, each of them characterizing the class of path properties that are either existentially or universally quantified.

We introduce two linear-time timed logics TATL^-_p and TATL^+_p whose sets of formulas are respectively the sets of path formulas ψ and ψ' defined above, without occurrences of path quantifiers (where each state formula can only be an atomic proposition). We introduce also the fragments TPTL^-_p and TPTL^+_p of TPTL defined in the obvious way. The denotations of the corresponding fragments without past operators are obtained by removing the subscript p.

Proposition 4.1 *For every $TATL^+_{(p)}$ (resp. $TPTL^+_{(p)}$) closed formula ψ, there exists a $TATL^-_{(p)}$ (resp. $TPTL^-_{(p)}$) closed formula ψ' such that $[\![\neg\psi]\!] = [\![\psi']\!]$, and conversely.*

We denote by BTPTL^*_p the fragment of BTATL^*_p where existentially (resp. universally) quantified path formulas are in TPTL^-_p (resp. TPTL^+_p).

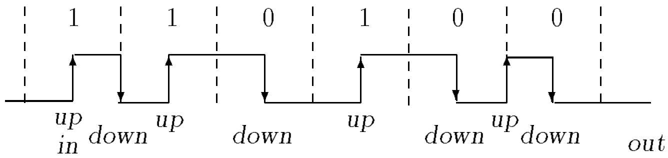
Let us discuss briefly the expressiveness of the logics we have introduced. It has been shown in [7] that TATL^+ as well as TATL^- are more expressive than deterministic timed Muller ω-automata. While TATL^+ can express properties that are not definable by means of timed automata [7], we show in Section 6 that every TATL^-_p formula can be characterized by a timed Muller automaton. Observe that TATL^+_p and TPTL^+_p can express time-constrained invariance and response properties as for instance (1), (2), (3), and (4). The following pictures compare the expressiveness of the logics we consider.

$$\begin{tikzcd}
\mathrm{TATL}^-_{(p)} \arrow[r] & \mathrm{TATL}_{(p)} & \mathrm{TATL}^+_{(p)} \arrow[l] & & \mathrm{BTATL}^*_{(p)} \arrow[r] & \mathrm{BTATL}_{(p)} \\
\mathrm{TPTL}^-_{(p)} \arrow[r] \arrow[u] & \mathrm{TPTL}_{(p)} \arrow[u] & \mathrm{TPTL}^+_{(p)} \arrow[l] \arrow[u] & \mathrm{TCTL} \arrow[r] & \mathrm{BTPTL}^*_{(p)} \arrow[r] \arrow[u] & \mathrm{TCTL}^*_{(p)} \arrow[u]
\end{tikzcd}$$

Besides, TATL_p^+ and TATL_p^-, as well as their fragments TPTL_p^+ and TPTL_p^- are incomparable with the linear-time logics MITL [3], its extension with past operators MITL^P [5], and EMITL [16]. Indeed, all these logics forbid punctual time constraints whereas TATL_p^+ and TPTL_p^+ do not. Moreover, each time constraint in MITL^P and EMITL formulas is related to one temporal operator whereas constraints in TATL_p^+ and TPTL_p^+ formulas are related to position variables, which allows to constrain overlapped computation segments.

5 Example

In this section we show using the Philips audio control protocol [6] that BTATL_p^* allows to naturally express the requirements of a realistic case study. The system is composed of a sender and a receiver connected through a wire. The sender sends bit streams using Manchester encoding. A "1" is sent by raising the voltage in the middle of the bit slot, that is, the voltage is low during the first half of the slot and high in the second one. Sending a "0" is just the reverse. When the same bit is sent in two consecutive bit slots, an additional edge is placed in between. The following picture illustrates the encoding of the bit stream "110100".



The length of the bit slot is 4. The events *in* and *out* model respectively the request to transmit a bit stream and the corresponding output. Upgoing and downgoing edges are modeled by the events *up* and *down*. The first *up* occurs concurrently with the input *in*.

The particular hardware used by Philips is such that downgoing edges cannot be reliably detected, and therefore the receiver must decode the binary signal only from the upgoing ones, and that the length of the bit slot can only be measured with some known error $\epsilon \leq \frac{1}{20}$. Following [6] we also assume that sender and receiver synchronize on the event *up*, that is, there is no propagation delay, and that whenever an action *in* happens before the action *out* corresponding to the previous message, the protocol moves to a *chaotic* state where everything is allowed.

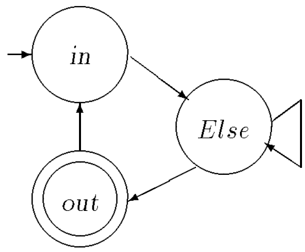
We do not model here the behavior of the components of the system but we concentrate instead on the properties the protocol is required to satisfy. Timed automata based descriptions of the sender and the receiver are given in [11, 8].

The first requirement is that

the received bit stream is equal to the one sent, provided the protocol is not in a chaotic state.

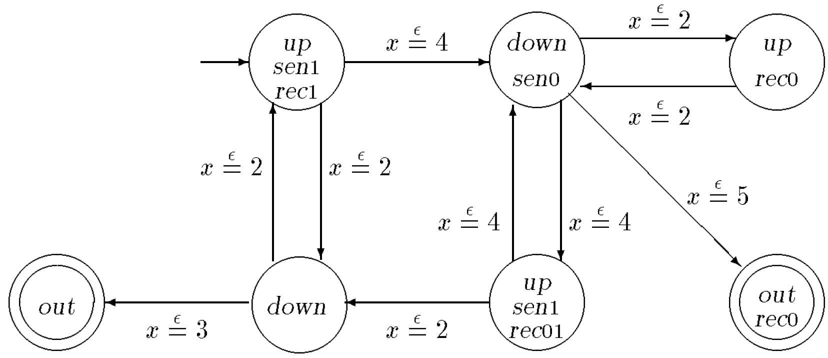
That is, whenever the receiver produces an *out* in a non-chaotic state then necessarily an *in* occurred in the past when the sender started transmitting the bit stream, and since this last occurrence of *in* the system must have transmitted correctly the bit stream according to the Manchester encoding. In order to express this property in BTATL_p^* we need to characterize the set of behaviors leading to non-chaotic *out*-states, and the ones corresponding to correct transmissions.

Non-chaotic states are those satisfying that everywhere in the past, between any two consecutive occurrences of *in* there is an *out* and vice versa. Besides, it must be ensured that no bit has been sent nor received between an *out* and the next *in*. This means that the transmission terminates with the *out* event (i.e., both the sender stopped transmitting and the receiver stopped receiving). Behaviors leading to non-chaotic states where *out* holds are characterized by the automaton A_{out} depicted below where *Else* stands for the set of events not containing neither *in* nor *out*.



In order to characterize the correct transmissions we have to ensure that the bit stream has been sent using Manchester encoding and that it has been correctly received. For that we construct a timed automaton B as follows. B uses a clock x to measure the time elapsed since the last change in the voltage of the wire. The value of x also allows to distinguish between the first half and the second half of the bit slot. Let $x \stackrel{\epsilon}{=} c$ stand for $c - \epsilon \leq T - x \leq c + \epsilon$. An occurrence of *up* when $x \stackrel{\epsilon}{=} 4$ corresponds to the emission of a "1" which is modeled by the event *sen*1. Otherwise, if *up* occurs when $x \stackrel{\epsilon}{=} 2$, then it must be the required upgoing edge between to consecutive "0"s. To send a "0", modeled by the event *sen*0, the sender produces a *down* when $x \stackrel{\epsilon}{=} 4$. *down*'s occurring when $x \stackrel{\epsilon}{=} 2$ correspond to downgoing edges placed between to consecutive "1"s. Besides, *up*'s and *down*'s must alternate. Since the receiver does not see *out*'s, it cannot decode "0"s at the same time they are sent but must wait until the next *up*, so it may be *behind* the sender, that is, the received bit stream may be shorter than the one sent. Anyway, the receiver is only allowed to decode at most two bits at a time, namely a single "0" or "1", or a sequence "01", respectively modeled by the events *rec*0, *rec*1 and *rec*01. Thus, proving that the receiver correctly decodes the bit stream comes down to showing that at any time the receiver is at most a "0" behind the sender, and that it eventually adds the missing "0" at the end of message whenever required. Besides, since the voltage must be low at the end of the transmission, the event *out* can only occur after a *down*.

The automaton B is depicted in the following picture where the labeling of a location must be interpreted as a set of events that must occur simultaneously, and the clock x is set to T by every transition (omitted in the figure).



Then, the required property can be expressed by the following BTATL$^*_p$ formula:

$$\forall u.\Box\,[A^u_{out} \Rightarrow \neg in\,\mathcal{S}\,(in \wedge v.\Diamond B^v)] \tag{5}$$

The previous property stated that every non-chaotic *out* state corresponds to the end of a correct transmission. We need also to express that every non-chaotic *in* state is the starting of a correct transmission. We do not exclude that some behaviors after *in* correspond to incorrect transmissions leading to chaotic states (if another *in* occurs before *out*). However, the protocol must guarantee that:

whenever the sender starts transmitting a message in a non-chaotic state, there exists at least one behavior leading to a correct transmission.

To express this property, we need a universal path quantification over paths leading to non-chaotic *in* states, and existential path quantification on paths starting at these states. Non-chaotic *in* states are characterized by an automaton A_{in} defined as A_{out} above, by considering as final control location the *in* location instead of the *out* one. Then, the property we are interested in can be expressed by the following BTATL$^*_p$ formula:

$$\forall u.\Box(A^u_{in} \Rightarrow \exists v.\Diamond B^v) \tag{6}$$

6 Relating logics to automata

We show in this section that TATL^-_p closed formulas can be effectively characterized by means of 1-bounded 2-way timed Büchi ω-automata. By transforming these automata into 1-way automata, we get a construction of timed Büchi ω-automata for TATL^-_p closed formulas. We give also a direct construction of (1-way) timed Büchi ω-automata in the case of TPTL^-_p closed formulas.

First of all, we need to put TATL^-_p formulas in a special syntactical form. We say that a TATL^-_p formula is in *normal form* if it is a disjunction of formulas ξ defined by:

$\xi ::= \phi \mid u.\xi \mid \xi \wedge \xi \mid \phi \mathcal{U} \xi \mid \phi \mathcal{S} \xi$
$\phi ::= P \mid \neg P \mid A^u, A$ is deterministic $\mid \phi \vee \phi \mid \phi \wedge \phi \mid \Box \phi \mid \Box_p \phi \mid \phi \mathcal{U} \phi \mid \phi \mathcal{S} \phi$

Lemma 6.1 *Every $TATL_p^-$ formula can be transformed into an equivalent formula in normal form.*

Theorem 6.1 *For every $TATL_p^-$ closed formula ψ, we can effectively construct a 1-bounded 2-way timed Büchi ω-automaton $\mathcal{A}_\psi$ over Σ such that $L(\mathcal{A}_\psi) = [\![\psi]\!]$.*

Proof. By Lemma 6.1, we can assume that ψ is in normal form. We show hereafter how we can construct a 2-way timed Büchi ω-automaton $\mathcal{A}_\xi$ for any disjunct ξ of ψ that recognizes precisely the set of timed sequences $[\![\xi]\!]$. Then, the theorem follows from the fact that 2-way timed Büchi ω-automata are closed under union.

The construction generalizes the one given in [7] for TATL$^-$ by taking into account the fact that, due to the combination of future and past operators, automata constraints can now be expressed forward and backward starting from some given position. The basic observation behind our construction is that every TATL$_p^-$ formula allows to consider only a finite number of positions on timed sequences from which automata constraints can start. This is due to the fact that position quantification cannot occur in the left-hand side of $\mathcal{U}$ and $\mathcal{S}$ as well as in the scope of $\Box$ and $\Box_p$ (see definition of normal form above). Hence, for each position variable appearing in the formula corresponds one position on the sequence. This fact allows to decompose the construction of $\mathcal{A}_\xi$ by considering separately the two following problems:

- checking "*propositional consistency*", that is, whether a timed sequence satisfies the formula modulo abstraction from automata constraints,
- checking "*automata constraints consistency*", that is, whether a timed sequence respects the automata constraints imposed by the formula.

To construct the automaton $\mathcal{A}_\xi$, we proceed in the following manner:

First, we abstract from position quantification and automata constraints and replace them by special atomic propositions; these propositions are used later as markers of the positions where automata (involved in automata constraints) must start running and where they must accept. So, let us associate with each position variable u (resp. constraint A^u) appearing in ξ a new atomic proposition at-u (resp. $[A^u]$). We denote by ξ^* the formula obtained by substituting in ξ each occurrence of a position quantification $u.$ (resp. constraint A^u) by the corresponding proposition at-u (resp. $[A^u]$). Clearly, the formula ξ^* is syntactically in the propositional temporal logic with past operators PTL$_p$. Therefore, we can construct (using [15] for instance) a Büchi ω-automaton $\mathcal{B}_{\xi^*}$ which accepts all the infinite sequences satisfying (according to PTL$_p$ semantics) the PTL$_p$ formula ξ^*. Then, by considering ξ^* as a TPTL$_p$ formula, a timed ω-automaton $\mathcal{A}_{\xi^*}$ recognizing the set of timed sequences $[\![\xi^*]\!]$ is straightforwardly obtained from $\mathcal{B}_{\xi^*}$ by adding trivial constraints on its transitions.

Then, we introduce a 1-bounded 2-way timed Büchi ω-automaton $\mathcal{A}_{const}$ which ensures that the timed sequences fulfil the automata constraints, i.e., every proposition at-u occurs exactly at one position, and all subsequences delimited by propositions at-u and $[A^u]$ must be accepted by the automaton A. The construction of $\mathcal{A}_{const}$ is done as follows: The automaton $\mathcal{A}_{const}$ starts by a forward pass where it looks for the first position in the sequence where all the propositions at-u's have been previously encountered. Then, $\mathcal{A}_{const}$ checks the satisfaction of automata constraints in two independent phases, backward until the initial position is reached, and then forward, forever. In each phase, whenever $\mathcal{A}_{const}$ encounters a proposition at-u, it starts simulating in parallel all the automata involved in formulas A^u referring to u, together with all the automata already running since previously visited initialization positions. Several propositions of the form at-u can be true at the same position, and hence, $\mathcal{A}_{const}$ must consider all their related automata simultaneously. Then, whenever some proposition $[A^u]$ holds, $\mathcal{A}_{const}$ checks that the corresponding automaton A is at some final control location. In the backward pass, $\mathcal{A}_{const}$ simulates the automata involved in automata constraints after reversing the direction indicators of their edges (all the δ's become equal to "$-$"). Moreover, during these two phases, $\mathcal{A}_{const}$ checks also that each proposition at-u does not occur twice on the sequence. A special atomic proposition *init* is introduced to mark the initial position on the sequence. The proposition *init* is true at the initial location of $\mathcal{A}_{const}$ and false in all the other locations of the first forward pass (the initial location has no self loops). Then, *init* is used to detect the end of the backward pass and becomes false in all the locations of the last forward pass. The set of repeating control locations of $\mathcal{A}_{const}$ are all those corresponding to the last forward pass.

Finally, we consider the *product* automaton $\mathcal{A}_{\xi^*} \times \mathcal{A}_{const}$ which works in the following manner: First, it starts by simulating $\mathcal{A}_{const}$ during its first forward pass and its backward pass, and then, it simulates $\mathcal{A}_{\xi^*}$ in parallel with $\mathcal{A}_{const}$ in its last forward pass. The repeating control locations of $\mathcal{A}_{\xi^*} \times \mathcal{A}_{const}$ are the pairs of $\mathcal{A}_{\xi^*}$ and $\mathcal{A}_{const}$ locations whose first components are repeating locations of $\mathcal{A}_{\xi^*}$. Clearly, $\mathcal{A}_{\xi^*} \times \mathcal{A}_{const}$ recognizes the intersection of the languages of the two automata $\mathcal{A}_{\xi^*}$ and $\mathcal{A}_{const}$. We obtain the automaton $\mathcal{A}_{\xi}$ as a projection on $\mathcal{P}$ of $\mathcal{A}_{\xi^*} \times \mathcal{A}_{const}$. ■

Corollary 6.1 *For every TATL$_p^-$ closed formula ψ, we can effectively construct a timed Büchi ω-automaton $\mathcal{A}_\psi$ over Σ such that $L(\mathcal{A}_\psi) = [\![\psi]\!]$.*

Proof. By Theorem 6.1, we can associate with each TATL$_p^-$ closed formula ψ a 1-bounded 2-way timed ω-automaton. Moreover, it has been shown in [5] that, for every integer $k \geq 0$, k-bounded 2-way timed automata on finite sequences can be transformed into equivalent 1-way timed automata. Since, the 2-way ω-automata we construct for TATL$_p^-$ formulas have only one reversal, it is possible to adapt the transformation of [5] to their case, and get equivalent 1-way ω-automata. ■

We present hereafter two subclasses of TATL$_p^-$ for which it is possible to construct a 1-way ω-automaton directly without resorting to 2-way automata.

The first class consists of the set of formulas where between every occurrence of a position variable u and every occurrence of a formula A^u there is no past operator. Indeed, in the proof of Theorem 6.1, reversals of the automaton are only needed to check automata constraints A^u expressed between the position u and some other position in the *past* of u. For an example of a formula in this class, see formula (5) given in Section 5.

Then, the second class is the sublogic TPTL_p^-. The following theorem holds.

Theorem 6.2 *For every $TPTL_p^-$ closed formula ψ, we can effectively construct a timed Büchi ω-automaton $\mathcal{A}_\psi$ over Σ such that $L(\mathcal{A}_\psi) = [\![\psi]\!]$.*

Proof. The automaton $\mathcal{A}_{const}$ of the previous construction can be defined as a timed Büchi automaton in the following manner: To check the satisfaction of past time constraints, for each subformula $T^u_{\prec c}$ (resp. $T^u_{\succ c}$) of ξ, where $\prec$ (resp. $\succ$) stands for either $<$ or $\leq$ (resp. $>$ or $\geq$), the automaton $\mathcal{A}_{const}$ memorizes the time t of the first (resp. last) occurrence of the proposition $[T^u_{\prec c}]$ (resp. $[T^u_{\succ c}]$) before the position where at-u is true. When $\mathcal{A}_{const}$ reaches this latter position at some time t_u, it checks that $t_u - t \prec c$ (resp. $t_u - t \succ c$) holds. This procedure is possible since only a finite number of clock variables are needed, one for each time constraint appearing in ξ. Now, in order to check the satisfaction of future time constraints, the automaton $\mathcal{A}_{const}$ memorizes the time t_u associated with each proposition at-u, and then, it can check that all the forthcoming occurrences of the propositions $[T^u_{\prec c}]$ and $[T^u_{\succ c}]$ are at time distances from t_u that satisfy the corresponding constraints. Again, this procedure is possible since there are only finitely many positions where the propositions at-u's occur. ■

7 Model checking: the linear-time case

In this section, we consider the verification problem of timed systems with respect to the linear-time logic TATL_p^+. We show that this problem is decidable.

Let ψ be a TATL_p^+ closed formula. Then, by Proposition 4.1 and Corollary 6.1, we can effectively construct a timed Büchi ω-automaton $\mathcal{A}_{\neg\psi}$ which recognizes the set of timed sequences $[\![\neg\psi]\!]$. Hence, given a timed system $\mathcal{T}$ and a configuration κ of $\mathcal{T}$, verifying that $Trail(\mathcal{C}(\mathcal{T},\kappa)) \subseteq [\![\psi]\!]$ (i.e., that κ satisfies ψ) consists in deciding whether $Trail(\mathcal{C}(\mathcal{T},\kappa)) \cap L(\mathcal{A}_{\neg\psi}) = \emptyset$. To do so, we proceed as in [2] by introducing a finite partition of the configuration space which preserves the emptiness property above.

Given a timed graph $\mathcal{G} = (\mathcal{Q}, \Pi, \mathcal{X}, \mathcal{E})$, for every $x \in \mathcal{X}$, we denote by $m(x,\mathcal{G})$ the maximal natural number c such that $T - x \prec c$ occurs in the clock constraints on the edges in $\mathcal{E}$. Then, given two valuations ν and ν', and two time values t and t', we write $(\nu, t) \sim_{\mathcal{G}} (\nu', t')$ if and only if the following assertions are true:

- for every $x \in \mathcal{X}$, and every natural numbers $c, c' \leq m(x,\mathcal{G})$, $t - \nu(x) = c$ iff $t' - \nu'(x) = c$, and $c < t - \nu(x) < c'$ iff $c < t' - \nu'(x) < c'$,
- for every $x, y \in \mathcal{X}$ such that $t - \nu(x) \leq m(x,\mathcal{G})$ and $t - \nu(y) \leq m(y,\mathcal{G})$, $fract(t - \nu(x)) \leq fract(t - \nu(y))$ iff $fract(t' - \nu'(x)) \leq fract(t' - \nu'(y))$.

It can be verified that $\sim_{\mathcal{G}}$ is a finite index equivalence between the valuations of the clocks of $\mathcal{G}$. Moreover, it is not difficult to show the following facts:

Lemma 7.1 *If* $(\nu_1, t_1) \sim_{\mathcal{G}} (\nu_2, t_2)$ *then*

1. $\forall X \subseteq \mathcal{X}.\ (\nu_1[X \mapsto t_1], t_1) \sim_{\mathcal{G}} (\nu_2[X \mapsto t_2], t_2)$,
2. $\forall t_1' > t_1$ *with* $(\nu_1, t_1') \models f$, $\exists t_2' > t_2$ *such that* $(\nu_2, t_2') \models f$ *and* $(\nu_1, t_1') \sim_{\mathcal{G}} (\nu_2, t_2')$.

Then, given two configurations $\langle q, \nu, t\rangle$ and $\langle q', \nu', t'\rangle$ of $\mathcal{G}$, we write $\langle q, \nu, t\rangle \sim_{\mathcal{G}} \langle q', \nu', t'\rangle$ iff $q = q'$ and $(\nu, t) \sim_{\mathcal{G}} (\nu', t')$. For every configuration κ of $\mathcal{G}$, we denote by $[\kappa]$ its class of equivalence. Using Lemma 7.1, we can prove that:

Lemma 7.2 *Let* $\mathcal{T} = (\mathcal{G}, \Omega)$ *be a timed system,* κ *and* κ' *two configurations of* $\mathcal{T}$ *such that* $\kappa \sim_{\mathcal{G}} \kappa'$, *and* $\mathcal{A} = (\mathcal{G}', \mathcal{I}, \Gamma)$ *a timed* ω*-automaton. If there exist a computation* $\rho = (\langle q_i, \nu_i, t_i\rangle)_{i\geq 1}$ *starting from* κ *and an accepting run* $\langle \tilde{q}_i, \tilde{\nu}_i, t_i\rangle_{i\geq 1}$ *of* $\mathcal{A}$ *over* $Trail(\rho)$, *then there exist a computation* $\rho' = (\langle q_i, \nu_i', t_i'\rangle)_{i\geq 1}$ *starting from* κ' *and an accepting run* $(\langle \tilde{q}_i, \tilde{\nu}_i', t_i'\rangle)_{i\geq 1}$ *of* $\mathcal{A}$ *over* $Trail(\rho')$ *such that* $\forall i \geq 1.\ \langle q_i, \nu_i, t_i\rangle \sim_{\mathcal{G}} \langle q_i, \nu_i', t_i'\rangle$ *and* $\langle \tilde{q}_i, \tilde{\nu}_i, t_i\rangle \sim_{\mathcal{G}'} \langle \tilde{q}_i, \tilde{\nu}_i', t_i'\rangle$.

As an immediate consequence of Lemma 7.2, we have the following fact:

Lemma 7.3 *Let* $\mathcal{T} = (\mathcal{G}, \Omega)$ *be a timed system, and let* κ *and* κ' *be two configurations of* $\mathcal{T}$ *such that* $\kappa \sim_{\mathcal{G}} \kappa'$. *Then, for every timed* ω*-automaton* $\mathcal{A}$, $Trail(\mathcal{C}(\mathcal{T}, \kappa)) \cap L(\mathcal{A}) = \emptyset$ *iff* $Trail(\mathcal{C}(\mathcal{T}, \kappa')) \cap L(\mathcal{A}) = \emptyset$.

Using Lemma 7.3 we prove that:

Lemma 7.4 *Given a timed system* $\mathcal{T}$, *a configuration* κ *of* $\mathcal{T}$ *and timed* ω*-automaton* $\mathcal{A}$. *The problem whether* $Trail(\mathcal{C}(\mathcal{T}, \kappa)) \cap L(\mathcal{A}) = \emptyset$ *is decidable.*

Proof. Let $\mathcal{T} = (\mathcal{G}, \Omega)$, $\kappa = \langle q, \nu, t\rangle$ of $\mathcal{T}$, and $\mathcal{A} = (\mathcal{G}_{\mathcal{A}}, \mathcal{I}_{\mathcal{A}}, \mathcal{F}_{\mathcal{A}})$. First, we transform the timed system $\mathcal{T}$ in such a way that the fairness condition expressed as a set of fair edges Ω becomes a Büchi-like fairness condition expressed by a set of repeating control locations $\mathcal{F}_{\Omega}$. This transformation uses a standard technique which consists in considering a new set of locations obtained as the product of the set of locations of $\mathcal{T}$ with the set of its edges $\mathcal{E}$, and taking all the edges $((q, e), f, X, (q', e'))$ such that $e' = (q, f, X, q')$. The set of clocks is obviously unchanged, and each new location (q, e) inherits the label of q. Let $\mathcal{G}^*$ be the new timed graph obtained by this transformation. The corresponding set of repeating locations $\mathcal{F}_{\Omega}$ is the set of locations whose projections on $\mathcal{E}$ are in Ω.

Then, consider the timed graph $\mathcal{G}^* \times \mathcal{G}_{\mathcal{A}}$ obtained as a product of $\mathcal{G}^*$ and $\mathcal{G}_{\mathcal{A}}$: The control locations of the product are pairs of locations in $\mathcal{G}^*$ and $\mathcal{G}_{\mathcal{A}}$ with the same label, the set of clocks is the union of the two sets of clocks (supposed to be disjoint without loss of generality), and the edges are parallel compositions of edges in the two graphs (the clock constraints of the paired edges are obtained by conjunction and their sets of reset clocks are obtained by union).

By Lemma 7.3, we can decide the emptiness of the set $Trail(\mathcal{C}(\mathcal{T}, \langle q, \nu, t\rangle)) \cap L(\mathcal{A})$ by considering the finite-state *region graph* defined as the quotient of $\mathcal{G}^* \times \mathcal{G}_{\mathcal{A}}$ w.r.t. the equivalence $\sim_{\mathcal{G}^* \times \mathcal{G}_{\mathcal{A}}}$, and by checking on this finite graph that, starting from some vertex $([\langle q, \nu, t\rangle], [\langle q_{init}, \nu_t, t\rangle])$ where $q_{init} \in \mathcal{I}_{\mathcal{A}}$, and ν_t is the valuation associating t with all the clock variables of $\mathcal{A}$, there exists a path which visits infinitely often locations in $\mathcal{F}_\Omega$ and in $\mathcal{F}_\mathcal{A}$. This reasoning can be straightforwardly adapted to the case where $\mathcal{A}$ is a timed Muller ω-automaton.■

Since verifying that a configuration κ satisfies a TATL$_p^+$ closed formula ψ consists in checking whether $Trail(\mathcal{C}(\mathcal{T},\kappa)) \cap L(\mathcal{A}_{\neg\psi}) = \emptyset$, we have

Theorem 7.1 *The verification problem of timed systems w.r.t. TATL$_p^+$ is decidable.*

8 Model checking: the branching-time case

Let us address now the verification problem of timed systems w.r.t. BTATL$_p^*$. We proceed in the following manner. First, we introduce a logic, called timed-ECTL$_\exists^*$, where path properties are expressed by means of timed Muller automata, in the spirit of [15]. We show that it is possible to construct for every BTATL$_p^*$ formula an equivalent formula in timed-ECTL$_\exists^*$, and then, we give a model checking algorithm for the logic timed-ECTL$_\exists^*$.

8.1 The logic timed-ECTL$_\exists^*$

Timed-ECTL$_\exists^*$ is an extension a timed version of ECTL* [9]. The set of timed-ECTL$_\exists^*$ (state) formulas φ is given by:

$$\varphi ::= P \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists\mathcal{A}(\varphi_1,\ldots,\varphi_n)$$

where $\mathcal{A}$ is a timed Muller ω-automaton over a finite nonempty alphabet $V = \{a_1,\ldots,a_n\}$.

In timed-ECTL$_\exists^*$, timed Muller automata on infinite sequences are used to express path properties: the formula $\exists\mathcal{A}(\varphi_1,\ldots,\varphi_n)$ says that there exists a computation ρ, and there exists an infinite timed sequence $\langle a_{j_1},t_1\rangle\langle a_{j_2},t_2\rangle\cdots$ accepted by the automaton $\mathcal{A}$, such that, at each position i of ρ, the state formula φ_{j_i} is satisfied.

Let us give the formal semantics of timed-ECTL$_\exists^*$ formulas. This semantics is given by means of a satisfaction relation between configurations and formulas. Let $\mathcal{T}$ be a timed system over Σ. Then, for every configuration $\langle q,\nu,t\rangle$, and every formula φ, the meaning of $\langle q,\nu,t\rangle \models_\mathcal{T} \varphi$ is inductively defined by:

$$\begin{array}{ll}
\langle q,\nu,t\rangle \models_\mathcal{T} P & \text{iff } P \in \Pi(q) \\
\langle q,\nu,t\rangle \models_\mathcal{T} \neg\varphi & \text{iff } \langle q,\nu,t\rangle \not\models_\mathcal{T} \varphi \\
\langle q,\nu,t\rangle \models_\mathcal{T} \varphi_1 \vee \varphi_2 & \text{iff } \langle q,\nu,t\rangle \models_\mathcal{T} \varphi_1 \text{ or } \langle q,\nu,t\rangle \models_\mathcal{T} \varphi_2 \\
\langle q,\nu,t\rangle \models_\mathcal{T} \exists\mathcal{A}(\varphi_1,\ldots,\varphi_n) & \text{iff } \exists\rho \in \mathcal{C}(\mathcal{T},\langle q,\nu,t\rangle).\ \exists\langle a_{j_1},t_1\rangle\langle a_{j_2},t_2\rangle\cdots \in L(\mathcal{A}). \\
 & \quad \forall i \geq 1.\ \rho[i,i] \models_\mathcal{T} \varphi_{j_i}
\end{array}$$

For every timed-ECTL$_\exists^*$ formula φ, $[\![\varphi]\!]_\mathcal{T}$ denotes the set of configurations κ of $\mathcal{T}$ satisfying φ.

Theorem 8.1 *Let $\mathcal{T}$ be a timed system. Then, for every BTATL$_p^*$ state formula φ, we can effectively construct a timed-ECTL$_\exists^*$ formula φ' such that $[\![\varphi]\!]_\mathcal{T} = [\![\varphi']\!]_\mathcal{T}$.*

Proof. By induction on the level of nesting of path quantifiers, and is based on the fact that we can associate a timed Büchi ω-automaton with every TATL$_p^-$ closed formula, using Theorem 6.1 and the translation of bounded 2-way timed automata into 1-way timed automata. ∎

8.2 Model checking algorithm

Given a timed system $\mathcal{T}$ and a timed-ECTL$_\exists^*$ formula φ, we propose a model-checking algorithm which consists in deciding for each of its subformulas φ' which configurations of $\mathcal{T}$ satisfy φ', starting from the inner-most subformulas and finishing by the formula φ itself. This algorithm generalizes the one given in [1] for the logic TCTL. We start by proving the following key lemma.

Lemma 8.1 *Let $\mathcal{T} = (\mathcal{G}, \Omega)$ be a timed system and let κ and κ' be two configurations of $\mathcal{T}$ such that $\kappa \sim_\mathcal{G} \kappa'$. Then, for every timed-ECTL$_\exists^*$ formula φ, $\kappa \models_\mathcal{T} \varphi$ iff $\kappa' \models_\mathcal{T} \varphi$.*

Proof. By induction on the structure of the formulas. The cases of atomic propositions and boolean connectives are immediate. The case of a formula of the form $\exists\mathcal{A}(\varphi_1, \cdots, \varphi_n)$ is tackled using Lemma 7.2 and the induction hypothesis. ∎

Roughly speaking, Lemma 8.1 allows to reason about classes of configurations instead of configurations. Let us present informally the main steps leading to the definition of the model checking algorithm.

Given a timed system $\mathcal{T} = (\mathcal{G}, \Omega)$ where $\mathcal{G} = (\mathcal{Q}, \Pi, \mathcal{X}, \mathcal{E})$, the algorithm runs on a *region timed system* $\mathcal{T}_R$. The control locations of $\mathcal{T}_R$ correspond to equivalence classes of configurations of $\mathcal{T}$ w.r.t. the equivalence $\sim_\mathcal{G}$, and the edges of this system are compatible with these classes (they ensures that each location can be reached only when the current configuration is in the corresponding class). So, the construction of the timed system $\mathcal{T}_R$ can be seen as a duplication of the control locations of $\mathcal{T}$ according to the equivalences classes of the possible valuations of the clocks. The system $\mathcal{T}_R$ generates the same timed sequences as the original system $\mathcal{T}$. More precisely, we can prove that for every configuration $\kappa = \langle q, \nu, t\rangle$ of $\mathcal{T}$, we have $\kappa \models_\mathcal{T} \varphi$ iff $\langle [\kappa], \nu, t\rangle \models_{\mathcal{T}_R} \varphi$. This fact implies, by Lemma 8.1, that given two equivalent configurations $\kappa = \langle q, \nu, t\rangle$ and $\kappa' = \langle q, \nu', t'\rangle$ of $\mathcal{T}$, and a timed-ECTL$_\exists^*$ formula φ, we have $\langle [\kappa], \nu, t\rangle \models_{\mathcal{T}_R} \varphi$ iff $\langle [\kappa], \nu', t'\rangle \models_{\mathcal{T}_R} \varphi$. Hence, the model checking algorithm consists in recursively labelling the control locations of the timed system $\mathcal{T}_R$ by the subformulas of φ such that a control location $[\langle q, \nu, t\rangle]$ of $\mathcal{T}_R$ is labelled by a subformula φ' iff $\langle [\langle q, \nu, t\rangle], \nu, t\rangle \models_{\mathcal{T}_R} \varphi'$. Since the number of subformulas is finite, the algorithm terminates, and then, we have $\kappa \models_\mathcal{T} \varphi$ if and only if the location $[\kappa]$ of $\mathcal{T}_R$ is labelled by φ.

It is worth to note that, in contrast to the algorithm for TCTL in [1] which runs on an *untimed* region graph, our algorithm labels the locations of a timed system. Indeed, to take into account the time constraints imposed by the formula, the algorithm in [1] consists in extending the system by the clocks used in the formula, and then reasoning about the region graph of the obtained extended system. In the case of timed-ECTL$^*_\exists$, taken into account the time constraints imposed by the formula is technically more complicated, and the time constraints on the clocks of $\mathcal{T}$ (or equivalently of $\mathcal{T}_R$) must be considered at each step of the algorithm for deciding the satisfaction of nested state formulas involving ω-automata, e.g., $\exists\mathcal{A}(\varphi_1, \ldots, \varphi_n)$. In fact, at each level of nesting, deciding such a satisfaction problem consists in finding a computation $\rho = (\kappa_i)_{i\geq 1}$ of $\mathcal{T}_R$ and a timed sequence $(\langle a_{j_i}, t_i\rangle)_{i\geq 1}$ accepted by $\mathcal{A}$ (over some alphabet $\{a_1, \ldots, a_n\}$) such that, at each position i, the configuration κ_i satisfies the formula φ_{j_i}. Since for each subformula φ_i, we know already whether it is satisfied or not at each location of $\mathcal{T}_R$, the problem we have to solve reduces to checking the emptiness of a *timed* ω-language defined in terms of a product of the *timed* graphs of $\mathcal{T}_R$ (with an updated labelling function) and $\mathcal{A}$. Now, to decide this emptiness problem, we proceed as in the proof of Lemma 7.4 by reasoning about the *untimed* region graph of the considered product timed graph.

Let us formalize the description of the algorithm. Before giving the definition of the timed system $\mathcal{T}_R$, let us introduce some notations. First of all, it is easy to see that for every equivalence class $[\kappa]$ with $\kappa = \langle q, \nu, t\rangle$, there is a clock constraint $\Delta_{[\kappa]}$ such that for every $\kappa' = \langle q, \nu', t'\rangle$, $\kappa \sim_{\mathcal{G}} \kappa'$ iff $(\nu', t') \models \Delta_{[\kappa]}$. In fact, $\Delta_{[\kappa]}$ can be written in the form $\bigwedge_{x\in\mathcal{X}} a_x \sim^1_x T - x \sim^2_x b_x$ where $a_x, b_x \in I\!N \cup \{\infty\}$ and $\sim^1_x, \sim^2_x \in \{<, \leq\}$. Given a clock constraint $f = \bigwedge_{x\in\mathcal{X}} a_x \sim^1_x T - x \sim^2_x b_x$ we denote by $f(X)$ the conjunction $\bigwedge_{x\in X} a_x \sim^1_x T - x \sim^2_x b_x$.

Then, let $\mathcal{T}_R = (\mathcal{Q}_R, \Pi_R, \mathcal{X}_R, \mathcal{E}_R, \Omega_R)$ be the timed system defined as follows: The set $\mathcal{Q}_R$ of control locations is the set of equivalence classes of $\mathcal{T}$ w.r.t. $\sim_{\mathcal{G}}$. The set of clocks $\mathcal{X}_R$ is equal to $\mathcal{X}$. The labelling function Π_R is such that $\Pi_R([\langle q, \nu, t\rangle]) = \Pi(q)$. An edge $([\kappa], f, X, [\kappa'])$ is in $\mathcal{E}_R$ iff either $[\kappa] = [\kappa']$, $f = true$, and $X = \emptyset$, or $\kappa = \langle q, \nu, t\rangle$, $\kappa' = \langle q', \nu', t'\rangle$, and there exists an edge $e = (q, f', X, q')$ of $\mathcal{T}$ such that $\kappa \xrightarrow{e} \kappa'$ and $f = f' \wedge \Delta_{[\kappa']}(\mathcal{X} \setminus X)$. Finally, Ω_R consists of all the edges $([\kappa], f, X, [\kappa'])$, with $\kappa = \langle q, \nu, t\rangle$ and $\kappa' = \langle q', \nu', t'\rangle$, such that $(q, f, X, q') \in \Omega$.

It can be verified that $(\kappa_i)_{i\geq 1} = (\langle q_i, \nu_i, t_i\rangle)_{i\geq 1}$ is a computation of $\mathcal{T}$ iff $(\langle[\kappa_i], \nu_i, t_i\rangle)_{i\geq 1}$ is a computation of $\mathcal{T}_R$. Moreover, it can be seen that $\exists e \in \Omega$ such that $\overset{\infty}{\exists}\, i \geq 1.\ \kappa_i \xrightarrow{e} \kappa_{i+1}$ iff $\exists e' \in \Omega_R$ such that $\overset{\infty}{\exists}\, i \geq 1.\ \langle[\kappa_i], \nu_i, t_i\rangle \xrightarrow{e'} \langle[\kappa_{i+1}], \nu_{i+1}, t_{i+1}\rangle$. Using this fact, we can prove that:

Lemma 8.2 *For every timed-ECTL$^*_\exists$ formula φ and configuration $\kappa = \langle q, \nu, t\rangle$ of $\mathcal{T}$, $\kappa \models_{\mathcal{T}} \varphi$ iff $\langle[\kappa], \nu, t\rangle \models_{\mathcal{T}_R} \varphi$.*

Then, by Lemma 8.1 and Lemma 8.2, we deduce that:

Lemma 8.3 *For every timed-ECTL$^*_\exists$ formula φ, and configurations $\kappa = \langle q, \nu, t\rangle$ and $\kappa' = \langle q, \nu', t'\rangle$ of $\mathcal{T}$ such that $\kappa \sim_{\mathcal{G}} \kappa'$, $\langle[\kappa], \nu, t\rangle \models_{\mathcal{T}_R} \varphi$ iff $\langle[\kappa], \nu', t'\rangle \models_{\mathcal{T}_R} \varphi$.*

Finally, let us show that given a formula φ in timed-ECTL$^*_\exists$, we can effectively label each control locations $[\langle q, \nu, t\rangle]$ of $\mathcal{T}_R$ with all subformulas φ' of φ

such that $\langle [\langle q, \nu, t\rangle], \nu, t\rangle \models_{\mathcal{T}_R} \varphi'$. This labelling is done recursively on the structure of φ. The cases of atomic propositions and boolean combinations of formulas are immediate. Then, consider the case of a formula $\exists\mathcal{A}(\varphi_1, \cdots, \varphi_n)$ and let $\{a_1, \cdots, a_n\}$ be the alphabet of $\mathcal{A}$. Assume that the labelling has been already done for all the formulas $\varphi_1, \cdots, \varphi_n$ and all the locations of $\mathcal{T}_R$, and consider a configuration $\kappa = \langle q, \nu, t\rangle$ of $\mathcal{T}$. We construct a timed system $\mathcal{T}'$ over $\{a_1, \cdots, a_n\}$ from $\mathcal{T}_R$ by replacing the labelling function Π_R by a function Π' such that, for every $i \in \{1, \ldots, n\}$, $a_i \in \Pi'([\kappa'])$ iff $[\kappa']$ is labelled by φ_i. Then, $\langle [\kappa], \nu, t\rangle \models_{\mathcal{T}_R} \exists\mathcal{A}(\varphi_1, \cdots, \varphi_n)$ iff $Trail(\mathcal{C}(\mathcal{T}', \kappa)) \cap L(\mathcal{A}) \neq \emptyset$. By Lemma 7.4, we can decide whether the control location $[\kappa]$ of $\mathcal{T}_R$ must be labelled by the formula $\exists\mathcal{A}(\varphi_1, \cdots, \varphi_n)$ or not. Since the construction of $\mathcal{T}_R$ as well as its labelling are effective, we obtain the following result:

Theorem 8.2 *The verification problem of timed systems w.r.t. timed-ECTL$^*_\exists$ is decidable.*

Using Theorem 8.1, we can reduce the verification problem of timed systems w.r.t. BTATL$^*_p$ to the verification problem of timed systems w.r.t. timed-ECTL$^*_\exists$. Hence, by Theorem 8.2:

Theorem 8.3 *The verification problem of timed systems w.r.t. BTATL$^*_p$ is decidable.*

9 Conclusion

We have introduced a logic called BTATL$^*_p$ as a general specification formalism for real-time systems, and provided a model checking algorithm for it. BTATL$^*_p$ is a branching-time temporal logic with past operators where timed automata are used to express complex timing constraints.

We have shown that properties encountered in a realistic case study can be expressed naturally and concisely in BTATL$^*_p$ using formulas combining past operators, automata constraints, and alternation between existential and universal path quantifiers. Such properties cannot be in general expressed in the other known timed formalisms for which the verification problem is decidable.

It is worth to note that, while the extension of temporal logics by automata has been widely studied in the untimed case [17, 12, 15, 9, 14, 13], very few has been done in the timed case. The only existing works we know on this topic are [16] concerning EMITL, and [7] concerning TATL (in both cases, the logics have a linear-time semantics). The approach adopted in [16] for extending MITL is different from ours. It consists in using *untimed* automata on finite sequences to define temporal operators and expressing time constraints by parameterizing each temporal operator by a time interval, as in MITL.

Our model checking algorithm for BTATL$^*_p$ is based on a translation of the existentially quantified path formulas into timed Muller ω-automata. In general, this translation involves the construction of 2-way automata as an intermediate step. However, most of the formulas used in practice (for instance, those needed in the example of the Philips audio protocol), fit into fragments for which this intermediate step can be avoided. In particular, due to Theorem 6.2, this can be done for formulas in BTPTL_p^* which is a subset of TCTL^* that is more expressive than TCTL.

The model checking algorithm we provide in this paper consists in solving nested emptiness problems of ω-automata, using the region graph technique [2]. To make this algorithm more efficient, a symbolic technique could be developed, similar to the one proposed in [10].

References

1. R. Alur, C. Courcoubetis, and D. Dill. Model-Checking for Real-Time Systems. In *LICS'90*. IEEE, 1990.
2. R. Alur and D. Dill. A Theory of Timed Automata. *TCS*, 126, 1994.
3. R. Alur, T. Feder, and T. Henzinger. The Benefits of Relaxing Punctuality. In *PODC'91*, 1991.
4. R. Alur and T. Henzinger. A Really Temporal Logic. In *FOCS'89*. IEEE, 1989.
5. R. Alur and T. Henzinger. Back to the Future: Towards a Theory of Timed Regular Languages. In *FOCS'92*. IEEE, 1992.
6. D. Bosscher, I. Polak, and F. Vaandrager. Verification of an audio control protocol. In H. Langmaack, W.-P. de Roever, and J. Vytopil, editors, *FTRTFT'94: Formal Techniques in Real-Time and Fault-Taulerant Systems*, pages 170–192. Lecture Notes in Computer Science 863, Springer-Verlag, 1994.
7. A. Bouajjani and Y. Lakhnech. Temporal Logic + Timed Automata : Expressiveness and Decidability. In *CONCUR'95*. LNCS 962, 1995.
8. C. Daws and S. Yovine. Two examples of verification of multirate timed automata with KRONOS. In *Proc. 1995 IEEE Real-Time Systems Symposium, RTSS'95*, Pisa, Italy, December 1995. IEEE Computer Society Press.
9. T. Hafer and W. Thomas. Computation Tree Logic CTL* and Path Quantifiers in the Monadic Theory of the Binary Tree. In *ICALP'87*. LNCS 267, 1987.
10. T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic Model-Checking for Real-Time Systems. In *LICS'92*. IEEE, 1992.
11. P-H. Ho and H. Wong-Toi. Automated analysis of an audio control protocol. In *Proc. 7th Computer-Aided Verification*, Liège, Belgium, July 1995.
12. O. Lichtenstein, A. Pnueli, and L. Zuck. The Glory of the Past. In *Conf. Logics of Programs*. LNCS 193, 1985.
13. Z. Manna and A. Pnueli. A Hierarchy of Temporal Properties. In *PODC'90*. ACM, 1990.
14. W. Thomas. Computation Tree Logic and Regular ω-Languages. LNCS 354, 1989.
15. M.Y. Vardi and P. Wolper. An Automata-Theoretic Approach to Automatic Program Verification. In *LICS'86*. IEEE, 1986.
16. Th. Wilke. Specifying Timed State Sequences in Powerful Decidable Logics and Timed Automata. In *FTRTFT'94*. LNCS 863, 1994.
17. P. Wolper. Temporal Logic Can Be More Expressive. *Inform. and Cont.*, 56, 1983.