Logics vs. Automata: The Hybrid Case (Extended Abstract)

Ahmed Bouajjani[1] Yassine Lakhnech[2*]

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

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

1 Introduction

Hybrid systems can be seen as extensions of reactive and timed systems [13, 14]. Hence, their specification formalisms are obtained naturally by extending those of reactive systems in such a way to be able to reason about the notions of (real) time, duration (time relatively to some state property), etc. There are mainly two families of such formalisms: timed/hybrid automata that are finite-state automata supplied with continuous variables like clocks (to measure time) or integrators (to measure durations) [13, 14, 3], and timed/hybrid logics that are temporal logics allowing time or duration constraints, as for instance the timed temporal logics TPTL [6] and MITL [5], the duration temporal logic DTL [7], or the calculus of durations CoD [10].

The verification problem of hybrid systems is whether some given *control design* of a system *satisfies* (implements) some given *requirements* on its behaviours. There are three approaches to tackle this problem depending on the formalisms used to describe the control design and the requirements. The first one is the *logic-based* approach where the design and the requirements are described by formulas φ_1 and φ_2 of a same logical language, and the verification problem is stated as the implication $\varphi_1 \Rightarrow \varphi_2$. The second approach is the *automata-based* approach where the design and the requirements are described using hybrid automata and the implementation relation is inclusion of languages (sets of trajectories). The third approach, the *automata-logic-based* approach, is a mixed approach where the design is described by an automaton $\mathcal{A}$ whereas the requirements are expressed by a formula φ, and the implementation relation is the inclusion of the language of $\mathcal{A}$ in the set of trajectories satisfying φ. Methods for solving the verification problem are either *deductive*, i.e., based on the use of proof systems, or *algorithmic*, i.e., based on applying decision procedures or approximate analysis techniques to solve reachability problems in the (sub)classes of hybrid automata under consideration.

Logic-based formalisms provide high-level, concise, and expressive description languages; they are naturally associated with deductive verification techniques. On the other hand, automata-based formalisms are more concrete and suitable for modeling the dynamics of a system, and they allow automatic verification techniques. Hence, it is interesting to combine the different approaches of specification and

* This work has been performed while this author was visiting Verimag.

verification to benefit from their respective advantages. The kernel of such a combination is the link between logics and automata. Indeed, it is possible to translate a verification problem from one approach to another one if it is possible to characterize formulas by automata and conversely. In particular, constructing for any given formula φ an automaton $\mathcal{A}_\varphi$ that recognizes exactly the set of trajectories satisfying φ (in the spirit of [17]), allows to apply the existing algorithmic analysis techniques of hybrid automata, e.g. see [1], for verification problems stated in the logic or the automata-logic based approaches. Moreover, this allows to identify decidable logical specification languages, namely those corresponding to automata with decidable emptiness problem. The problem we address in this work is to identify expressive logical languages that can be translated to hybrid automata, in particular, to subclasses of hybrid automata with decidable emptiness problem.

We introduce a logic called HATL which is an extension of the linear-time temporal logic with (hybrid) automata constraints. These constraints allow to say that the segment of a trajectory since some designated point in the past is accepted by a given linear hybrid automaton. The reference points are introduced using the *position quantification* "$u.$" which gives the current point the name u. For instance, the formula $\Box(P \Rightarrow u.\Diamond A^u)$ means that whenever P holds at some point, let us call it u, then there exists some future point such that the trajectory since u until that point is accepted by the automaton A. The logic HATL generalizes the timed logic TATL [8] where only timed automata [4] are used, as well as dense-time TPTL [6] and MITL [5].

Then, we define two fragments of HATL, called HATL$^+$ and HATL$^-$, such that the negation of each formula in HATL$^+$ is equivalent to a formula in HATL$^-$ and conversely, and show that for every formula φ in HATL$^-$, we can construct a linear hybrid (Büchi) automaton $\mathcal{A}_\varphi$ which recognizes the set of trajectories satisfying φ. This allows to reduce the verification problem of hybrid automata w.r.t. HATL$^+$ formulas to the emptiness problem of hybrid automata. The specialization of our reduction to timed automata and the timed logics TATL and TPTL allows to show that the verification problem of timed automata w.r.t. the fragments TATL$^+$ and TPTL$^+$ is decidable [8].

Finally, we consider the logic-based approach using the CoD. Fragments of this logic have been identified in [11] for the description of control systems and their requirements. We show that a verification problem stated in CoD using these fragments is decidable. To obtain this result, we define first two fragments of HATL called CDF (Control Design Formulas) and RF (Requirement Formulas), and prove that we can translate CDF formulas and negations of RF formulas to hybrid automata, reducing by that the verification problem of CDF formulas w.r.t. RF formulas to the emptiness problem of hybrid automata. Then, we define subclasses of CDF and RF subsuming their corresponding fragments of CoD, such that the obtained emptiness problem by our reduction consists of solving a decidable reachability problem (using [2]). These results generalize those in [9].

2 Preliminaries

2.1 Simple constraints

Let $\mathcal{V}$ be a set of variables ranging over reals ($\mathbb{R}$). A *simple constraint* over $\mathcal{V}$ is a boolean combination of constraints of the form $x \prec c$ where $x \in \mathcal{V}$, $\prec \in \{<, \leq\}$, and

c is an integer constant ($c \in \mathbb{Z}$). The symbols $<$ and $\leq$ represent the usual (strict and non-strict) ordering relations over $\mathbb{R}$. Let $\mathcal{C}_{\mathcal{V}}$ be the set of simple constraints over $\mathcal{V}$.

A *valuation* over $\mathcal{V}$ is a function in $[\mathcal{V} \rightarrow \mathbb{R}]$. A satisfaction relation is defined as usual between valuations and constraints. Given a valuation ν and $f \in \mathcal{C}_{\mathcal{V}}$, we denote by $\nu \models f$ the fact that ν satisfies f.

We denote by $\mathbf{0}$ the valuation that associates with each clock the value 0. Given a valuation ν and a set of variables $X \subseteq \mathcal{V}$, we denote by $\nu[X \mapsto 0]$ the valuation which associates with each variable in X the value 0, and coincides with ν on all the other variables.

2.2 States, interval sequences, and trajectories

Let $\mathcal{P}$ be a finite set of *atomic propositions*. A *state* is a subset of $\mathcal{P}$. Let Σ be the set of all possible states, i.e., $\Sigma = 2^{\mathcal{P}}$.

Let I be a nonempty subinterval of $\mathbb{R}_{\geq 0}$. Then, we denote respectively by $lb(I)$ and $ub(I)$ the lower and upper bounds of I. We denote also by $\ell(I)$ the length of the interval I, i.e., $\ell(I) = ub(I) - lb(I)$. Given two nonempty intervals I_1 and I_2, I_1 is called *adjacent to* I_2, if $I_1 \cap I_2 = \emptyset$ and $lb(I_2) = ub(I_1)$.

Let I be a left-closed nonempty subinterval of $\mathbb{R}_{\geq 0}$ such that $lb(I) = 0$. An *interval sequence* over I is a finite or infinite sequence of nonempty intervals $\vec{I} = I_0 I_1 \cdots I_i \cdots$ such that

- $\forall i < |\vec{I}|$, I_i is adjacent to I_{i+1},
- $\forall t \in I.\ \exists i < |\vec{I}|.\ t \in I_i$,
- If I is bounded (resp. unbounded), the sequence $\vec{I}$ is finite (resp. infinite).

A mapping $\tau : I \rightarrow \Sigma$ is called *finite-variable* if in every finite subinterval of I, τ changes its value a finite number of times (τ has a finite number of discontinuity points). We call *trajectory* over I any mapping $\tau : I \rightarrow \Sigma$ which is finite-variable. We say that a trajectory $\tau : I \rightarrow \Sigma$ is *bounded* (resp. *unbounded*) if the interval I is bounded (resp. unbounded).

Let $\tau : I \rightarrow \Sigma$ be a trajectory, and I' a bounded closed nonempty subinterval of I. Then, we denote by $\tau(I')$ the bounded trajectory $\tau' : [0, \ell(I')] \rightarrow \Sigma$ such that $\forall t \in [0, \ell(I')].\ \tau'(t) = \tau(t + lb(I'))$.

3 Linear Hybrid Automata

We define in this section linear hybrid automata on bounded or unbounded trajectories.

3.1 Linear hybrid transition tables

Recall that $\Sigma = 2^{\mathcal{P}}$ where $\mathcal{P}$ is a finite set of atomic propositions. A *linear hybrid transition table* $\mathcal{T}$ over Σ consists of the following components:

- $\mathcal{X}$, a finite set of variables,
- $\mathcal{Q}$, a finite set of control locations,
- $\mathcal{I}nit$, a set of initial control locations ($\mathcal{I}nit \subseteq \mathcal{Q}$),
- $\mathcal{E}$, a set of edges. Each edge is a triplet $E = (q, X, q')$ where $q, q' \in \mathcal{Q}$ are the source and target locations, and $X \subseteq \mathcal{X}$ is the set of reset variables,

- Π, a function in $[\mathcal{Q} \rightarrow \Sigma]$, associating a state with each control location,
- Γ, a function in $[\mathcal{Q} \rightarrow \mathcal{C}_{\mathcal{X}}]$, associating with each control location q a simple constraint under which the computation is allowed to stay in q.
- ∂, a function in $[\mathcal{Q} \times \mathcal{X} \rightarrow \mathbb{Z}]$, associating with each $q \in \mathcal{Q}$ and $x \in \mathcal{X}$, an integer rate at which x changes continuously while the control is at q.

A *clock* (resp. *integrator*) is a variable $x \in \mathcal{X}$ such that $\forall q \in \mathcal{Q}.\ \partial(q,x) = 1$ (resp. $\partial(q,x) \in \{0,1\}$). A *timed transition table* is a linear hybrid transition table such that all the variables in $\mathcal{X}$ are clocks.

Given a valuation ν, a control location q, and $t \in \mathbb{R}_{\geq 0}$, we denote by $[\nu+t]_q$ the valuation ν' such that $\forall x \in \mathcal{X}.\ \nu'(x) = \nu(x) + \partial(q,x)\cdot t$, i.e., the valuation obtained from ν by staying at q for an amount of time equal to t.

Let I be a left-closed subinterval of $\mathbb{R}_{\geq 0}$ such that $lb(I) = 0$. Then, a *run* of $\mathcal{T}$ over the interval I is a finite or infinite sequence $\rho = \langle q_0, \nu_0, I_0\rangle \cdots \langle q_i, \nu_i, I_i\rangle \cdots$ such that

- $q_0 \in \mathcal{I}nit$, $\nu_0 = \mathbf{0}$, and
- $I_0 I_1 \cdots I_i \cdots$ is an interval sequence over I, and
- $\forall i < |\rho|.\ \forall t \in I_i.\ [\nu_i + (t - lb(I_i))]_{q_i} \models \Gamma(q_i)$, and
- $\forall i < |\rho|.\ q_i = q_{i+1}$ and $\nu_{i+1} = [\nu_i + \ell(I_i)]_{q_i}$, or $\exists (q_i, X, q_{i+1}) \in \mathcal{E}.\ \nu_{i+1} = [\nu_i + \ell(I_i)]_{q_i}[X \mapsto 0]$.

With every run $\rho = \langle q_0, \nu_0, I_0\rangle \cdots \langle q_i, \nu_i, I_i\rangle \cdots$ defined as above, we associate a *path* of $\mathcal{T}$ over I which is a mapping $\theta : I \rightarrow \mathcal{Q}$ such that $\forall i < |\rho|.\ \forall t \in I_i.\ \theta(t) = q_i$. Then, the *trajectory* generated by ρ is defined as the mapping $\tau = \Pi \circ \theta$.

Deterministic linear hybrid transition tables can be defined straightforwardly (see the definition given in [8] for timed transition tables).

3.2 Linear Hybrid Automata on bounded trajectories

A bounded trajectories *linear hybrid automaton* (LHA) over Σ is a pair $\mathcal{A} = (\mathcal{T}, \mathcal{F})$, where $\mathcal{T} = (\mathcal{X}, \mathcal{Q}, \mathcal{I}nit, \mathcal{E}, \Pi, \Gamma)$ is a linear hybrid transition table over Σ, and $\mathcal{F} \subseteq \mathcal{Q}$ is a set of *final* control locations.

Let $\rho = \langle q_0, \nu_0, I_0\rangle \cdots \langle q_n, \nu_n, I_n\rangle$ be a finite run of $\mathcal{T}$. Then, we say that ρ is an *accepting run* of $\mathcal{A}$ if $q_n \in \mathcal{F}$. We denote by $L(\mathcal{A})$ the set of bounded trajectories generated by accepting runs of $\mathcal{A}$.

When $\mathcal{T}$ is a timed transition table, $\mathcal{A}$ is called *timed automaton* (TA). The automaton $\mathcal{A}$ is *deterministic* if its table $\mathcal{T}$ is deterministic. A *simple linear hybrid automaton* is a deterministic LHA such that all its variables are clocks except at most one which may be an integrator, and if it has an integrator, say y, then only control locations with no outgoing edges can have constraints on y, and every constraint on y appearing on any final location must be of the form $(y \sim c) \vee f$ where $\sim \in \{>, \geq\}$.

3.3 Linear Hybrid Automata on unbounded trajectories

A linear hybrid automaton on unbounded trajectories is defined, as in the case of bounded trajectories, by a transition table and an accepting condition. We consider here the 1, 2, and 3 acceptance conditions of the ω-automata theory [16]. For $i \in \{1,2,3\}$, an *i-linear hybrid automaton* over Σ (i-LHA) is a pair $\mathcal{A} = (\mathcal{T}, \mathcal{F})$ where

$\mathcal{T}$ is a linear hybrid transition table over Σ, $\mathcal{F} \subseteq \mathcal{Q}$ when $i \in \{1,2\}$, and $\mathcal{F} \subseteq 2^{\mathcal{Q}}$ when $i = 3$.

Given an infinite run $\rho = \langle q_0, \nu_0, I_0 \rangle \cdots \langle q_i, \nu_i, I_i \rangle \cdots$, we denote by $Loc(\rho)$ the set of all control locations q such that $q = q_i$ for some $i \geq 0$, and by $Inf(\rho)$ the set of all contol locations q such that $q = q_i$ for infinitely many i's. Then, we say that ρ is i-accepting if $Loc(\rho) \cap \mathcal{F} \neq \emptyset$ when $i = 1$, $Inf(\rho) \cap \mathcal{F} \neq \emptyset$ when $i = 2$, or $Inf(\rho) \in \mathcal{F}$ when $i = 3$. Notice that the 2-acceptance (resp. 3-acceptance) condition is the Büchi's (resp. Muller's) acceptance condition, whereas the 1-acceptance condition consists of a reachability condition.

As in the case of bounded trajectories, when $\mathcal{T}$ is a timed transition table, $\mathcal{A}$ is called *i-timed automaton* (i-TA), and an i-LHA is *deterministic* if its table $\mathcal{T}$ is deterministic. We call *linear hybrid graph* (resp. *timed graph*) a (1-)linear hybrid automaton (resp. (1-)timed automaton) with a trivial acceptance condition (i.e., $\mathcal{F} = \mathcal{Q}$). A 1-LHA is called *restricted* if all its variables are clocks except at most one which may be an integrator, and if it has an integrator, say y, then only control locations with no outgoing edges can have constraints on y, and every constraint on y appearing on any location in $\mathcal{F}$ must be of the form $(c_1 \sim y \sim c_2) \wedge f$ where $\sim \in \{\leq, <\}$.

4 Hybrid Automata Temporal Logic

We introduce the logic HATL which is an extension of the propositional temporal logic PTL [15] with bounded trajectories linear hybrid automata.

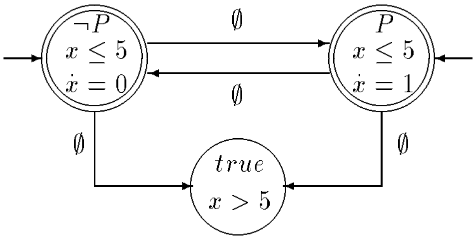
4.1 The logic HATL

Recall that $\mathcal{P}$ is a finite set of atomic propositions, and that $\Sigma = 2^{\mathcal{P}}$. We use letters $P, Q, \ldots$ to range over atomic propositions, and greek letters $\pi, \delta, \ldots$ to range over their boolean combinations. We introduce a set $\mathcal{W}$ of *position variables*, and use letters $u, v, \ldots$ to range over $\mathcal{W}$. Finally, we use $A, B, \ldots$ to range over bounded trajectories LHA's over Σ. The set of formulas of HATL is defined by:

$$\varphi ::= P \mid \upharpoonright\pi \mid u.\varphi \mid A^u \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\,\mathcal{U}\,\varphi$$

We consider as abbreviations the usual boolean connectives as conjunction ($\wedge$) and implication ($\Rightarrow$), as well as $\Diamond\varphi$ for $true\,\mathcal{U}\,\varphi$, and $\Box\varphi$ for $\neg\Diamond\neg\varphi$.

HATL formulas are interpreted on unbounded trajectories. The operators $\mathcal{U}$, $\Diamond$, and $\Box$, are the classical *until*, *eventually*, and *always* operators of PTL. The formula $\upharpoonright\pi$ is true at some point of a trajectory if π changes its value from false to true exactly at that point. Such a formula allow to refer to the starting point of some new phase. In the formula $u.\varphi$, the current time is associated with the position variable u. Then, subformulas of φ of the form A^u are used to express the fact that the bounded trajectory since the time associated with u is accepted by the automaton A. Formulas of the form A^u, which we call *automata constraints*, allow to express constraints on delays, i.e., time distances since some designated points in the past, or, more generally, on *durations* of state properties, i.e., accumulated times that some properties hold. For example, let B be the (deterministic) automaton represented by the following picture:



Then, the formula $u.\Diamond(Q \wedge B^u)$ expresses the fact that "*Q will be eventually true while the duration of P is less than* 5 *time units*".

In the formula $u.\varphi$ (e.g., $u.\Diamond(Q \wedge B^u)$), the construction "$u.$", called *position quantification*, binds the position variable u in the subformula φ. 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 *open* if it contains free variables, otherwise it is *closed*.

The formal semantics of HATL is defined using a satisfaction relation $\models$ between unbounded trajectories, time values, and formulas. Since formulas may be open, the satisfaction relation between a trajectory τ, a time value t, and a formula φ, is defined w.r.t. a *position association* E that associates with each position variable u appearing in φ the time (which is less or equal than t) where u has been introduced. Then, the position association E is updated along the trajectory τ due to the introduction of new position variables. We denote by $E[u \mapsto t]$ the position association whose domain is the extension of the domain of E by u, and which associates with u the time value t and coincides with E on all the other position variables.

Let τ be an unbounded trajectory. For every $t \in I\!R_{\geq 0}$, every position association E (such that for every u in the domain of E, $0 \leq E(u) \leq t$), and every HATL formula φ, we define the meaning of $\langle \tau, t \rangle \models_E \varphi$ inductively on the structure of φ.

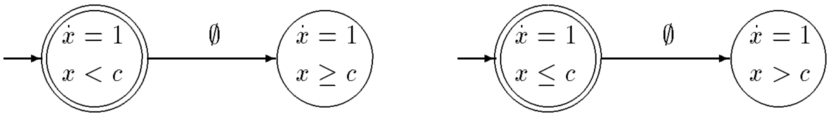
$$
\begin{array}{ll}
\langle \tau, t \rangle \models_E P & \text{iff } P \in \tau(t) \\
\langle \tau, t \rangle \models_E \upharpoonleft\!\pi & \text{iff } \exists \epsilon > 0.\ \forall t' \in (t, t+\epsilon).\ \langle \tau, t' \rangle \models_E \pi, \text{ and} \\
 & \quad \forall t' \in I\!R_{\geq 0} \cap (t - \epsilon, t).\ \langle \tau, t' \rangle \models_E \neg\pi \\
\langle \tau, t \rangle \models_E u.\varphi & \text{iff } \langle \tau, t \rangle \models_{E[u \mapsto t]} \varphi \\
\langle \tau, t \rangle \models_E A^u & \text{iff } \tau([E(u), t]) \in L(A) \\
\langle \tau, t \rangle \models_E \neg\varphi & \text{iff } \langle \tau, t \rangle \not\models_E \varphi \\
\langle \tau, t \rangle \models_E \varphi_1 \vee \varphi_2 & \text{iff } \langle \tau, t \rangle \models_E \varphi_1 \text{ or } \langle \tau, t \rangle \models_E \varphi_2 \\
\langle \tau, t \rangle \models_E \varphi_1 \mathcal{U} \varphi_2 & \text{iff } \exists t' \geq t.\ \langle \tau, t' \rangle \models_E \varphi_2, \text{ and } \forall t'' \in [t, t').\ \langle \tau, t'' \rangle \models_E \varphi_1
\end{array}
$$

When φ is a closed formula, we omit the position association E, and write simply $\langle \tau, t \rangle \models \varphi$. Given a closed formula φ, we denote by $[\![\varphi]\!]$ the set of unbounded trajectories τ such that $\langle \tau, 0 \rangle \models \varphi$. The formula φ is *satisfiable* iff $[\![\varphi]\!] \neq \emptyset$; it is *valid* iff $[\![\varphi]\!]$ is the set of all possible unbounded trajectories.

4.2 Sublogics

The logic TATL introduced in [8] is obtained by imposing the restriction that only timed automata can be used to express automata constraints.

Moreover, the (dense-time) TPTL [6] corresponds to the fragment of TATL where automata constraints involve only the timed automata $T_{<c}$ and $T_{\leq c}$, where c is some natural constant, that are represented respectively by the following pictures:



Notice that $T_{<c}$ and $T_{\leq c}$ are deterministic and that the automata $T_{\geq c}$ and $T_{>c}$ can be respectively obtained from them by inversing final and nonfinal locations.

Finally, the logic MITL [5] is a sublogic of TPTL obtained by restraining the use of position quantification (called reset quantification in TPTL) and constraints: time constraints can only be expressed using the constrained until operator $\mathcal{U}_I$ where I is a *nonponctual* interval with integer bounds.

4.3 The fragments HATL$^+$ and HATL$^-$

We introduce hereafter syntactical fragments of the logic HATL. In these fragments, we use two new operators $\widetilde{\mathcal{U}}$ and $\overline{\mathcal{U}}$ defined by:

$$\varphi_1 \widetilde{\mathcal{U}} \varphi_2 = (\varphi_1 \vee \varphi_2) \mathcal{U} \varphi_2 \qquad \varphi_1 \overline{\mathcal{U}} \varphi_2 = \varphi_1 \mathcal{U} (\varphi_1 \wedge \varphi_2)$$

A motivation for introducing these operators can be found in [8]. Notice that $\Diamond\varphi$ is equivalent to $true\widetilde{\mathcal{U}}\varphi$ as well as to $true\overline{\mathcal{U}}\varphi$. Then, the fragment HATL$^+$ consists of the set of formulas φ defined by:

$$\varphi ::= \psi \mid u.\varphi \mid \neg A^u \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \Box\varphi \mid \varphi \widetilde{\mathcal{U}} \psi \mid \varphi \overline{\mathcal{U}} \psi$$

and the fragment HATL$^-$ corresponds to the set of formulas φ defined by:

$$\varphi ::= \psi \mid u.\varphi \mid A^u \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \Box\psi \mid \psi \widetilde{\mathcal{U}} \varphi \mid \psi \overline{\mathcal{U}} \varphi$$

where the formulas ψ are defined by:

$$\psi ::= P \mid \uparrow\pi \mid A^u, A \text{ is deterministic } \mid \neg\psi \mid \psi \vee \psi \mid \psi \widetilde{\mathcal{U}} \psi \mid \psi \overline{\mathcal{U}} \psi \tag{1}$$

We consider also the fragments TATL$^+$, TPTL$^+$, TATL$^-$, and TPTL$^-$ defined in the obvious way.

We can prove that HATL$^-$ characterizes exactly the complements of the HATL$^+$ definable sets of trajectories, and conversely. It is easy to see that this fact holds also when we consider the corresponding fragments of TATL and TPTL.

Proposition 4.1 *For every HATL$^+$ (resp. TATL$^+$, TPTL$^+$) closed formula φ, we can construct an HATL$^-$ (resp. TATL$^-$, TPTL$^-$) closed formula φ' such that $[\![\neg\varphi]\!] = [\![\varphi']\!]$, and conversely.*

4.4 Expressiveness

The fragments TPTL$^+$, TATL$^+$, and HATL$^+$ allow to express the important classes of timed and duration properties that are the *constrained invariance properties* corresponding to formulas of the form

$$u_0.\,\Box(A^{u_0}_{(0,0)} \Rightarrow \cdots u_n.\,\Box(\bigwedge_{i=0}^{n} A^{u_i}_{(i,n)} \Rightarrow v.\,\Box(\bigwedge_{i=0}^{n} B_i^{u_i} \wedge C^v))\cdots)$$

and the *constrained response properties* corresponding to formulas of the form:

$$u.\,\Box(A^u \Rightarrow v.\,\Diamond(B_0^u \wedge C_0^v \wedge \Diamond(B_1^u \wedge C_1^v \wedge \cdots \Diamond(B_n^u \wedge C_n^v)) \cdots))$$

As for the fragments TPTL$^+$, TATL$^+$, and HATL$^+$, they allow to express the dual properties that are *constrained eventuality* and *constrained persistence properties.*

The fragment TATL$^+$ (resp. TATL$^-$) is more expressive than TPTL$^+$ (resp. TPTL$^-$) since the latter cannot express all the properties that are definable by means of automata constraints. Moreover, HATL$^+$ (resp. HATL$^-$) is more expressive than TATL$^+$ (resp. TATL$^-$) since, for instance, it allows to reason about duration properties, in the style of DTL [7].

We can show also that the logic MITL, which is less expressive than 2-TA's [5], is comparable with none of the fragments TPTL$^+$, TATL$^+$, TPTL$^+$, and TATL$^-$. Indeed, these fragments allow ponctual time constraints as well as overlaping of time constraints using position (reset) quantification. On the other hand, MITL allows to express properties with nested constrained until operators that are expressible in TPTL$^-$ but not in TPTL$^+$, whereas their negations are expressible in TPTL$^+$ but not in TPTL$^-$. The same reasons make the logic EMITL [18] not comparable with the four fragments mentioned above.

5 Logics vs. Automata

We show the links between HATL$^+$, HATL$^-$, and linear hybrid automata. First of all, we prove that deterministic 3-LHA's can be characterized by HATL$^+$ as well as by HATL$^-$ closed formulas.

Theorem 5.1 *For every deterministic 3-LHA (resp. 3-TA) $\mathcal{A}$, there exists a closed formula φ in HATL$^+$ and HATL$^-$ (resp. TATL$^+$ and TATL$^-$) with $L(\mathcal{A}) = [\![\varphi]\!]$.*

Indeed, let $\mathcal{A} = (\mathcal{T}, \mathcal{F})$ be a deterministic linear hybrid (resp. timed) Muller automaton over $\Sigma = 2^{\mathcal{P}}$, with $\mathcal{F} = \{\Delta_1, \cdots, \Delta_n\}$. For every control location q in $\mathcal{A}$, we denote by A_q the deterministic timed automaton $(\mathcal{T}, \{q\})$ which accepts the set of bounded trajectories reaching q. Then, it can easily be seen that the set of trajectories recognized by the automaton $\mathcal{A}$ is characterized by the formula

$$u.\,\bigvee_{i=1}^{n}((\bigwedge_{q\in\Delta_i} \Box\Diamond A_q^u) \wedge (\bigwedge_{q\notin\Delta_i} \neg\Box\Diamond A_q^u)) \qquad (2)$$

which is in both fragments HATL$^+$ and HATL$^-$ (resp. TATL$^+$ and TATL$^-$).

Let us address now the problem of characterizing formulas by automata. It can be shown that HATL$^+$ cannot in general be characterized by linear hybrid automata (with finitely many variables). On the other hand, we show that every HATL$^-$ closed formula can be characterized by a 2-LHA (linear hybrid Büchi automaton). This allows to reduce the verification problem of 2-LHA's w.r.t. HATL$^+$ closed formulas to the emptiness problem of 2-LHA's.

Theorem 5.2 *For every HATL$^-$ closed formula φ, we can construct a 2-LHA $\mathcal{A}_\varphi$ such that $L(\mathcal{A}_\varphi) = [\![\varphi]\!]$.*

This construction is possible because HATL$^-$ formulas allow to consider on trajectories only a finite number of points (positions) where automata constraints can be initialized. Indeed, we can show that each HATL$^-$ formula can be transformed into an equivalent formula in *normal form*, i.e., a disjunction of formulas of the form

$$u_0 .\, (\psi_0 \wedge (\psi_0' \widetilde{\mathcal{U}}_{>0}\, u_1 .\, (\psi_1 \wedge (\psi_1' \widetilde{\mathcal{U}}_{>0} \cdots u_n .\, (\psi_n \wedge (\psi_n' \widetilde{\mathcal{U}}_{>0}\, u_{n+1} .\, \psi_{n+1})) \cdots))))$$

where the ψ_i's and the ψ_i''s are of the form given by (1).

Then, given a closed formula φ in normal form, we proceed as follows: In a first step, we abstract from each automata constraint A^u by replacing it by a new atomic proposition $[A^u]$. The obtained formula φ^* is in the propositional temporal logic. Therefore, using the construction of [17], we get a Büchi automaton $\mathcal{B}_{\varphi^*}$ which characterizes φ^*. Interpreted as a Büchi LHA (by adding trivial timing constraints), this automaton accepts the trajectories that are propositionally consistent with φ. In a second step, we constrain $\mathcal{B}_{\varphi^*}$ by an LHA $\mathcal{B}_{const}$ which is constructed from the automata constraints appearing in φ. This automaton ensures that whenever the proposition $[A^u]$ appears at some point of a trajectory, then the automaton A accepts indeed the bounded trajectory since the point associated with u.

The translation from HATL$^-$ formulas to 2-LHA's outlined above yields a 2-TA when applied to TATL$^-$ formulas. Therefore, we obtain the following result.

Theorem 5.3 *For every TATL$^-$ closed formula φ, we can construct a 2-TA $\mathcal{A}_\varphi$ such that $L(\mathcal{A}_\varphi) = [\![\varphi]\!]$.*

6 Verification problem: the automata-logic-based approach

Let $\mathcal{A}_S$ be a 2-LHA that models the behaviours of some hybrid system, and let φ be an HATL$^+$ closed formula expressing its specifications. Then, the problem whether $\mathcal{A}_S$ satisfies φ, i.e., whether $L(\mathcal{A}_S) \subseteq [\![\varphi]\!]$, can be reduced using Proposition 4.1 and Theorem 5.2 to the emptiness problem $L(\mathcal{A}_S \times \mathcal{A}_{\neg\varphi}) = \emptyset$. Since 2-LHA are closed under product, we have the following fact.

Theorem 6.1 *The verification problem of 2-LHA's w.r.t. HATL$^+$ closed formulas is reducible to the emptiness problem of 2-LHA's.*

Using Theorem 5.3 and the same arguments as for Theorem 6.1, we can reduce the verification problem of 2-TA's w.r.t. TATL$^+$ closed formulas to the emptiness problem of 2-TA's. Since this latter is decidable [4], we obtain the following result.

Theorem 6.2 *The verification problem of 2-TA's w.r.t. TATL$^+$ closed formulas is decidable.*

7 Control Design and Requirement Formulas

We consider now the logic-based approach where the system as well as its specifications are expressed using formulas.

We introduce hereafter two subsets of HATL called CDF (Control Design Formulas) and RF (Requirement Formulas). Formulas in CDF and RF are positive boolean combinations of invariance formulas of the general form

$$\Box(\psi \Rightarrow u.\, \Box(A^u \Rightarrow \bigvee_{j=1}^{m} B_j^u)) \qquad (3)$$

where ψ stands for a formula of the form π or $\uparrow\pi$, and such that the formula ψ, the automaton A, and the automata B_j's satisfy some conditions we define below.

Control Design Formulas: Given an automaton A, the *prolongation* of A is an automaton $\vec{A}$ which accepts all the bounded trajectories having a prefix accepted by A. The construction of this automaton is straightforward. Now, we say that the automaton A is π-alternation-free if, when we consider the control transition graph associated with its transition table, all the paths relating an initial location with a final one do not contain any occurrence of a transition from π to $\neg\pi$ which is followed by an occurrence of a transition from $\neg\pi$ to π. This constraint ensures that for every trajectory τ, $\forall t_1, t_2 \in I\!R^+$ such that $t_1 \leq t_2$ and $\tau[t_1, t_2] \in L(A)$, and $\forall t \in [t_1, t_2)$, if $\langle \tau, t\rangle \models \uparrow\neg\pi$ then $\forall t' \in (t, t_2)\langle \tau, t'\rangle \not\models \uparrow\pi$. Then, CDF formulas are defined as positive boolean combinations of formulas of the form (3) where

- ψ is of the form $\uparrow\pi$,
- A is timed automaton $T_{>c}$ for some natural constant c,
- the B_j's are prolongations of deterministic π-alternation-free automata.

A *pure-time* CDF formula is a CDF formula which is a TATL formula, i.e., where only timed automata are used in automata constraints.

Requirement Formulas: RF formulas are defined as positive boolean combinations of formulas of the form (3) where the automata B_j's are deterministic. Notice that every RF formula is in HATL$^+$. A *simple* RF formula is a conjunction of formulas of the form (3) where A is a timed automaton, and all the B_j's are timed automata except at most one which may be a simple LHA.

8 From CDF and RF to Linear Hybrid Automata

We show hereafter translations from the fragments CDF and RF to linear hybrid automata. We start by showing that for every CDF formula φ, we can construct an automaton $\mathcal{A}_\varphi$ which recognizes $[\![\varphi]\!]$. Consider a formula φ of the general form $\Box(\uparrow\pi \Rightarrow u.\, \Box(T^u_{>c} \Rightarrow \bigvee_{j=1}^{m} B_j^u))$. Intuitively, an automaton that recognizes $[\![\varphi]\!]$ does the following. Given a trajectory τ, it checks for each time t where $\uparrow\pi$ holds and each $t' > t + c$, whether the trajectory $\tau[t, t']$ is accepted by some B_j. We can show that in general (without any restriction on the B_j's and π), this automaton would need infinitely many variables because the number of points where $\uparrow\pi$ may hold in a bounded interval (of 1 time unit, say) is unbounded, and since for each point t where $\uparrow\pi$ holds, the automaton has to check that for every $t' > t + c$, some B_j accepts $\tau[t, t']$. However, if φ is a CDF formula, then we can prove that there are deterministic π-alternation-free automata C_j's contructed from the B_j's such that, "*for every $t' > t + c$, there exists B_j accepting $\tau[t, t']$*" if and only if "*there exists some length d and some automaton C_j which accepts $\tau[t, t + d]$*". Hence, we can construct an automaton characterizing φ which repetitively looks for the next point where $\uparrow\pi$ holds and then guesses d and C_j.

Theorem 8.1 *For every CDF formula φ, we can construct a linear hybrid graph $\mathcal{A}_\varphi$ such that $L(\mathcal{A}_\varphi) = [\![\varphi]\!]$. Moreover, if φ is a pure-time CDF formula, then $\mathcal{A}_\varphi$ is a timed graph.*

Now, since RF is a subset of HATL$^+$, by Proposition 4.1 and Theorem 5.2 we deduce that for every RF formula φ we can construct a linear hybrid Büchi automaton $\mathcal{A}_{\neg\varphi}$ which recognizes $[\![\neg\varphi]\!]$. It can be seen that the specialization of the construction used in Theorem 5.2 to the cases of RF formulas and simple RF formulas leads to the following fact.

Theorem 8.2 *For every RF formula φ, we can construct a 1-LHA $\mathcal{A}_{\neg\varphi}$ such that $L(\mathcal{A}_{\neg\varphi}) = [\![\neg\varphi]\!]$. Moreover, if φ is a simple RF formula, then $\mathcal{A}_{\neg\varphi}$ is a restricted 1-LHA.*

9 Verification problem: the logic-based approach

Let φ_1 be a CDF formula, and φ_2 be an RF formula. Deciding whether $\varphi_1 \Rightarrow \varphi_2$ is valid consists in checking whether $[\![\varphi_1]\!] \subseteq [\![\varphi_2]\!]$, or equivalently, whether $[\![\varphi_1]\!] \cap [\![\neg\varphi_2]\!] = \emptyset$. Then, by Theorems 8.1 and 8.2, and since 1-LHA's (resp. restricted 1-LHA's) are closed under product with linear hybrid graphs (resp. timed graphs), we deduce the following fact.

Theorem 9.1 *The verification problem of CDF (resp. pure-time CDF) formulas w.r.t. RF (resp. simple RF) formulas is reducible to the emptiness problem of 1-LHA's (resp. restricted 1-LHA's).*

The emptiness problem of 1-LHA is undecidable [12], however, it is decidable for restricted 1-LHA's (using [2]). Hence, we obtain the following result.

Theorem 9.2 *The verification problem of pure-time CDF formulas w.r.t. simple RF formulas is decidable.*

Consider now the fragments of CoD defined in [11] to describe control designs and their requirements. We show that the verification problem stated as implication between two formulas in these fragments is decidable. For that, we show that the CoD formulas we are interested in can be translated to pure-time CDF and simple RF formulas.

Proposition 9.1 *For every CoD control design (resp. CoD requirement) formula, there exists an equivalent pure-time CDF (resp. simple RF) formula.*

Using Proposition 9.1 and Theorem 9.2, we obtain the following result.

Theorem 9.3 *The verification problem for CoD control design formulas w.r.t. CoD requirement formulas is decidable.*

10 Conclusion

We provided a framework for the specification and the verification of hybrid systems. We introduced a general specification logic HATL which is an extension of the temporal logic allowing constraints on bounded trajectories expressed by means of hybrid automata. We have shown how to translate verification problems stated in the automata-logic-based or in the logic-based approaches into the automata-based approach in order to apply algorithmic analysis techniques on automata. These translations are based on effective constructions of timed or hybrid automata from formulas in some fragments of HATL. Using these translations, we proved the decidability of the verification problems of timed Büchi automata w.r.t. TATL$^+$ formulas, and of pure-time CDF formulas w.r.t. simple RF formulas.

The fragment TATL$^+$ is a fairly powerful specification language since it subsumes deterministic timed Muller automata, and allows the expression of response as well as invariance timed properties. Its sublanguage TPTL$^+$ constitues a fragment of TPTL whose verification problem is decidable. Moreover, TATL$^+$ (as well as TPTL$^+$) is not comparable with the known logics whose verification problem is decidable, as MITL and EMITL.

Pure-time CDF and simple RF are more expressive than the fragments of CoD that have been suggested to specify control systems and requirements. Thus, our results show, in particular, that the verification of CoD control designs w.r.t. CoD requirement formulas is decidable. This is the first decidability result concerning fragments of (dense time) CoD that are not purely propositional.

References

1. R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The Algorithmic Analysis of Hybrid Systems. *TCS*, 138, 1995.
2. R. Alur, C. Courcoubetis, and T. Henzinger. Computing Accumulated Delays in Real-time Systems. In *CAV'93*. LNCS 697, 1993.
3. R. Alur, C. Courcoubetis, T. Henzinger, and P-H. Ho. Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems. In *Hybrid Systems*. LNCS 736, 1993.
4. R. Alur and D. Dill. A Theory of Timed Automata. *TCS*, 126, 1994.
5. R. Alur, T. Feder, and T. Henzinger. The Benefits of Relaxing Punctuality. In *PODC'91*, 1991.
6. R. Alur and T. Henzinger. A Really Temporal Logic. In *FOCS'89*. IEEE, 1989.
7. A. Bouajjani, R. Echahed, and J. Sifakis. On Model Checking for Real-Time Properties with Durations. In *LICS'93*. IEEE, 1993.
8. A. Bouajjani and Y. Lakhnech. Temporal Logic + Timed Automata : Expressiveness and Decidability. In *CONCUR'95*. LNCS 962, 1995.
9. A. Bouajjani, Y. Lakhnech, and R. Robbana. From Duration Calculus to Linear Hybrid Automata. In *CAV'95*. LNCS 939, 1995.
10. Z. Chaochen, C.A.R. Hoare, and A.P. Ravn. A Calculus of Durations. *IPL*, 40, 1991.
11. J. He, C.A.R. Hoare, M. Fränzle, M. Müller-Olm, E.R. Olderog, M. Schenke, M.R. Hansen, A.P. Ravn, and H. Rishel. Provably Correct Systems. In *FTRTFT'94*. LNCS 863, 1994.
12. Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine. Integration Graphs: A Class of Decidable Hybrid Systems. In *Hybrid Systems*. LNCS 736, 1993.
13. O. Maler, Z. Manna, and A. Pnueli. From Timed to Hybrid Systems. In *REX workshop on Real-Time: Theory and Practice*. LNCS 600, 1992.
14. X. Nicollin, J. Sifakis, and S. Yovine. From ATP to Timed Graphs and Hybrid Systems. In *REX workshop on Real-Time: Theory and Practice*. LNCS 600, 1992.
15. A. Pnueli. The Temporal Logic of Programs. In *FOCS'77*. IEEE, 1977.
16. W. Thomas. Automata on Infinite Objects. In *Handbook of Theo. Comp. Sci.* Elsevier Sci. Pub., 1990.
17. M.Y. Vardi and P. Wolper. An Automata-Theoretic Approach to Automatic Program Verification. In *LICS'86*. IEEE, 1986.
18. Th. Wilke. Specifying Timed State Sequences in Powerful Decidable Logics and Timed Automata. In *FTRTFT'94*. LNCS 863, 1994.