Uniformity for the Decidability of Hybrid Automata

Olivier Roux*, Vlad Rusu**

Abstract. We present some new decidability results on the verification of hybrid automata by symbolic analysis (abstract interpretation using polyhedra). The results include defining a class of hybrid automata for which all properties expressed in the real-time temporal logic TCTL are decidable. The obtained class of automata is shown powerful enough to model reactive applications in which every task eventually terminates within uniformly-bounded time. Indeed, the restrictions we use for obtaining decidability have a physical meaning, and they all impose some kind of uniformity to the runs of hybrid automata.

Keywords: decidability, uniformity, hybrid automata, symbolic verification, temporal logic TCTL

1 Introduction

The verification of concurrent programs is often achieved by modeling programs as transition systems, and by model-checking temporal logic formulas for the desired properties. This approach is acceptable for systems whose time durations and delays are not relevant; but it does not work when one wants to take into account temporal durations of progams in a dense time.

Thus, *timed automata* [ACD90] and *hybrid automata* [ACH+95] arose, allowing specification of both continuous evolutions (such as the passing of time) and discrete changes. Timed automata are quite adapted for the verification of real-time logic properties (e.g. TCTL [ACD90, HNSY94]). But such properties are only *semi-decidable* for hybrid automata [ACH+95]; and even in the decidable cases, the verification algorithms are not very efficient.

For these reasons, the verification problem has been attacked from two directions. The first one is to obtain restricted, decidable classes of hybrid automata [KPSY93, PV94, HKPV95, BER94, BES93, HH95, Hen95, RR94, RR95]. The second consists in using abstractions [HPR94] or compositionality and on-the-fly techniques [BGK+96] to accelerate reaching a result.

Our current proposal lies in the first of these two directions, and it aims at obtaining decidability for classes of reactive and hybrid systems that are useful in practice, by imposing restrictions that have a physical meaning.

We obtain results in two steps:

* Institut Universitaire de France and Laboratoire d'Automatique de Nantes

** Laboratoire d'Automatique de Nantes (CNRS, Univ.Nantes, Ecole Centrale de Nantes) 44072 NANTES Cedex 03, France. E-mail: {roux|rusu}@lan.ec-nantes.fr

– first, we prove that the time-bounded fragment of logic TCTL is decidable on *strongly-non-Zeno* hybrid automata (a relevant subclass of hybrid automata);
– second, we investigate an even more restrictive class of hybrid automata (namely the *uniformly-periodic*) for which the whole logic TCTL is decidable.

These results hold because we progressively introduce some structure (we call it *uniformity*) in the systems, that limits unnatural behaviours and provides decidability: the more uniformity, the more decidability.

As an application, we show that reactive programs in which events do not occur too often so that every task eventually terminates within a uniformly-bounded time, can be modeled by *strongly non-Zeno*, *uniformly periodic* hybrid automata. Decidability of bounded-TCTL then allows to verify such properties as *bounded response* (e.g. 'the system responds within 3 time units'); and the decidability of full TCTL makes it possible to check for *safety* as well as performing the so-called *parametric verification*: instead of verifying a time-bounded property with a given bound, rather synthethise the bound from the program's model.

The rest of the paper is organized as follows. In the next section, we recall the fundamentals of hybrid automata. Section 3 deals with the definition and the decidability of bounded-TCTL for strongly-non-Zeno hybrid automata, together with a decidable equivalent criterion for the strongly-non-Zeno condition, followed by a generalization of this class. Similarly, section 4 contains the definition and the decidability of full TCTL for hybrid automata that are both uniformly-periodic and strongly-non-Zeno, together with a sufficient criterion for uniform periodicity, and followed by a generalization. An experiment is reported in section 5, and we conclude in section 6.

2 Hybrid automata

We use hybrid automata [ACH+95] as a model for reactive applications that consist in lengthing *tasks* that can be preempted and/or launched by instantaneous *events*. We limit ourselves to *asynchronous* systems[3] in the sense that tasks are lengthing and that two events cannot occur at the same time instant[4]. As we shall see in the following, the obtained results apply also to the more general class of *hybrid systems*.

Example. Figure 1 is a graphical representation of a hybrid automaton that describes the iteration of two tasks `A`, `B`. Task `B` is executed either after `A` completes, or following an event `x` that preempts task `A` and launches `B`; when `B` completes, `A` will be resumed at preemption point.

The duration of task `A` (`B`) is any real in the interval $[7, 8)$, and respectively the interval $[4, 5)$; and there is a minimum delay of 3 time units between two consecutive events `x`.

[3] in opposition to the *synchronous* approach, [Hal93]
[4] this is the approach taken in ELECTRE language [RCCE92]

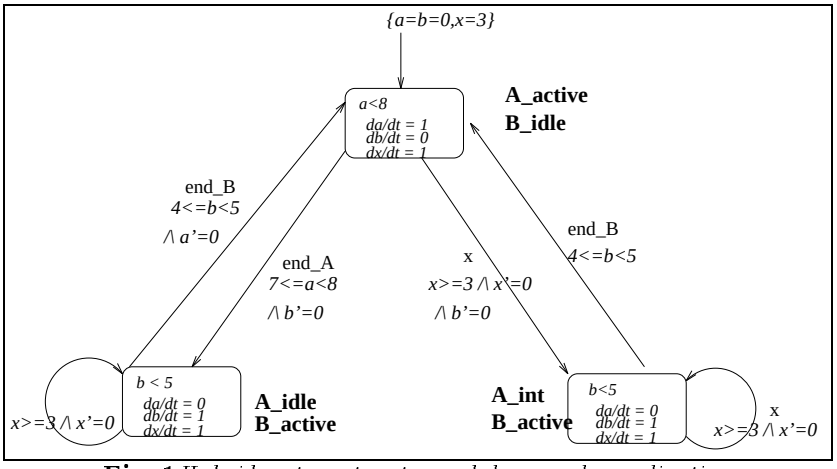

Fig. 1 *Hybrid automaton to model example application*

The states (called *locations*) are labeled with propositions — like **A_active**, **A_int**, **B_idle** — defining at each location which tasks are *active* (executing), *interrupted* (not executing, following some event that has preempted it) or *idle* (neither active nor interrupted). The transitions (called *edges*) are labeled with the possible discrete changes between locations: events or task completions.

There are three *variables* a, b and x, to count the time for tasks `A`, `B` and event `x`. Variables evolve at locations following simple *differential laws* - like $da/dt = 1, db/dt = 0$ at the upmost location, meaning that *task* `A` *is active (running)* and *task* `B` *is idle*. The inequalities associated to locations and edges express durations for tasks and events: for instance, when entering the upmost location with variable $a = 0$, we start counting time for task `A`, and control has to leave the location between 7 and 8 time units after that, by the inequality $7 \leq a < 8$ on the outcoming edge. We now make precise these intuitions.

Syntax of hybrid automata. A hybrid automaton is a sextuple $(X, L, E, \mathcal{I}, \Delta, \mathcal{R})$:

- $X = \{x_1, \ldots x_n\}$ is a finite set of *variables*;
- L is a finite set of *locations*;
- $E \subseteq L \times A \times L$ is a finite set of *edges* (A is an edge-labeling alphabet);
- $\mathcal{I}$ is a function that associates to each location $l \in L$, an *invariant* $\mathcal{I}_l$ which is a system of linear inequalities on variables X;
- Δ is a function that that associates to each location $l \in L$ and each variable $x \in X$, a *differential law* of the form $\frac{dx}{dt} \in [k_{l,x}, K_{l,x}]$ with $k_{l,x}, K_{l,x} \in \mathbb{Q}$;
- $\mathcal{R}$ is a function that maps each edge $e \in E$ to a *relation* $\mathcal{R}_e$, which is a a system of linear inequalities on variables $X \cup X'$ (where $X' = \{x' | x \in X\}$).

Semantics. A *state* (l, v) of a hybrid automaton consists of a location $l \in L$ and a real valuation of the variables $v : X \rightarrow \mathbb{R}$, such that the valuation satisfies the location's invariant: $\mathcal{I}_s(v) = true$ [5].

[5] formally, $\mathcal{I}_s(v)$ denotes the truth value obtained by replacing in the system of linear inequalities $\mathcal{I}_s$, each variable x by its value $v(x)$

A *continuous step* consists in staying at some location l and letting the variables evolve following the corresponding laws (see figure 2(a)) inside the location's invariant. That is, we have a continuous step denoted $(l, v) \xrightarrow{t,f} (l, v')$ if: $t > 0$ and $f = (f_x)_{x \in X}$ is a n-uple of piece-wise continuously differentiable functions, such that: all the derivative values df_x/dt lie in the corresponding interval $[k_{l,x}, K_{l,x}]$; the final values of variables are $v'(x) = v(x) + f_x(t)$; and all the intermediary values of variables satisfy the location's invariant. The real $t > 0$ is the *duration* of the continuous step.

A *discrete step* consists in crossing an edge $(l, a, l') \in E$ such that the values of variables before and after the edge is crossed, satisfy the edge's relation (see figure 2(b)). So we have a discrete step denoted $(l, v) \xrightarrow{e} (l', v')$ if: the edge's relation $\mathcal{R}_e(v, v') = true$ [6]. The duration of a discrete step is 0.

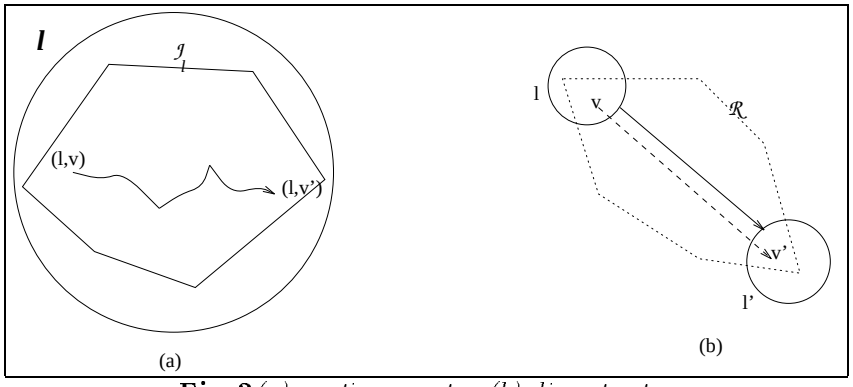

Fig. 2 *(a) continuous step (b) discrete step*

A *run* $\rho = (l_0, v_0) \rightarrow (l_1, v_1) \rightarrow \ldots \rightarrow (l_i, v_i) \rightarrow (l_{i+1}, v_{i+1}) \ldots$ is a finite or infinite sequence of steps, such that: all continuous steps have strictly positive durations (by definition); the first step is continuous; and there is at least one continuous step between any two consecutive discrete steps. These restrictions are not usually considered in the literature, but they are imposed by our *asynchronous* context and they will later prove to be useful.

The *duration* of a run is the sum of the durations of all its component steps. A run is *divergent* if its duration is infinite.

2.1 Symbolic analysis

The verification problem for hybrid automata consists in answering questions of the form:

(α) 'starting from an initial set of states, is it possible (or inevitable) to reach some final set of states, in time bounded by constant c (or unbounded time)'

[6] formally, $\mathcal{R}_e(v, v')$ denotes the truth value obtained by replacing in the system of linear inequalities $\mathcal{R}_e$, each variable x by its value $v(x)$ and each variable x' by its value $v'(x')$

It turns out that most practically interesting properties that arise in the analysis of reactive and hybrid systems are of the previous form. But it is obvious that this kind of property cannot be verified directly, since the state space of hybrid automata is continuously infinite. To deal with this, the so-called *symbolic analysis* or symbolic model-checking technique [HNSY94] proposes to represent infinite sets of states by so-called *symbolic states* and infinite sets of runs by *symbolic runs*. A symbolic state is defined by a location and a domain of values for variables indentifiable with a polyhedron in $\boldsymbol{R}^n$ (such as the pairs (l_1, φ_1), (l_2, φ_2) and (l_3, φ_3) in figure 3); and a symbolic run is a sequence of symbolic states:

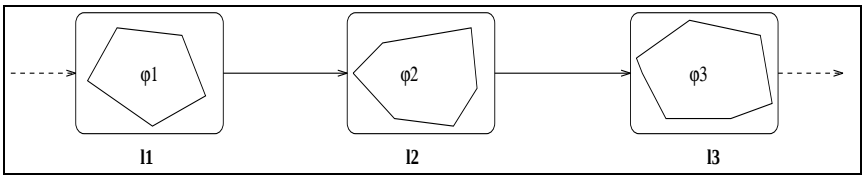

Fig. 3 *Three symbolic states and a symbolic run*

A symbolic run simulates all the runs starting from an initial set of states, that cross a given sequence of edges[7]. It has been shown that symbolic runs can be computed algorithmically by iterating so-called *continous successors* and *discrete successors* of symbolic states [ACH+95]. Computing the continuous and discrete successors comes to performing geometrical operations on n-dimensional polyhedra [HPR94].

So, in order to answer questions of the form (α) above, one should basically compute all the symbolic runs that start from the initial set of states, and see whether the final set of states is reached, by one or by all the symbolic runs. This is the so-called *symbolic forward analysis* algorithm of [ACH+95]. But another serious problem arises: this algorithm is not guaranteed to terminate ever; indeed, properties of the form (α) are only *semi-decidable* on general hybrid automata, and undecidability is shown to affect even quite simple automata [KPSY93, HKPV95]. This is no surprise, because hybrid automata, seen as machines, are extremely powerful (when compared to Turing or to n-counter machines).

However, some partial decidability results exist [KPSY93, PV94, HKPV95, BER94, BES93, HH95, Hen95]. All consist in restricting (drastically) the classes of hybrid automata and/or properties to be verified.

In the following, we systematically pursue such decidability results. The restrictions we impose have an intuitive, physical sense, and they still allow to verify quite large and natural classes of applications. All restrictions have in common some kind of 'uniformity' that is imposed to the continuously infinite sets of runs of hybrid automata, thus reducing their battling undecidability.

[7] in this sense, symbolic states and runs perform an *abstract interpretation* of hybrid automata

3 Strongly-Non-Zeno hybrid automata and time-bounded properties

Hybrid automata are subject to the so-called *Zeno paradoxes* that occur at the border of continuous and discrete phenomena. We point out two such paradoxes, and then show they are connected with the undecidability problems.

An infinity of discrete steps in finite time. Consider the automaton in figure 4(a), with one location, one edge and one variable x. Then the edge can be crossed at instants $\frac{1}{2}, \frac{3}{4}, \ldots \frac{2^n-1}{2^n} \ldots$ which makes an infinity of discrete steps in one time unit. We prohibit this kind of pathological behaviour.

Blocking [HNSY94]. Consider the automaton in figure 4(b). When entering location **0** with $x = 1$, there is nothing else to do: the edge cannot be crossed rightaway, since the definition of runs imposes that a non-zero time be spent in location **0**, but this is also impossible - because of incoherence between the location's invariant, differential law, and the edge's relation. The run is blocked. We also prohibit such situations.

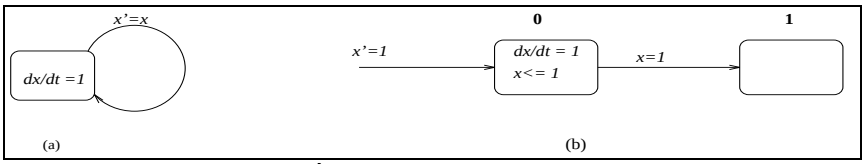

Fig. 4 *Zeno's paradoxes*

Definition: strongly non-Zeno hybrid automata. An *edge-cyclic* run is a run that crosses twice some edge. A *strongly non-Zeno hybrid automaton* respects the following restrictions to avoid Zeno's paradoxes:

- *Uniformly low-bounded:* there exists $\epsilon > 0$ such that all edge-cyclic runs have durations $> \epsilon$;
- *Non-blocking:* all finite runs can be extended to *divergent* runs, by starting with a *continuous* first step.

3.1 Time bounded properties: the logic bounded-TCTL

We present the logic bounded-TCTL, that allows to express *bounded reachability* and *bounded response* properties of hybrid automata. Informally, bounded-TCTL formulas express properties like: starting from an initial set of states ψ, and by remaining in an intermediary set of states ψ_1, it is possible (or inevitable) to reach a final set of states ψ_2, in a time bounded by $c \in \mathbb{Q}$.

Syntax. The bounded-TCTL formulas are $\psi_1 \exists \mathcal{U}_{\prec c} \psi_2$ (called bounded reachability) and $\psi_1 \forall \mathcal{U}_{\prec c} \psi_2$ (called bounded response). The items involved in the formulas are the intermediary and final sets of states ψ_1 and ψ_2, and the *bounded temporal operators* $\exists \mathcal{U}_{\prec c}$ and $\forall \mathcal{U}_{\prec c}$ with $c \in \mathbb{Q}$ and $\prec \in \{<, \leq, =\}$.

Without restricting the generality, we consider that the final set of states is included in the intermediary set : $\psi_2 \subseteq \psi_1$.

Semantics. The meaning of the formulas is:

- *Bounded reachability:* a state (l_0, v_0) satisfies the formula $\psi_1 \exists \mathcal{U}_{\prec c} \psi_2$ if there exists a finite run ρ of duration $d(\rho) \prec c$, originating from (l_0, v_0), such that all its states belong to ψ_1, and its final state belongs to ψ_2;
- *Bounded response:* a state (l_0, v_0) satisfies the formula $\psi_1 \forall \mathcal{U}_{\prec c} \psi_2$ if for any run ρ originating from (l_0, v_0), there exists an initial prefix ρ', of duration $d(\rho') \prec c$, such that all its states belong to ψ_1, and its final state is in ψ_2.

If a state satisfies a formula, we say the formula is *true* at that state. The set of states that satisfy a bounded-TCTL formula ψ is called the *characteristic set* of the formula.

As an example, consider the hybrid automaton in figure 1 and the formula $\mathbf{true} \forall \mathcal{U}_{\leq 48} \mathbf{A_idle}$. This formula expresses the fact that task `A` inevitably terminates within 48 time units. The set of states **true** designs all states of the hybrid automaton, and the set **A_idle** designs all states in the upmost location. The previous formula can be shown to be *false* at all states in location **A_active**.

Symbolic backward analysis. Verifying a bounded-TCTL formula comes to computing its characteristic set. As pointed out in section 2.1, this is impossible to do directly because the state space of a hybrid automaton is a continuously infinite set. However, this problem can be dealt with by *symbolic backward analysis*, which is symmetrical to the symbolic forward analysis presented in section 2.1.

Symbolic backward analysis, when applied to a bounded-TCTL formula, will simulate *backwards* the symbolic runs that arrive in set ψ_2 and that continuously remain is set ψ_1 (we assume that sets ψ_1, ψ_2 are *linear*, so they can be represented by *symbolic states*, cf. section 2.1). By iterating this computation and gathering all the encountered states, one would get all states satisfying the formula. However, an infinity of iterations might be necessary to obtain all these states. It is the same problem (undecidability) pointed out in section 2.1.

3.2 Decidability of bounded-TCTL on strongly non-Zeno hybrid automata

Bounded reachability is decidable. The idea is that, when performing backward analysis for a formula $\psi_1 \exists \mathcal{U}_{\prec c} \psi_2$, the uniformly low-bounded durations of edge-cyclic runs (section 3) make time to 'decrease' uniformly from c to 0, by at least the amount ϵ on each cycle (ϵ is the uniform lower bound for edge-cyclic runs). So, the time bound c is exhausted within a finite number of steps in the symbolic backward analysis, and we obtain all the states that satisfy the formula.

Bounded response is decidable. We now show that bounded response can be translated into bounded reachability. The idea is the following: consider a simple bounded response formula like $\mathbf{true} \forall \mathcal{U}_{\leq c} \psi$. Roughly, its meaning is that each run eventually encounters a state in set ψ within c time units. This comes to saying: it is impossible to remain in set $\neg\psi$ (the complement of ψ) for c time units. So an equivalent formula could be $\neg(\neg\psi \exists \mathcal{U}_{=c} \neg\psi)$.

This is true, in fact, but the full argument is a little more complicated, and it is essentially based on the *non-blocking* quality of strongly non-Zeno hybrid automata (section 3) and on the particular definition that we have given of runs (section 2: continuous steps have strictly positive durations, and any run starts with a continuous step).

For suppose we drop the non-blocking requirement. Then consider the hybrid automaton in figure 4(b), which we have seen precisely to be blocking. The state $(\mathbf{0}, x = 0)$ satisfies formula $\mathbf{true}\forall\mathcal{U}_{\leq 1}\mathbf{location1}$ since on each *divergent* run, it is possible to find a finite initial prefix of duration ≤ 1 that arrives in location $\mathbf{1}$ — note that divergent (i.e. infinite-duration) runs necessarily exit location $\mathbf{0}$. But the same state $(\mathbf{0}, x = 0)$ does not satisfy the supposedly equivalent formula $\neg(\neg\mathbf{location1}\exists\mathcal{U}_{=1}\neg\mathbf{location1})$ — because it is possible to remain in location $\mathbf{0}$ for 1 time unit. So the equivalence does not hold any more when non-blocking is violated.

Furthermore, suppose we admit runs with continuous steps of 0 duration, or runs that start with a discrete step. Using this relaxed definition of runs, the automaton in figure 4(b) becomes non-blocking. By the same argument as above, we show that the equivalence between formulas $\mathbf{true}\forall\mathcal{U}_{\leq 1}\mathbf{location1}$ and $\neg(\neg\mathbf{location1}\exists\mathcal{U}_{=1}\neg\mathbf{location1})$ does not hold.

So the exact definitions are essential. With these definitions, we can prove that in general, bounded response is equivalent to bounded reachability [Rus96]:

$$\psi_1\forall U_{\leq c}\psi_2 \equiv \neg(\neg\psi_2\exists U_{\leq c}\neg\psi_1) \cap \neg(\neg\psi_2\exists U_{=c}\neg\psi_2)$$

This strengthens an existing result about equivalence between bounded response in time $\leq c$ and *unbounded reachability* (in time $> \mathbf{c}$) [HNSY94].

Equivalent conditions for Strongly non-Zeno. The strongly non-Zeno conditions have been shown to guarantee decidability for the time-bounded properties of hybrid automata. It is then interesting to know when a hybrid automaton satisfies these conditions. For instance, the hybrid automaton in figure 1 is strongly non-Zeno: it has *uniform low-bounds for edge-cyclic runs* because for each cycle (in the sense of graphs) there is a stopwatch[8] a that is reset on some edge of the cycle ($a' = 0$) and tested on some edge of cycle by a condition of the form $c \leq a$ with $c \in \mathbb{Q}^+ \setminus \{0\}$; and it is *non-blocking* since all the upper bounds in invariants and edge guards are *strict*. These syntactical restrictions are sufficient to imply strong-non-Zenoness. But we can give *decidable, equivalent* conditions for a hybrid automaton to be strongly non-Zeno.

Checking uniform low-boundedness. For this, we first enrich the automaton with a new variable z, that systematically evolves like global time (continuously by $dz/dt = 1$, discretely by $z' = z$). Then, for each edge of the hybrid automaton perform the following: let l be the origin and l' be the destination of the edge.

[8] variable a is a *stopwatch* if it evolves continuously only by $da/dt \in [0, 1]$ and discretely by $a' = a$ or $a' = 0$

Consider the symbolic state $(l, z = 0 \wedge \mathcal{I}_l)$ — where $\mathcal{I}_l$ is the invariant of l — a polyhedron in $\mathbb{R}^n$, so $z = 0 \wedge \mathcal{I}_l$ makes a polyhedron in $\mathbb{R}^{n+1}$. We compute each symbolic run (see section 2.1) that originates from symbolic state $(l, z = 0 \wedge \mathcal{I}_l)$ until *the chosen edge from l to l' is crossed once more*, and no other edge is crossed twice. This happens on each symbolic run after a finite number of steps.

When this is done, we look at the polyhedra in the last symbolic states. If all these polyhedra have z-components greater than some $\epsilon > 0$, then all edge-cyclic runs that repeat twice our chosen edge (and no other), have uniformly low-bounded durations (by ϵ). Otherwise uniform low-boundedness is violated.

Checking non-blocking. If the previous test has answered *true*, then we proceed with checking the non-blocking condition. We use the following result [HNSY94]: a hybrid automaton is non-blocking iff all its states satisfy the (bounded reachability) formula $\mathbf{true}\exists\mathcal{U}_{=1}\mathbf{true}$. This bounded reachability formula is decidable because the hybrid automaton has just been shown to respect uniform low-boundedness (cf. section 3.2: note there that uniform low-boundedness is enough to guarantee decidability of bounded reachability formulas).

The two tests, for low-boundedness and non-blocking, allow to decide the strongly-non-Zeno quality.

Generalization: uniform divergence. Uniform low-boundedness was proved sufficient for the decidability of bounded reachability formulas. We now show that there is a strictly weaker condition, *uniform divergence*, that still guarantees decidability.

Definition: uniformly divergent hybrid automata. A hybrid automaton is *uniformly divergent* if: for each $c > 0$, there exists a $N(c) \in \mathbb{N}$ *depending only on* c, such that any run of the automaton that crosses N edges has duration $d(\rho) > c$.

So what happens when symbolic backward analysis (section 3.2) is performed on a bounded reachability formula $\psi_1 \exists\mathcal{U}_{\prec c}\psi_2$: the time-bound c decreases on each backwards simulated symbolic run — since it decreases *uniformly* on each run — until it is exhausted, in exactly $N(c)$ steps. Thus, bounded reachability is decidable, and bounded-TCTL is still decidable on the class of *uniformly divergent, non-blocking* hybrid automata.

Obviously, uniformly low-bounded hybrid automata are uniformly divergent. The converse is not true; consider the following *non-linear* hybrid automaton:

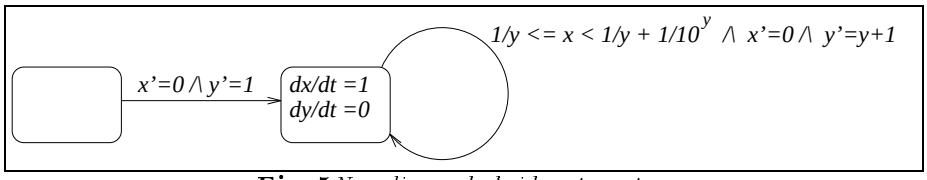

Fig. 5 *Non-linear hybrid automaton*

Variable y takes discrete values: $y = n$ at the n-th crossing of the looping edge. So, the n-th edge-cycle lasts between $\frac{1}{n}$ and $\frac{1}{n} + \frac{1}{10^n}$ time units. Thus,

even if individual edge-cycles can be arbitrary brief, the hybrid automaton is uniformly-divergent.

We can also build a *linear* automaton with the same kind of behaviour: just replace the non-linear guard $\frac{1}{y} \leq x < \frac{1}{y} + \frac{1}{10^y}$ in figure 5, by *another hybrid automaton that performs the division of 1 by y with y decimals and affects the result to x*. This can be accomplished by simulating the Euclid algorithm (hybrid automata can perform all the necessary arithmetic operations with the instantaneous, discrete steps); so this 'Euclid hybrid automaton' can generate values between $\frac{1}{y}$ and $\frac{1}{y} + \frac{1}{10^y}$ in arbitrary small time[9], say $\frac{1}{10^y}$.

The linear hybrid automaton obtained by replacing the looping edge by the Euclid hybrid automaton is uniformly divergent, but not uniformly low-bounded.

4 Uniformly-periodic hybrid automata

We now further restrict the hybrid automata in order to obtain decidability for time-unbounded properties. These are also very important: for instance, the basic *safety* is just unbounded reachability of a bad state. For *unbounded response* (e.g. the system will respond eventually), it can be argued that it is not relevant for real-time applications in which precisely the delays are interesting; but we can use unbounded response to *synthethise* such a delay, by model-checking an unbounded response property with a parameter (variable) modelling the delay.

Definition: uniformly periodic hybrid automata. A hybrid automaton is *uniformly periodic* if there exists $T > 0$ and a distinguished state (l, v) (location+ values of all variables) such that any run of the automaton meets state (l, v) every $\leq T$ time units.

So, all the runs of a uniformly periodic hybrid automaton repeat some given state every at most some time T which is *independent of particular runs*.

Unbounded properties: the logic TCTL**.** To express unbounded properties, the logic bounded-TCTL is extended with the *time-unbounded formulas*. These are, syntactically and semantically, obtained by just dropping the bound c in the corresponding bounded formulas (see section 3.1):

- *Reachability:* a state (l_0, v_0) satisfies the reachability formula $\psi_1 \exists \mathcal{U} \psi_2$ if there exists a finite run originating from (l_0, v_0), such that all its states belong to ψ_1, and its final state belongs to ψ_2;
- *Response:* a state (l_0, v_0) satisfies the formula $\psi_1 \forall \mathcal{U} \psi_2$ if for any run ρ originating from (l_0, v_0), there exists an initial prefix ρ' such that all its states belong to ψ_1, and its final state is in ψ_2.

Other derived properties, for instance *response in at least c time units* denoted $\psi_1 \forall \mathcal{U}_{\geq c} \psi_2$, can be translated into unbounded formulas over modified arguments. For instance, the previous formula can be dealt with by translating it into $\psi_1 \forall \mathcal{U}(\psi_2 \wedge z \geq c)$, where z is a new variable denoting global time.

[9] this shows the extreme computational power of hybrid automata...

Decidability of TCTL on uniformly periodic, strongly non-Zeno hybrid automata. The decidability argument is: since all runs of the (uniformly periodic) hybrid automaton meet some distinguished state every at most T time units, it should be clear that any unbounded property of the automaton is equivalent to a time-bounded property with bound $\leq 2T$; and since time-bounded properties are decidable on the (strongly non-Zeno) hybrid automaton, we have the desired decidability.

Application. Uniformly periodic hybrid automata allow to model systems that are re-initialized to a given state within some time bound. This happens, for instance, in reactive programs (tasks+events) if the tasks are not preempted too often so that they can eventually terminate within *uniformly* bounded time and reset all variables to some fixed value.

Consider for instance the simple reactive program previously modeled in figure 1, which describes the iteration of tasks `A` and `B` disturbed by event `x`. We keep the same durations for tasks `A` (`B`) in interval $[7,8)$, and respectively $[4,5)$; but we change the specification for event `x`, imposing that two consecutive events `x` are separated by at least 6 time units. Then, since the maximum duration of task `B` is 5, one less than the minimum interval between the preempting events `x`, task `A` is not systematically preempted and it eventually terminates, in at most 48 time units. When this happens, all variables are set to fixed values (on edge `end_A`):

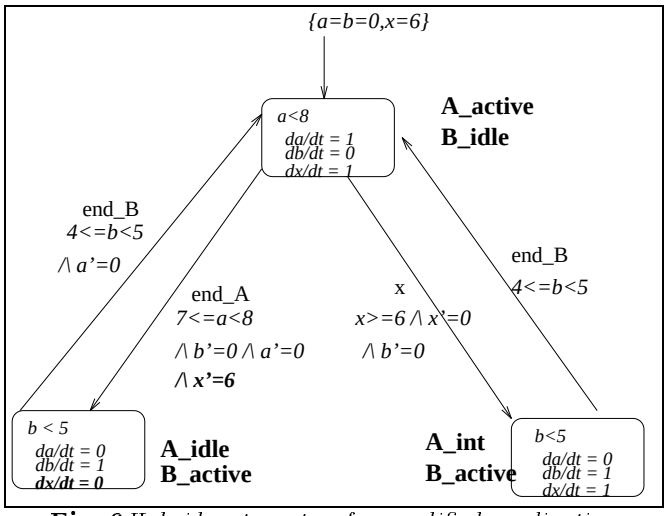

Fig. 6 *Hybrid automaton for modified application*

Note that this resetting slightly changes the timed specification for events `x`: the minimum interval of 6 time units between two consecutive events `x` is *not counted while the concerned task* `A` *is idle*, and the timer `x` is already set to 6 when `A` restarts, so that `x` can occur at once.

We now generalize this idea. Consider a subclass of reactive systems that we shall call *hierarchical*, with the following structure: there is a partial order $\preceq$ on the tasks such that

- tasks which are minimal with regard to $\preceq$ (like `A` in previous example) do not execute in parallel with any others;
- if `B` can be active while `A` is preempted, then necessarily `A` $\preceq$ `B` (this is the case in the previous example)
- there is a minimum interval of time between preemptions such that every task eventually terminates within some uniform time T.

These conditions are met, for instance, in real-time systems with several preemption levels, in which interrupt procedures can be themselves interrupted; but this does not occur too often so that every task manages to terminate. Then it can be argued that such hierarchical systems can be modeled by *strongly non-Zeno, uniformly-periodic* hybrid automata. Indeed, any task eventually terminates within some uniform time T, in particular the ones that are minimal with regard to order $\preceq$. When such a minimal task terminates, all variables can be set to fixed values (task timers to 0 and event timers to their minimal separation value — as x was set to 6 in the previous example) because at that moment, the above conditions guarantee that *there is not other task running or preempted*.

We shall see another example of such hierarchical systems in section 5.

Criterion. We now give a sufficient criterion for a hybrid automaton to be uniformly periodic. The idea is to generalize the automaton of figure 6: we impose that on each cycle of the automaton (in the sense of graphs), there exists a variable a and a location l on the cycle such that:

- the invariant of location l includes an inequality $a < M$ with $M > 0$;
- a has differential law at location l of the form $da/dt = k$ or $da/dt \in [k, K]$ with $k > 0$;
- there is an outcoming edge from l that we call the *resetting edge*, whose relation is of the form $m \leq a < M \wedge \bigwedge_{x \in X} x' = c_x$ where all the $c_x \in \mathbb{Q}$;
- and any edge-cyclic run of the automaton that performs the considered cycle, spends some time $\geq \epsilon$ in location l, for some $\epsilon > 0$.

Then obviously no run can loop on the considered cycle without augmenting the value of a, and so the resetting edge is eventually crossed (in uniformly bounded time): the uniform periodicity is guaranteed.

Note that all the above conditions are checkable: all but the last are syntactical, and the last can be tested by a *forward simulation* of location l's invariant on the cycle (similar to the technique for checking *uniform low-boundedness*, section 3.1).

Generalization: uniform reachability. Uniform periodicity, together with strong non-Zenoness, guarantees the decidability of time-unbounded properties. But uniform periodicity can be relaxed to weaker conditions.

Definition: uniformly reachable hybrid automaton. A hybrid automaton is *uniformly reachable* if: there exists $T > 0$ such that for each pair of states (l_1, v_1) and (l_2, v_2), whenever there is a run from (l_1, v_1) to (l_2, v_2), then there is an initial prefix of the run, of duration $\leq T$, from (l_1, v_1) to (l_2, v_2).

This means that states are not necessarily reachable one from another, but when this is the case, they are reachable in some *uniformly* bounded time. With this definition, it is clear that unbounded properties of hybrid automata reduce to bounded ones, which are decidable under the strongly non-Zeno condition.

A classification of all the obtained decidability results is given in figure 8.

5 Verification example

A real-time application consists of four tasks $\mathtt{A}_{reactive}$, $\mathtt{A}_{back}$, `B` and `C` executing on one processor. Task $\mathtt{A}_{reactive}$ lasts between 7 and 8 time units, and it can be preempted by events `x` and `y`; task $\mathtt{A}_{back}$ lasts between 4 and 5 time units, it does not accept events `x`, `y` but it shares the processor with other time-consuming processes outside our application, such that it gets at any moment at least half of processing time.

An event `x` (resp. `y`) launches task `B` (resp. `C`) which lasts between 1 and 2 (resp. between 0.5 and 1) time units; these two tasks can run in parallel if their launching events occur one after the other; when they complete, task $\mathtt{A}_{reactive}$ will be resumed at preemption point. Two consecutive events `x` (resp. `y`) are separated by at least 2 (resp. 3) time units, and these intervals are not counted when task $\mathtt{A}_{reactive}$ is idle (at the leftmost location); furthermore, these intervals are reset when $\mathtt{A}_{reactive}$ is launched from its beginning, so that `x` and `y` can occur at any moment after that.

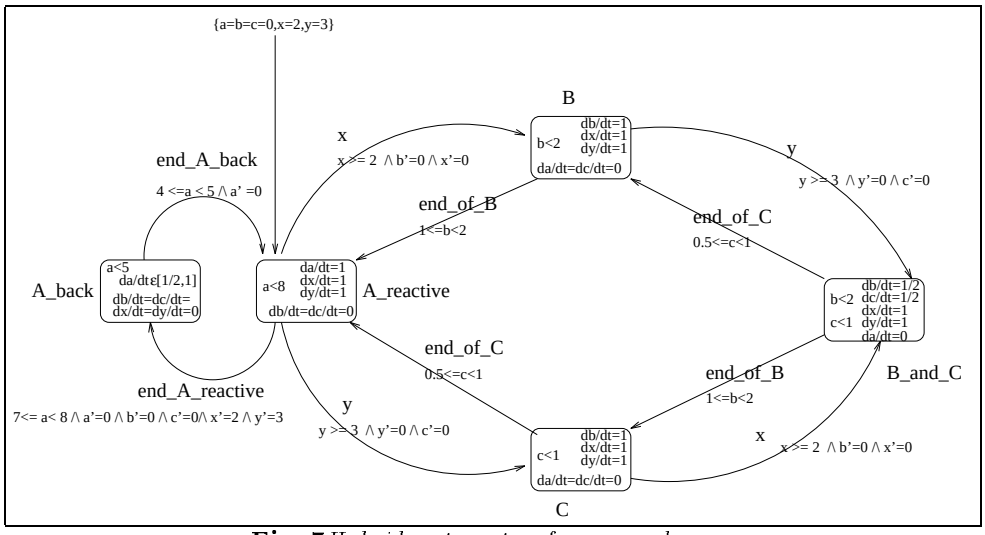

Fig. 7 *Hybrid automaton for example*

This can be modeled by the *strongly non-Zeno* hybrid automaton in figure 7, in which for the sake of simplicity only the active tasks are indicated on locations (like `B_and_C` at the rightmost location, meaning that `B` and `C` are executed in parallel). The differential law $da/dt \in [1/2, 1]$ at location $\mathtt{A}_{back}$ is there to model the variable speed of task $\mathtt{A}_{back}$ when it shares the processor with other tasks (and gets at least half of processing time), and the laws $db/dt = dc/dt = 1/2$ at location `B_and_C` mean that the two tasks are scheduled by equal time sharing.

Temporal properties of the example. We verify the following time-bounded properties with our protype verification tool Valet [Rus96] [10] on a Sun Sparc 10:

1. If only events `y` occur, then task task $\mathtt{A}_{reactive}$ inevitably terminates in at most 8 time units:
   true, checked by Valet in 8 steps and 3 seconds of CPU time.
2. If only events `x` occur, then the task task $\mathtt{A}_{reactive}$ inevitably terminates in at most 250 time units:
   false, checked by Valet in 253 steps and 5 minutes of CPU time.

Both are examples of bounded response properties, and they were also chosen to show the behaviour of task $\mathtt{A}_{reactive}$ with regards to preemptions. In fact it can be seen on the above figure that in general task $\mathtt{A}_{reactive}$ is not guaranteed to terminate ever, because of events `x` that can occur every 2 time units, launching task `B` and preempting $\mathtt{A}_{reactive}$ too often so that it cannot complete.

Indeed, it would be enough to have a greater interval of time (e.g. 4 time units) between two consecutive events `x`, in order to obtain the termination of task $\mathtt{A}_{reactive}$ in a *uniformly bounded* time. This would make the above automaton *uniformly periodic* and since it was already strongly non-Zeno, all TCTL properties become decidable for this application (in fact, we obtain a so-called hierarchical system, described in section 4).

6 Conclusion

In this paper, we have shown that it is possible to get several decidability results from the structural uniformity of runs in hybrid automata. By imposing increasing uniformity, namely *uniform low-boundedness*, *non-blocking* and *uniform periodicity*, we achieve to prove the decidability of increasingly demanding timed properties expressed in the real-time temporal logic TCTL: first the time-bounded properties, and then the time-unbounded ones. Decidable criteria and generalizations are provided for the most important uniformity conditions.

These classes of automata endowed with uniformity came into sight from experiments of real reactive applications programming. A major result is that general TCTL properties are decidable on a class of hybrid automata powerful

[10] other existing verification tools are Kronos [OSY94], Polka [HPR94], HyTech [HHWT94] and UPPAAL [BGK+96]

enough to model real-time applications in which every task eventually terminates in uniformly bounded time.

The obtained results are gathered in figure 8 (we have emphasized the classes of hybrid automata for which there are also decidable criteria).

	TCTL properties	
	bounded	unbounded
Hybrid Automata (H.A.)	semi-decidable	semi-decidable
Strongly-Non-Zeno H.A.(S-N-Z.H.A.) *uniform low-bound on duration of cycles, non-blocking*	decidable	semi-decidable
Uniformly Divergent H.A. *uniform low-bound on duration of N edges, non-blocking*	decidable	semi-decidable
Uniformly Periodic S-N-Z.H.A. *uniform upper-bound of duration between occurences of periodic state*	decidable	decidable
Uniformly Reachable S-N-Z.H.A. *uniform upper-bound of duration between reachable states*	decidable	decidable

Fig. 8 *Summary of main decidability results*

As we have mentionned in the introduction, there are two main directions for improving the verification of reactive and hybrid systems using symbolic model-cheking and hybrid automata. This paper presents some advance in the first direction: restricting hybrid automata in order to obtain decidable subclasses. Now, we are currently starting to work in the second direction: accelerating the verification algorithms. The idea — confirmed until now by experimentation— is that the uniformity we have imposed to hybrid automata, translates into some kind of uniformity in the symbolic states and runs that constitute their abstract interpretation by polyhedra. More precisely, it seems that when a symbolic run completes a cycle, the borders of polyhedra are just *translated* by some fixed amount.

This would mean that any generated symbolic state needs only to visit at most once each location of the hybrid automaton, leading to a substantial improvement of the algorithms implemented in the verification tool Valet.

References

[ACD90] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. In *Proc. IEEE 5th Symp. Logic in Computer Science*, 1990.

[ACH+95] 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. *Theoretical Computer Science B*, 137, January 1995.

[BER94] A. Bouajjani, R. Echahed, and R. Robbana. Verifying invariance properties for timed systems with duration variables. In *Formal Techniques in Real-Time and Fault-Tolerant Systems, LNCS 863*, 1994.

[BES93] A. Bouajjani, R. Echahed, and J. Sifakis. On model checking for real-time properties with durations. In *Proc Symp. on Logic in Computer Science'93, lNCS*, 1993.

[BGK+96] J. Bengtsson, D. Griffioen, K. Kristoferssen, K.Larsen, F.Larsson, P.Petersson, and W.Yi. Verification of an audio protocol with bus collision using UPPAAL. In *Computer-Aided Verification*, 1996.

[Hal93] Nicolas Halbwachs. *Synchronous Programming of Reactive Systems.* Kluwer Academic Publishers, 1993.

[Hen95] T.A. Henzinger. Hybrid automata with finite bisimulations. In *Proc. 22th International Colloquium on Automata, Languages, and Programming, LNCS 944*, 1995.

[HH95] T.A. Henzinger and P.H. Ho. Algorithmic analysis of nonlinear hybrid systems. In *Proc. 7th Conf. Computer-Aided Verification, LNCS 939*, 1995.

[HHWT94] T.A. Henzinger, P.H. Ho, and H. Wong-Toi. A user guide to hytech. In *Proc. 1st. Workshop on Tools and Algorithms for the Analisys of Systems, LNCS*, 1994.

[HKPV95] T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata. In *STOCS'95, LNCS*, 1995.

[HNSY94] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model-checking for real-time systems. *Information and Computation*, (111):193–244, 1994.

[HPR94] N. Halbwachs, Y. E. Proy, and P. Raymond. Verification of linear hybrid systems by means of convex approximations. In *International Symposium on Static Analysis*, 1994.

[KPSY93] Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine. Integration graphs : a class of decidable hybrid systems. In *Proc. Workshop on Theory of Hybrid Systems, LNCS*, 1993.

[OSY94] A. Olivero, J. Sifakis, and S. Yovine. Using abstractions for the verification of linear hybrid systems. In *Proc. Conference on Computer Aided Verification*, 1994.

[PV94] A. Puri and P. Varaiya. Decidability of hybrid systems with rectangular differential inclusions. In *Proc. Conference on Computer-Aided Verification, LNCS*, 1994.

[RCCE92] O. Roux, D. Creusot, F. Cassez, and J.P. Elloy. Le langage réactif asynchrone ELECTRE. *Technique et Science Informatiques (TSI)*, 11(5):35–66, 1992.

[RR94] O. Roux and V. Rusu. Deciding time-bounded properties for ELECTRE reactive programs with stopwatch automata. In *Proc. Workshop on Hybrid Systems and Autonomous Control (Lecture Notes in Computer Science 999)*, pages 405–416, 1994.

[RR95] O. Roux and V. Rusu. Decidable hybrid automata to model and verify real-time systems. In *Second European Workshop on Real-Time and Hybrid Systems*, 1995.

[Rus96] V. Rusu. *Vérification temporelle de programmes* ELECTRE. PhD thesis, Laboratoire d'Automatique de Nantes, Ecole Centrale de Nantes (FRANCE), 1996.