The Impressive Power of Stopwatches[⋆]

Franck Cassez[1] and Kim Larsen[2]

[1] IRCCyN/CNRS UMR 6597, France
Franck.Cassez@irccyn.ec-nantes.fr
[2] Dep. of Computer Science, Aalborg University, Denmark
kgl@cs.auc.dk

Abstract. In this paper we define and study the class of *stopwatch automata* which are timed automata augmented with *stopwatches* and *unobservable behaviour*. In particular, we investigate the expressive power of this class of automata, and show as a main result that any finite or infinite *timed language* accepted by a *linear hybrid automaton* is also acceptable by a stopwatch automaton. The consequences of this result are two-fold: firstly, it shows that the seemingly minor upgrade from timed automata to stopwatch automata immediately yields the full expressive power of linear hybrid automata. Secondly, reachability analysis of linear hybrid automata may effectively be reduced to reachability analysis of stopwatch automata. This, in turn, may be carried out using an easy (over-approximating) extension of the efficient reachability analysis for timed automata to stopwatch automata. We report on preliminary experiments on analyzing translations of linear hybrid automata using a stopwatch-extension of the real-time verification tool UPPAAL.

1 Introduction

Hybrid systems. Hybrid systems [ACH$^+$95,AHH96,Hen96] are a strong extension of timed automata [AD94] used to model systems which combine *discrete* and *continuous* evolutions. The *reachability* and ω*-language emptiness* problems (RP and LEP) are key problems for the verification of hybrid automata: these problems are decidable for *timed automata* (TA) [AD94] (and PSPACE-complete) but not for *linear hybrid automata* (LHA) [ACH$^+$95] for which the reachability problem is only semi-decidable. Decidability of RP and LEP have been extensively studied for subclasses of hybrid automata [KPSY93,AR95,Cer92,JP94,AP94] [HKPV98]. We investigate here a related issue which is the characterization of the expressive power of various subclasses of hybrid automata.

Related work. Concerning expressiveness of timed automata as timed language acceptors, it was proven in [AD94] that Timed Muller Automata (TMA) are as

⋆ This work is partially supported by the European Community Esprit-LTR Project 26270 VHS (Verification of Hybrid systems) and by the FIREworks Esprit WG 23531; it was carried out while the first author was visiting BRICS@Aalborg during the fall 1999.

expressive as Timed Büchi Automata (TBA), which in turn are more expressive than Deterministic TMA (DTMA), which are themselves more expressive than Deterministic TBA (DTBA).

More recent results concern the expressiveness of clocks and the power of (silent) τ-transitions:

– In [HKWT95], Henzinger and al. investigated the power of timing restrictions on finite automata and showed that clock constraints together with time divergence enables one to express Büchi, Muller, Streett, Rabin acceptance and fairness conditions for finite automata; In [ACH94] Alur and al. studied a variety of (un)timed equivalences for timed automata and the distinguishing power of clocks as observers.
– there are also papers devoted to the expressive power of τ-transitions for timed automata; in [BGP96] it is proven that τ-transitions strictly increase the power of timed automata (as timed language acceptors) if they reset clocks; moreover timed automata with τ-transitions are more robust than timed automata without in the sense that any language recognized when the time domain in $\mathbb{N}$ remains recognizable when the domain is $\mathbb{R}$. The expressive power of τ-transitions which reset clocks is settled in [DGP97]; [BDGP98] is a synthesis of the two above mentioned papers.
– other relevant results [HK97,HK96] concern a subclass of hybrid automata: *rectangular hybrid automata* (RHA); in [HKPV98] it is shown that initialized RHA (IRHA) are equivalent to timed automata and thus RP and LEP are decidable for this subclass. The initialization property states that whenever a slope of a variable changes it must be reset. The authors showed that relaxing either the rectangular or the initialization assumptions leads to undecidability of RP and LEP, thus proving that IRHA are at the border or decidability and undecidability.

Our contribution. We compare the expressive power of linear hybrid automata (LHA) and certain of its subclasses. More precisely, we show that, in terms of expressiveness, one important class is the one obtained by a simple addition of stopwatches to the class of timed automata (TA)[1]: we refer to the resulting class as the class of *stopwatch automata* (SWA).

Extending hybrid automata with *unobservable* timed transitions[2], we prove that SWA with unobservable delays are as expressive as LHA. That is, every (ω-) language accepted by a LHA is also accepted by some SWA with unobservable delays. We consider this result interesting for two reasons:

1. it indicates that undecidability of RP and LEP originates from the ability to stop time, and
2. it has a practical application to verification of linear hybrid systems.

The application to verification is based on an easy extension of algorithms for model-checking (safety properties of) TA to (over-approximating) algorithms

[1] here TA refers to timed automata with simple constraints as defined in [AD94].
[2] i.e. some durations are unobservable for this class of hybrid automata.

for model-checking SWA. In particular, this extension may apply the full range of efficient data-structures [LLPY97,ABK$^+$97,BLP$^+$99] currently applied in verification tools for timed automata [Yov97,LPY97]. Analysis of a LHA may now be reduced to a similar analysis on the equivalent SWA, avoiding the need for representing and manipulating general polyhedra.

Outline of the paper. In the next section 2 we give the definitions of LHA and of the subclasses of hybrid automata we will be interested in and we define *unobservable delays*. Section 3 states the main result: the fact that SWA have the same expressive power as LHA. In Section 4 we focus on the practical interest of this equivalence and give examples of LHA we are able to verify using our translation and the SWA-extension of UPPAAL. Finally we conclude in section 5.

2 Linear Hybrid Automata

2.1 Preliminaries

For a given alphabet Σ, Σ^* denotes the set of finite words and Σ^ω the set of infinite words over Σ. Also $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$. We also use the set of booleans $\mathbb{B} = \{\mathsf{tt}, \mathsf{ff}\}$.

A *valuation* is an element of $\mathbb{R}^V$, where V is a finite set of variables. If $|V| = n$, a valuation v can be interpreted as a vector $\overline{v}$ of $\mathbb{R}^n$. If $V' \subseteq V$ and $\nu \in \mathbb{R}^V$, we denote by $proj_{V'}(\nu)$ the valuation $\nu' \in \mathbb{R}^{V'}$ defined by $\nu'(x) = \nu(x), \forall x \in V$.

A *linear expression* $\phi(\overline{v})$ over V is of the form $\sum a_i v_i$ with $a_i \in \mathbb{Z}, v_i \in V$. A *linear constraint* is a propositional formula using the connectives $\vee, \wedge, \neg$ over atomic formulæ of the form $\phi(\overline{x}) \bowtie c$, where $\bowtie \in \{<, =, >\}$, $\phi(\overline{x})$ is a linear expression and $c \in \mathbb{N}$. $\mathcal{LC}(V)$ is the set of linear constraints. If we restrict the linear expressions to one of the simple forms $v - v' \bowtie c$ or $v \bowtie c$, $v, v' \in V$ we obtain the set $\mathcal{SC}(V)$ of *simple constraints* over V. A *linear assignment* over V is of the form $\overline{v} := A.\overline{v} + \overline{b}$ where A is a $n \times n$ matrix with coefficients in $\mathbb{Z}$ and $\overline{b}$ is a vector of $\mathbb{Z}^n$. $\mathcal{LA}(V)$ is the set of linear assignments over V. A *simple assignment* is such that all entries of A are either 0 or 1, and every row of A has at most one non-null coefficient.

Given a valuation v and a constraint γ, the boolean value $\gamma(v)$ describes whether γ is satisfied by v or not.

A *continuous change* of the variables is defined w.r.t. an element d of $\mathbb{Z}^V$ corresponding to the first derivative of each variable: given $t \in \mathbb{R}_{\geq 0}$, the valuation $v + d.t$ is defined by $(v + d.t)(x) = v(x) + d(x).t$.

2.2 Hybrid Automata

Hybrid automata [ACH$^+$95,AHH96,Hen96] are used to model systems which combine *discrete* and *continuous* evolutions. For general hybrid systems the activities can be any continuous functions. However, we restrict our attention to the subclass of *linear* hybrid systems[3]. Moreover we assume that the initial value of each variable is 0 (the corresponding initial valuation is denoted v_0).

Definition 1 (Linear hybrid automaton). *A* linear hybrid automaton *(in the sequel LHA)* $\mathcal{H}$ *is a 7-tuple* $(N, l_0, V, A, E, Act, Inv)$ *where:*

– N *is a finite set of* locations*,*
– $l_0 \in N$ *is the* initial location*,* v_0 *is the* initial valuation*,*
– V *is a finite set of real-valued* variables*,*
– A *is a finite set of* actions*,*
– $E \subseteq N \times \mathcal{LC}(V) \times A \times \mathcal{LA}(V) \times N$ *is a finite set of* edges*;* $e = \langle l, \gamma, a, \alpha, l' \rangle \in E$ *represents an edge from the location* l *to the location* l' *with the linear guard* γ*, the label* a *and the linear assignment* α*.*
– $Act \in \left((\mathbb{Z} \times \mathbb{Z})^V\right)^N$ *where* $Act(l)(x) = [u_1, u_2]$ *means that the first derivative of* x *in location* l *lies in the compact bounded interval* $[u_1, u_2]$ *of* $\mathbb{Z}$*.*
– $Inv \in \mathcal{LC}(V)^N$ *assigns a linear* invariant *to any location.*

To this standard definition we add the following features: the set of locations N *is partitioned into two subsets* $N_o \cup N_u$*;* N_o *(resp.* N_u*) is the set of locations where time-elapsing is* observable *(resp.* unobservable*). We also denote the unobservable action* τ *and consider hybrid automata with* τ *moves. We denote* A^τ *the set* $A \cup \{\tau\}$ *and* Δ^τ *the set* $\mathbb{R}_{\geq 0} \cup \{\tau(t), t \in \mathbb{R}_{\geq 0}\}$ □

Example 1 (Water-level monitor [ACH+95]). As a running example, we consider the water-level monitor described in [ACH+95] and illustrated [4] in Figure 1. The aim is to control the water level in a tank with a monitor. A pump can be turned on and off to control the level. When the pump is off (locations ℓ_2 and ℓ_3) the water level falls by two cms per second; when the pump is on (locations ℓ_0 and ℓ_1), the level rises by one cm per second. The delay to turn the pump on and off is 2 time units (measured by the variable x). Time elapsing is observable in each location of the water level monitor. □

2.3 Semantics

Definition 2. *The semantics of a hybrid automaton* $\mathcal{H} = (N, l_0, V, \Sigma, E, Act, Inv)$ *is a timed transition system* $S_{\mathcal{H}} = (Q, q_0, \Sigma, \rightarrow)$ *where* $Q = N \times \mathbb{R}^V$*,* $q_0 = (l_0, v_0)$ *is the initial state (*$v_0(x) = 0, \forall x \in V$*) and* $\rightarrow$ *is defined by:*

$$\langle l, v\rangle \xrightarrow{\ a\ } \langle l', v'\rangle \text{ iff } \exists\, (l, \gamma, a, \alpha, l') \in E \ \textit{s.t.} \begin{cases} \gamma(v) = \textit{tt},\ v' = \alpha(v) \ \textit{and} \\ Inv(l')(v') = \textit{tt} \end{cases}$$

$$\langle l, v\rangle \xrightarrow{\ e\ } \langle l', v'\rangle \text{ iff } \begin{cases} e = d \ \textit{if}\ l \in N_o, \quad e = \tau(d) \ \textit{if}\ l \in N_u \\ l = l' \ \textit{and}\ \exists d \in Act(l) \ \textit{s.t.}\ v' = v + d.t \ \textit{and} \\ \forall\, 0 \leq d' \leq d, v + d'.t \in Inv(l) \end{cases}$$

$\tau(d)$ *stands for an unobservable delay of duration* d. □

[3] The example of the thermostat in [ACH+95] is a hybrid automata which is *not* linear.

[4] automata designed with GasTeX (http://www.liafa.jussieu.fr/~gastin/gastex).

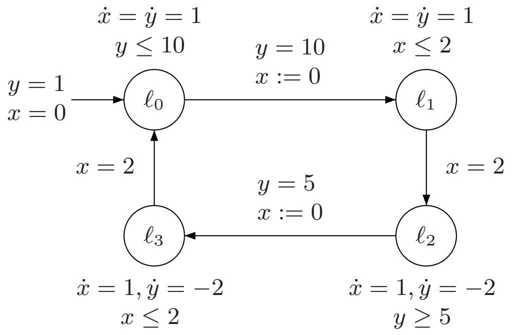


Fig. 1. The water-level monitor

A *run* from a state s_0 of a linear hybrid automaton $\mathcal{H}$ is a sequence of alternating discrete and continuous transitions of $S_{\mathcal{H}}$:

$$\rho = s_0 \xrightarrow{\delta_0} s_0' \xrightarrow{a_0} \dots s_i \xrightarrow{\delta_i} s_i' \xrightarrow{a_i} \dots$$

where $a_i \in A^\tau$ and $\delta_i \in \Delta^\tau$. Intuitively time δ_{i+1} elapses between the actions a_i and a_{i+1}. We also introduce the following derived notations:

$$\begin{aligned}
s &\overset{0}{\Longrightarrow} s' \text{ iff } \exists d \in \mathbb{R}_{\ge 0}\, s.t.\, s \xrightarrow{\tau(d)} s' \\
s &\overset{\varepsilon}{\Longrightarrow} s' \text{ iff } s(\xrightarrow{\tau} \cup \overset{0}{\Longrightarrow})^* s' \\
s &\overset{d}{\Longrightarrow} s' \text{ iff } s \overset{\varepsilon}{\Longrightarrow}\xrightarrow{d_1}\overset{\varepsilon}{\Longrightarrow}\xrightarrow{d_2} \cdots \overset{\varepsilon}{\Longrightarrow}\xrightarrow{d_n} s' \text{ with } d = \textstyle\sum_i d_i \\
s &\overset{a}{\Longrightarrow} s' \text{ iff } s \overset{\varepsilon}{\Longrightarrow}\xrightarrow{a} s'
\end{aligned}$$

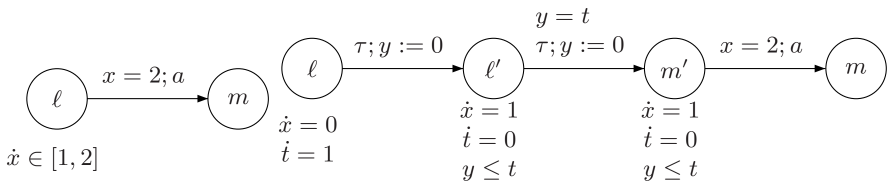
2.4 Timed Languages and Bisimulation for Hybrid Automata

The definition of hybrid automata may be extended with two sets of states: $F \subseteq N$ being a set of *final* states and $R \subseteq N$ being the set of *repeated* states.

The finite (resp. infinite) run ρ is *accepting* if ρ ends in a final state (resp. if there are infinitely many states from R in ρ). With every finite (resp. infinite) run we associate a finite (resp. infinite) τ-*timed word*: with every action a_i we associate a *timestamp* t_i, which is the accumulated observable time since the initial instant. Formally $t_i = \sum_{k=0}^{i}\{\delta_k \in \mathbb{R}_{\ge 0}\}$. Thus the run ρ accepts the τ-timed word $(a_0, t_0)\dots(a_n, t_n)\dots \in (A^\tau \times \mathbb{R}_{\ge 0})^\infty$ and since τ-transitions are unobservable we remove all pairs (τ, t) to obtain a *timed word* $(a_{i_0}, t_{i_0})\dots(a_{i_n}, t_{i_n})\dots \in (A \times \mathbb{R}_{\ge 0})^\infty$

The *timed language* $\mathcal{L}(\mathcal{H})$ accepted by the hybrid automaton $\mathcal{H}$ is the set of timed words which have accepting runs from the initial state of $S_{\mathcal{H}}$.

A symmetric relation $B \subseteq Q \times Q$ is a *weak timed bisimulation* if whenever $(s, t) \in B$, the following holds: (1) $s \in F \Leftrightarrow t \in F$ (resp. $s \in R \Leftrightarrow t \in R$), and (2) whenever $s \overset{d}{\Longrightarrow}\xrightarrow{a} s'$ then $t \overset{d}{\Longrightarrow}\xrightarrow{a} t'$ for some t' with $(s', t') \in B$. We say



(a) Initial LHA: H (b) Corresponding SWA: H'

Fig. 2. Two equivalent linear hybrid automata, H and H'.

that s and t are *weakly timed bisimilar* ($s \approx t$) provided (s, t) is contained in some weak timed bisimulation B. It follows easily that our chosen definition of weak timed bisimilarity between states s and t implies equality in terms of timed words accepted.

Example 2. Consider the linear hybrid automaton H in Figure 2. Assuming that m is final and both ℓ and m are observable, the timed language accepted by H is the set $\{(a, d) \mid 1 \le d \le 2\}$. For the linear hybrid automaton H' a typical run is of the form[5]:

$$\begin{aligned}
\langle \ell, t = 0, x = 0, y = 0\rangle &\xrightarrow{d}\xrightarrow{\tau} \langle \ell', t = d, x = 0, y = 0\rangle \\
&\xrightarrow{\tau(d)}\xrightarrow{\tau} \langle m', t = d, x = d, y = 0\rangle \\
&\xrightarrow{\tau(d')} \langle m', t = d, x = 2, y = d'\rangle \\
&\xrightarrow{a} \langle m, t = d, x = 2, y = d'\rangle
\end{aligned}$$

for $d' \le d$ and $d' + d = 2$. Assuming again that m is accepting, and removing the unobservable part of the above run, yields (a, d) as an accepted timed word. In fact it is not hard to see, that the timed words of the form (a, d), where $1 \le d \le 2$, are precisely the words accepted by H'; hence that $\mathcal{L}(H) = \mathcal{L}(H')$. □

Remark 1. Invariants are not useful when considering timed language acceptance for hybrid automata: we may remove every invariant[6], by translating every transition $\langle l, \gamma, a, \alpha, l'\rangle$ into two consecutive transitions: $\langle l, \gamma \wedge Inv(l), \tau, \alpha.[t := 0], m\rangle$ $\langle m, t = 0 \wedge Inv(l'), a, Id, l'\rangle$, where t is a fresh clock with $\dot{t} = 1$ in m. Leaving the set of final (resp. repeated) states unchanged we obtain the same accepted language. In the sequel we will not consider invariants anymore but assume that this simple translation has already been done for any LHA. □

[5] assuming that ℓ, m are observable and ℓ', m' are unobservable.

[6] this works only for convex invariants as pointed out by an anonymous referee; for a union of convex invariants in ℓ we may add copies of ℓ with convex invariants and apply our translation schema.

2.5 Subclasses of LHA

In the sequel we will consider the following subclasses of LHA (or extended classes of TA)[7]:

TA: the class of timed automata is the one defined in [AD94]; that is, only simple constraints are allowed, the derivatives of the variables (clocks) are all equal to 1 and only linear assignments $(A, \overline{b})$ with $A = 0$ and $\overline{b} \geq 0$ are allowed.

SWA: the class of TA extended with stopwatches; that is the derivative of a variable in a location can be either 0 or 1.

LSWA: the class of LHA, where the derivative of a variable in a location can be either 0 or 1. Alternatively, this class is the extension of SWA which allows linear constraints and assignments.

The set of timed languages accepted by a class C of hybrid automata is denoted $\mathbf{TL}_C$.

Remark 2. For SWA we can allow assignments of the form $x := x' + k$ as they can be written as a shorthand for (1) reset x; (2) let an unobservable delay of $x' + k$ elapse while stopping all the other variables. We will use this shorthand in section 3.1. Notice that allowing this type of assignments in TA make RP and LEP undecidable [BDFP00].

We also point out that stopwatches allow us to use unbounded integers: it suffices to use a variable which is stopped in every location. $\square$

3 From LHA to SWA

In this section we will show that every language accepted by a LHA is also accepted by a SWA with unobservable delays.

Theorem 1 (Expressiveness of SWA). *The classes LHA and SWA are equally expressive in the sense that* $\mathbf{TL}_{SWA} = \mathbf{TL}_{LHA}$. $\square$

The proof of theorem 1 is implied by the following two proof steps: (1) $\mathbf{TL}_{SWA} = \mathbf{TL}_{LSWA}$, and (2) $\mathbf{TL}_{LSWA} = \mathbf{TL}_{LHA}$. According to remark 1 we will only consider LHA with $\mathtt{tt}$ invariant in every location.

3.1 From LSWA to SWA

Let $L = (N, l_0, V, \Sigma, E, Act, Inv)$ be a LSWA (with $\mathtt{tt}$ invariants). We show how to translate the linear guards and linear assignments of L into simple guards and simple assignments thus obtaining a SWA. However, to prepare these translations, we first transform L in order to ensure that no (stopwatch) variable will ever obtain a negative value.

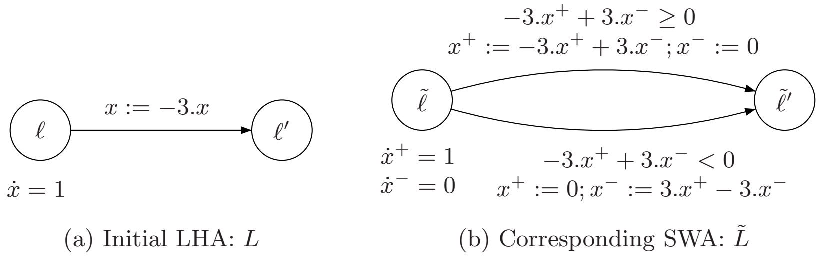


Fig. 3. Translating assignments to avoid negative values

Avoiding negative values. As L is a LSWA, the stopwatch variables are guaranteed never to *decrease* in any location. However, variables may obviously be assigned negative values when transitions are taken (a simple example is given in Figure 3). First, any transition $\langle \ell, a, g, x := \phi(\overline{x}), \ell' \rangle$[8] is split into two transitions $\langle \tilde{\ell}, a, g \wedge \phi(\overline{x}) \geq 0, x := \phi(\overline{x}), \tilde{\ell'} \rangle$ and $\langle \tilde{\ell}, a, g \wedge \phi(\overline{x}) < 0, x := \phi(\overline{x}), \tilde{\ell'} \rangle$. Obviously, this translation does not alter the behavior of L but provides the knowledge of the sign of the value assigned to x. Now for each variable x we introduce two new (stopwatch) variables x^+ and x^- intending that $x = x^+ - x^-$ and $x^+ \geq 0$ and $x^- \geq 0$ will hold invariantly. To ensure this, we do the following:

1. in each location l of L where $\dot{x} = 1$ we set $(\dot{x}^+, \dot{x}^-) = (1, 0)$; if $\dot{x} = 0$ we set $(\dot{x}^+, \dot{x}^-) = (0, 0)$;
2. we add the (obvious) assignments
   - $(x^+, x^-) := (\phi(\overline{x}), 0)$ for transitions where $\phi(\overline{x}) \geq 0$ is in the guard, and
   - $(x^+, x^-) := (0, -\phi(\overline{x}))$ for transitions with guard $\phi(\overline{x}) < 0$.

Finally, we may completely remove the old variable x by replacing it with $x^+ - x^-$ in all terms (in guards or assignments). Figure 3 shows an example of the transformation.

Translating linear guards. Now we focus on transforming linear guards in transitions into simple ones. Thus consider a general transition $\langle l, a, \phi(\overline{x}) \bowtie c, \alpha, l' \rangle$: a is the label, and α the linear assignment; the guard $\phi(\overline{x}) \bowtie c$ is a linear constraint over V with $\bowtie \in \{<, =, >\}$. Notice that it is possible to rewrite this guard in the form $\sum_{i=1}^{n} a_i x_i - \sum_{i=n+1}^{2n} a_i x_i \bowtie c'$ with $c' \in \mathbb{N}$ and $\mathbb{Z} \ni a_i \geq 0$. We demonstrate how to compute these two sums with two new fresh variables u and v. After this it only remains to replace the guard by the simple $u - v \bowtie c'$.

The translation[9] works as follows (see Figure 4[10]):

[7] the notion of unobservable delay is included in all of those classes.
[8] For simplicity we only consider transitions with a single assignment.
[9] $\dot{V} = 0$ means $\forall v \in V, \dot{v} = 0$.
[10] when no label is written on a transition this is a silent τ-transition.

1. we introduce the new stopwatch variables u and v to sum up the terms $a_i x_i, i \in [1 \dots n]$ and $a_i x_i, i \in [n+1 \dots 2n]$ respectively,
2. between the locations $\tilde{l}$ and $\tilde{l}'$ we introduce new locations $\tilde{l}_i$ where time elapsing is unobservable to perform the computations of u and v.

Note that this translation only works because we (by over previous preparation step) may assume that x_i will not have non-negative values: in particular in location $\tilde{l}_i$ we let time x_i elapse which is possible only if $x_i \geq 0$.

Let $\tilde{V} = V \cup \{u, v, t\}$ and $\nu, \nu' \in \mathbb{R}^V$, $\tilde{\nu}, \tilde{\nu}' \in \mathbb{R}^{\tilde{V}}$ s.t. $proj_V(\tilde{\nu}) = \nu$; we then have the following: $\langle \tilde{l}, \tilde{\nu} \rangle \overset{d}{\Longrightarrow} \xrightarrow{a} \langle \tilde{l}', \tilde{\nu}' \rangle$ iff $\langle l, \nu \rangle \overset{d}{\Longrightarrow} \xrightarrow{a} \langle l', \nu' \rangle$ whenever $proj_V(\tilde{\nu}') = \nu'$; thus $\langle \tilde{l}, \tilde{\nu} \rangle$ and $\langle l, \nu \rangle$ are weakly timed bisimilar and consequently accept the same timed words.

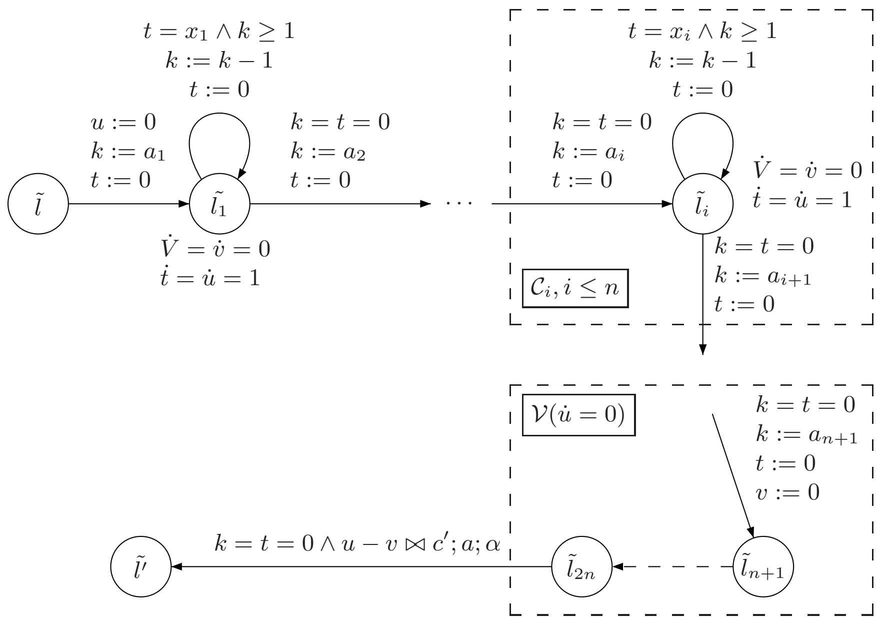


Fig. 4. Computation of u and v before evaluating the guard $u - v \bowtie c'$

Linear assignments. For translating linear assignments we apply the exact same technique as above when dealing with linear guards. Let $A = (a_{ij})$ and $\overline{b} = (b_i)$ with $i, j \in [1..n]$. We introduce n new fresh variables u_i to compute $\sum_{j \leq n} a_{ij} x_j$. At the end we perform the assignments $x_i := u_i + b_i$ (this is possible according to remark 2). Notice we need n new fresh variables as for two simultaneous assignments $x := y$ and $z := x$ we have to keep track of the old value of x needed in $z := x$. For a general transition with a guard and an assignment we perform the computation of the guard described previously and after the last unobservable state we update the variables according to the assignment using new unobservable states. Time elapsing is unobservable in all the intermediate locations. Thus we have the following theorem:

Theorem 2. *The classes LSWA and SWA are equally expressive in the sense that* $\mathbf{TL}_{LSWA} = \mathbf{TL}_{SWA}$. □

Complexity. We do not consider the addition of integer variables. For a LSWA L with n states, m transitions and k stopwatches, the resulting SWA has at most $3k+3$ stopwatches at most $n + 4m(3k+3)$ states.

3.2 From LHA to LSWA

Let $\mathcal{H} = (N, l_0, V, \Sigma, E, Act, Inv)$ be a LHA (with true invariants). In the following we show how to transform $\mathcal{H}$ into a LSWA accepting the same timed language.

Integer slopes. We first deal with integer *positive* slopes: $l \in N$ is a location of $\mathcal{H}$ and x a variable of V s.t. $Act(l)(x) = u_1 \in \mathbb{N}$. We translate location l as follows:

1. we introduce a fresh variable t which is reset when entering $\tilde{l}_1$ and will measure the time the automaton stays in $\tilde{l}_1$,
2. an auxiliary variable v used to update x,
3. 2 locations $\tilde{l}_i$ where time elapsing is unobservable.

The translation is given in Figure 5 within the dashed rectangle (automaton $\mathcal{S}$).

Note that if we enter $\tilde{l}_2$ with the value x^0 for x and t^0 for t we reach $\tilde{l}$ with $x = x^0 + u_1.t^0$. So we can easily prove the following: if $\nu, \nu' \in \mathbb{R}^V$ and $\tilde{\nu}, \tilde{\nu}' \in \mathbb{R}^{V \cup \{t,v\}}$ s.t. $proj_V(\tilde{\nu}) = \nu$ and $d \in \mathbb{R}_{\geq 0}$ then

$$\langle \tilde{l}, \tilde{\nu} \rangle \overset{d}{\Longrightarrow} \langle \tilde{l}', \tilde{\nu}' \rangle \text{ iff } \langle l, \nu \rangle \overset{d}{\Longrightarrow} \langle l', \nu' \rangle \tag{1}$$

Then it suffices to replace location l in $\mathcal{H}$ by the sequence described by the automaton $\mathcal{S}$ on Figure 5 to obtain an automaton which accepts the same timed language as $\mathcal{H}$.

To allow for negative slope, we introduce[11] two new variables x^+ and x^-: for locations where the slope of x is $u \geq 0$, we set $\dot{x}^+ = u$ and $\dot{x}^- = 0$. When the slope of x is $u < 0$ we use the slopes $\dot{x}^+ = 0$ and $\dot{x}^- = u$. Then the actual value of x is $x^+ - x^-$. It remains to replace any occurrence of x in $\mathcal{H}$ by $x^+ - x^-$ to obtain a LHA with only positive slopes.

Interval slopes. We now upgrade the previous construction to deal with slopes belonging to closed integer intervals. We assume $Act(l)(x) \in [u_1, u_2], u_1, u_2 \in \mathbb{Z}_{\geq 0}$ (for the case one of the value is negative we split the interval into a positive part and negative part and apply the translation of x in $x^+ - x^-$.) The construction is given on Figure 5 (ignore the dashed transition).

Statement (1) also holds in this case. Finally we obtain the following theorem:

[11] very similar to the handling of negative values.

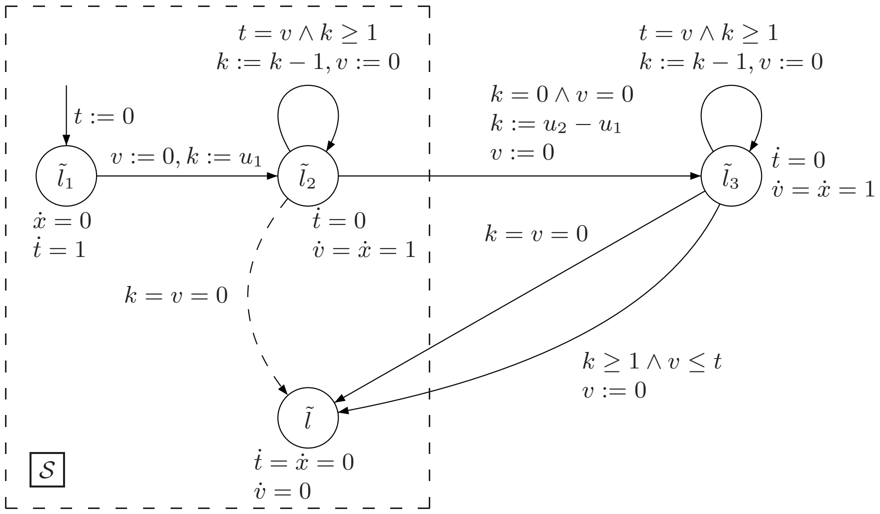


Fig. 5. Translating positive integer slopes

Theorem 3. *The classes LHA and LSWA are equally expressive in the sense that* $\mathbf{TL}_{LHA} = \mathbf{TL}_{LSWA}$. □

This ends the proof of theorem 1.

Complexity. The translation from LHA to LSWA does not require any new variables as we may use those introduced in the translation from LSWA to SWA. We only need to add two states per variable in each location. Then for a LHA with n states, m transitions and k variables the resulting SWA has at most $3k + 3$ variables and at most $2(3k + 3).(n + 4m(3k + 3) + 1)$ states. The translation is then in $\mathcal{O}(k)$ for the number of variables and in $\mathcal{O}(n.m.k^2)$ for the number of states.

4 Application to Symbolic Analysis of Hybrid Automata

Theorem 1 suggests a potential practical application as reachability analysis for a LHA L may effectively be transformed into a reachability analysis on the corresponding SWA $\tilde{L}$. Indeed, it follows from our translation[12] that for any $w \in (\Sigma \cup \mathbb{R}_{\geq 0})^*$

$$\langle l_0, \nu_0 \rangle \overset{w}{\Longrightarrow}^* \langle l, \nu \rangle \text{ iff } \langle \tilde{l}_0, \tilde{\nu}_0 \rangle \overset{w}{\Longrightarrow}^* \langle \tilde{l}, \tilde{\nu} \rangle \qquad (2)$$

whenever $proj_V(\tilde{\nu}_0) = \nu_0$ and $proj_V(\tilde{\nu}) = \nu[V/(V^+ - V^-)]$ where $\nu[V/(V^+ - V^-)]$ stands for ν with all the variables $x \in V$ replaced by $x^+ - x^-$. Thus reachability properties of L and $\tilde{L}$ are in a one-to-one correspondence.

[12] $\Longrightarrow^*$ stands for the transitive closure of $\Longrightarrow$.

4.1 Termination of Symbolic Analysis

Despite the close correspondence in (2), the usual forward reachability algorithm [AHH96][13] may behave differently on $\tilde{L}$ and L with respect to termination. Ideally, we want the algorithm to terminate on $\tilde{L}$ whenever it terminates on L. This is not always the case with the rough translation we have given.

A solution[14] is to ensure that every variable has a single representative (x^+, x^-) in each location.

4.2 Approximate Analysis – Preliminary Experiments

We have successfully analyzed a number of LHA using a stopwatch-extended version of the tool UPPAAL. The extension offers an approximate reachability algorithm for SWA obtained as a rather immediate extension of the existing reachability algorithm for TA. In particular, the existing efficient data-structures of UPPAAL have been reused.

When analyzing SWA using difference bounded matrices (DBMs [LLPY97]), the only operation applied during (symbolic) state-space exploration requiring redefinition is that of computing the *future* of a DBM.

The DBM-based algorithm for SWA yields an over-approximation of the reachable state-space as DBMs only allow to encode the difference between 2 variables. In general, an exact characterization of the reachable state-space of a SWA may require constraints involving several variables. Nevertheless, we have implemented a DBM-based, stopwatch-extension of UPPAAL. Combined with our translation from LHA to SWA we have successfully analyzed a number of examples demonstrating that the prize of over-approximation is not too high.

The examples include the water-level monitor [ACH+95] of Figure 1 for which the SWA is depicted in Figure 6. Another example is the *parameterized* version of Fischer's protocol [ACH+95]. Other successfully investigated examples include the gas burner [ACH+95] and the scheduler of [AHH96]. Also, we are currently investigating the tools applicability to case studies such as the ABR protocol [BF99] and the water-tank case study [KLPW] of the VHS project.

5 Conclusion and Future Work

In this paper we have extended hybrid automata with the notion of unobservable delays. We have proved that stopwatch automata (SWA) with unobservable delays (timed automata extended with stopwatches) are as expressive as linear hybrid automata (LHA) in the sense that every language accepted by a LHA is also accepted by a SWA. A practical application of this result is that reachability analysis for LHA may be reduced to reachability analysis for SWA. We have extended the real-time verification tool UPPAAL to SWA reusing its efficient data-structures, allowing for rather large SWA to be analyzed. Combined with

[13] as used in HyTech

[14] we cannot develop the proof in this paper.

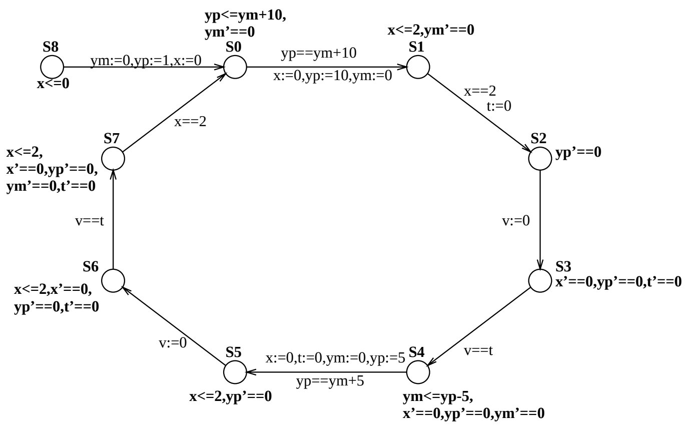


Fig. 6. The SWA water level monitor with UPPAAL

our translation result this offers a completely new method for analyzing LHA. However, the analysis is based on an over-approximation which for some cases may be too coarse to settle a given reachability property.

The translation given in this paper demonstrates that SWA equals LHA in expressive power. However, several alternative translations might have been given and from a practical point of view there are good reasons for looking at such alternative translations as they may be superior in various ways: (1) the translation may well have an impact on the *accuracy* of our tool's approximate analysis on the resulting SWA; (2) from a performance point of view it is important to limit the *complexity* of the translated stopwatch version of a LHA; in particular, translations introducing few additional variables are to be preferred; (3) we want our translation of LHA to SWA to preserve termination when applying non-approximating verification tools such as HyTech.

Our future investigations include: (1) rather than performing an approximate reachability analysis on the SWA (resulting from a translation), it might be possible (and performance-wise superior) to extend the DBM's data-structures to encode exactly the region of a SWA automaton and to perform an exact reachability analysis; (2) as a theoretical aspects we are currently studying the expressive power of unobservable delays for LHA and trying to extend our result to more general classes of hybrid automata (e.g. linear derivatives).

Acknowledgement

The authors would like to thank Gerd Behrmann, Johan Bengtsson and Paul Pettersson for numerous discussions and assistance in making the stopwatch-extension of UPPAAL.

References

ABK$^+$97. E. Asarin, M. Bozga, A. Kerbrat, O. Maler, A. Pnueli, and A. Rasse. Data-Structures for the Verification of Timed Automata. In *Proc. of HART'97, LNCS 1201*, 1997. 140

ACH94. Rajeev Alur, Costas Courcoubetis, and Thomas A. Henzinger. The observational power of clocks. In Bengt Jonsson and Joachim Parrow, editors, *CONCUR '94: Concurrency Theory, 5th International Conference*, volume 836 of *Lecture Notes in Computer Science*, pages 162–177, Uppsala, Sweden, 22–25August 1994. Springer-Verlag. 139

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. 138, 140, 141, 149

AD94. Rajeev Alur and David Dill. A theory of timed automata. *Theoretical Computer Science B*, 126:183–235, 1994. 138, 139, 144

AHH96. Rajeev Alur, Thomas A. Henziger, and Hei-Hsin Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions On Software Engineering*, 22(3):181–201, March 1996. 138, 140, 149

AP94. A. Puri and P. Varaiya. Decidability of hybrid systems with rectangular differential inclusions. In David L. Dill, editor, *Proceedings of the sixth International Conference on Computer-Aided Verification CAV*, volume 818 of *Lecture Notes in Computer Science*, pages 95–104, Standford, California, USA, June 1994. Springer-Verlag. 138

AR95. A. Bouajjani and R. Robbana. Verifying omeg-regular properties for a subclass of linear hybrid systems. In P. Wolper, editor, *Proceedings of the 7th International Conference On Computer Aided Verification*, volume 939 of *Lecture Notes in Computer Science*, pages 437–450, Liege, Belgium, July 1995. Springer Verlag. 138

BDFP00. Patricia Bouyer, Christine Dufourd, Eric Fleury, and Antoine Petit. Decidable updatable timed automata are bisimilar to timed automata. Research report, LSV, ENS Cachan, 2000. Forthcoming. 144

BDGP98. B. Bérard, V. Diekert, P. Gastin, and A. Petit. Characterization of the expressive power of silent transitions in timed automata. *Fundamenta Informaticæ*, 36:145–182, 1998. 139

BF99. B. Bérard and L. Fribourg. Automated verification of a parametric real-time program: the ABR conformance protocol. In *Proc. 11th Int. Conf. Computer Aided Verification (CAV'99), Trento, Italy, July 1999*, volume 1633 of *Lecture Notes in Computer Science*, pages 96–107. Springer, 1999. 149

BGP96. Béatrice Berard, Paul Gastin, and Antoine Petit. On the power of non-observable actions in timed automata. In *13th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1046 of *lncs*, pages 257–268, Grenoble, France, 22–24 February 1996. Springer. 139

BLP$^+$99. Gerd Behrmann, Kim G. Larsen, Justin Pearson, Carsten Weise, and Wang Yi. Efficient timed reachability analysis using clock difference diagrams. In *11th Computer-Aided Verification*, Trento, Italy, July 1999. 140

Cer92. K. Cerans. *Algorithmic problems in analysis in analysis of real-time specifications.* PhD thesis, University of Latvia, 1992. 138

DGP97. Volker Diekert, Paul Gastin, and Antoine Petit. Removing ε-transitions in timed automata. In R. Reischuk, editor, *Proceedings of the 14th Annual Symposium on Theoretical Aspects of Computer Science 1997*, number 1200 in Lecture Notes in Computer Science, pages 583–594, Berlin-Heidelberg-New York, 1997. Springer. 139

Hen96. Thomas A. Henzinger. The theory of hybrid automata. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science*, pages 278–292, New Brunswick, New Jersey, 27–30 July 1996. IEEE Computer Society Press. 138, 140

HK96. Thomas A. Henzinger and Peter W. Kopke. State equivalences for rectangular hybrid automata. In Ugo Montanari and Vladimiro Sassone, editors, *CONCUR '96: Concurrency Theory, 7th International Conference*, volume 1119 of *Lecture Notes in Computer Science*, pages 530–545, Pisa, Italy, 26–29 August 1996. Springer-Verlag. 139

HK97. Thomas A. Henzinger and Peter W. Kopke. Discrete-time control for rectangular hybrid automata. In Pierpaolo Degano, Robert Gorrieri, and Alberto Marchetti-Spaccamela, editors, *Automata, Languages and Programming, 24th International Colloquium*, volume 1256 of *Lecture Notes in Computer Science*, pages 582–593, Bologna, Italy, 7–11 July 1997. Springer-Verlag. 139

HKPV98. Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57(1):94–124, August 1998. 138, 139

HKWT95. Thomas A. Henzinger, Peter W. Kopke, and Howard Wong-Toi. The expressive power of clocks. In Zoltán Fülöp and Ferenc Gécseg, editors, *Automata, Languages and Programming, 22nd International Colloquium*, volume 944 of *Lecture Notes in Computer Science*, pages 417–428, Szeged, Hungary, 10–14 July 1995. Springer-Verlag. 139

JP94. J. McManis and P. Varaiya. Suspension automata: A decidable class of hybrid automata. In David L. Dill, editor, *Proceedings of the sixth International Conference on Computer-Aided Verification CAV*, volume 818 of *Lecture Notes in Computer Science*, pages 105–117, Standford, California, USA, June 1994. Springer-Verlag. 138

KLPW. K. J. Kristoffersen, K. G. Larsen, P. Pettersson, and C. Weise. Verification of an experimental batch plant. VHS internal document. 149

KPSY93. Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine. Integration graphs: A class of decidable hybrid systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 179–208. Springer-Verlag, 1993. 138

LLPY97. K. G. Larsen, F. Larsson, P. Pettersson, and W. Yi. Efficient verification of real-time systems: Compact data structure and state-space reduction. In *Proc. of the 18th IEEE Real-Time Systems Symposium, RTSS'97*. IEEE Computer Society Press, December 1997. 140, 149

LPY97. K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a Nutshell. *Journal of Software Tools for Technology Transfer*, 1(1/2):134–152, October 1997. 140

Yov97. S. Yovine. Kronos: A Verification Tool for real-Time Systems. *Journal of Software Tools for Technology Transfer*, 1(1/2):123–133, October 1997. 140