An Improved Reachability Analysis Method for Strongly Linear Hybrid Systems (Extended Abstract)

Bernard Boigelot*, Louis Bronne* and Stéphane Rassart

Université de Liège
Institut Montefiore, B28
B-4000 Liège Sart-Tilman, Belgium
{boigelot,bronne,rassart}@montefiore.ulg.ac.be

Abstract. This paper addresses the exact computation of the set of reachable states of a strongly linear hybrid system. It proposes an approach that is an extension of classical state-space exploration. This approach uses a new operation, based on a cycle analysis in the control graph of the system, for generating sets of reachable states, as well as a powerful representation system for sets of values. The method broadens the range of hybrid systems for which a finite and exact representation of the set of reachable states can be computed. In particular, the state-space exploration may be performed even if the set of variable values reachable at a given control location cannot be expressed as a finite union of convex regions. The technique is illustrated on a very simple example.

1 Introduction

Hybrid systems are dynamical systems whose variables change both discretely and continuously over time, which makes them well-suited for modeling real-life systems such as embedded controllers and clocked systems. Techniques have been developed for analyzing various properties of hybrid systems [ACHH93, HH94, HNSY94, ACH+95, LPY95, Hen96], and some of them have been implemented as tools such as HyTech [HH95, HHWT95a, HHWT95b], Kronos [DOTY96, DY95, MY96], and UPPAAL [BLL+95]. All the current analysis methods are based on symbolic state-space exploration. Thanks to various search strategies and approximations, a wide range of properties can be decided or semi-decided for some restricted classes of hybrid systems [KPSY92, ACD93, AD94, HKPV95, AHH96].

This paper deals with exact reachability analysis of hybrid systems, i.e., computing an exact and finite representation of their set of reachable states. We restrict our study to strongly linear hybrid systems, which are systems whose discrete variable changes are linear, and whose continuous variable changes obey constant-slope laws. Reachability analysis is traditionally done by performing a search in the state space of the system, while representing sets of reachable states with the help of some symbolic representation system. The usual representation consists of a finite set of

* "Aspirants" (Research Assistants) for the National Fund for Scientific Research (Belgium)

pairs (control location, region), where a region is a convex set of variable values bounded by conjunctive linear constraints. Regions are themselves represented by formulas expressed in some simple arithmetic, or specific mathematical objects such as convex polyhedra. This approach suffers from a major drawback: the exploration algorithm never terminates for systems whose reachable part of the state space cannot be expressed as a finite set of pairs (control location, region). There are however numerous examples of such systems which seem to be analyzable without resorting to approximation techniques, in spite of the fact that the general reachability problem is undecidable for the whole class of strongly linear hybrid systems.

In Section 3, we show how the classical state-space search algorithm can be improved in order to be able to analyze systems with an infinite number of reachable regions. Although our improved algorithm does not always terminate, which is not surprising since it addresses an undecidable problem, it makes it possible to broaden the class of systems for which an exact reachability analysis is possible. In particular, our analysis method is not limited to systems for which every reachable state is reachable by an exploration path of bounded length. Our technique relies on a powerful representation system for sets of values, the *Real Vector Automaton (RVA)*, which is described in Section 4. The technique is illustrated on a very simple example in Section 5.

2 Hybrid Systems

A hybrid system is a dynamical system with discrete and continuous components. It is modeled by a *hybrid automaton*, which consists of a finite-state automaton associated to a set of real variables. The control locations of a hybrid automaton are labeled with *evolution laws* (differential equations) that govern the continuous change of the variables with time, as well as with *invariant conditions* that must hold when the control resides in that location. The transitions of a hybrid automaton are labeled with *guarded assignments*. A transition is *enabled* when the values of the variables satisfy the guard. Following an enabled transition modifies the values of the variables according to the assignment labeling the transition.

In this paper, we restrict ourselves to *strongly linear hybrid systems*, which are hybrid systems with particular restrictions on their evolution laws, invariant conditions, and guarded assignments. A strongly linear hybrid system is composed of the following elements:

- A finite set C of *control locations*.
- A vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbf{R}^n$ of *variables*.
- An *initial location* $c_0 \in C$ and an *initial variable value* $\mathbf{x}_0 \in \mathbf{Z}^n$.
- A labeling function *inv* that assigns to each control location $c \in C$ an *invariant condition* $inv(c)$, which is a predicate over the domain $\mathbf{R}^n$ of the variables. Invariant conditions of strongly linear hybrid systems are of the form $P\mathbf{x} \leq \mathbf{q}$, where $P \in \mathbf{Z}^{m \times n}$ and $\mathbf{q} \in \mathbf{Z}^m (m \geq 0)$.
- A labeling function *eq* that assigns to each control location $c \in C$ a *control law* $eq(c)$, which consists of a differential equation involving the variables. Control laws of strongly linear hybrid systems are of the form $\dot{\mathbf{x}} = \mathbf{d}$, where $\mathbf{d} =$

$(d_1, \ldots, d_n) \in \mathbf{Z}^n$ contains the rates $d_1, \ldots, d_n$ according to which the variables $x_1, \ldots, x_n$ change with time. The vector $\mathbf{d}$ of rates associated to c is denoted *rates*(c).

- A finite set T of *transitions*. Each transition is a triple (c, a, c'), where $c, c' \in C$ are respectively *source* and *target* locations, and a is a *guarded assignment*. Guarded assignments of strongly linear hybrid systems are of the form $P\mathbf{x} \leq \mathbf{q} \rightarrow \mathbf{x} := A\mathbf{x} + \mathbf{b}$, with $P \in \mathbf{Z}^{m \times n}$, $\mathbf{q} \in \mathbf{Z}^m$ $(m \geq 0)$, $A \in \mathbf{Z}^{n \times n}$, and $\mathbf{b} \in \mathbf{Z}^n$. The guard $P\mathbf{x} \leq \mathbf{q}$ and the assignment $\mathbf{x} := A\mathbf{x}+\mathbf{b}$ of a are respectively denoted by *guard*(a) and *assgn*(a).

A *state* of a strongly linear hybrid system is a pair $(c, \mathbf{v})$, where c is a control location and $\mathbf{v}$ is a value for the variables. The state of the system can change in two ways:

- A time delay can modify the value of the variables according to the control law of the current control location, without changing this location. Let $c \in C$ be a control location and $\mathbf{v}, \mathbf{v}' \in \mathbf{R}^n$ be variable values. The state $s' = (c, \mathbf{v}')$ is a *time-step* successor of the state $s = (c, \mathbf{v})$, which we note $s \rightarrow_{\mathrm{ti}} s'$, if there exists $t \in \mathbf{R}^+$ such that $\mathbf{v}' = \mathbf{v} + t.\mathit{rates}(c)$ and both $\mathbf{v}$ and $\mathbf{v}'$ satisfy the invariant condition *inv*(c).
- Following a transition can change the control location and modify the values of the variables according to the guarded assignment of the transition. If $(c, a, c') \in T$ is a transition and $\mathbf{v}, \mathbf{v}' \in \mathbf{R}^n$ are variable values, then the state $s' = (c', \mathbf{v}')$ is a *transition-step* successor of the state $s = (c, \mathbf{v})$, which we note $s \rightarrow_{\mathrm{tr}} s'$, if *guard*$(a)$ is satisfied by $\mathbf{v}$ and *assgn*(a) transforms $\mathbf{v}$ into $\mathbf{v}'$.

Consider two states s and s'. We say that s' is a *step-successor* of s, which we note $s \rightarrow_{\mathrm{s}} s'$, if either we have $s \rightarrow_{\mathrm{ti}} s'$, or there exists $s'' \in C \times \mathbf{R}^n$ such that $s \rightarrow_{\mathrm{ti}} s''$ and $s'' \rightarrow_{\mathrm{tr}} s$. Let $\rightarrow_{\mathrm{s}}^*$ denote the transitive closure of the relation $\rightarrow_{\mathrm{s}}$. The state s' is said to be *reachable* from the state s if we have $s \rightarrow_{\mathrm{s}}^* s'$. A *reachable state* is a state that is reachable from the initial state $(c_0, \mathbf{v}_0)$.

3 Reachability Analysis

3.1 Principles

In this section, we address the problem of computing an exact and finite representation of the set of reachable states of a strongly linear hybrid system. The classical method consists of performing a state-space exploration of the system, starting from a set containing only the initial state and spreading reachability information along control locations and transitions until a stable set is obtained. Reachability information is propagated by executing time-step and transition-step operations from the current set of reachable states. Stabilization is detected by testing if the current set is included in the union of the sets obtained at previous steps. Various search strategies [Eve79] can be used for the exploration (depth-first, breadth-first, ...).

A time-step operation may generate an infinite number of reachable states from a finite number of them. It follows that state-space exploration techniques require

a symbolic representation system for the sets of states that have to be manipulated. Traditionally, sets of states are represented with the help of *regions*, which, for strongly linear hybrid systems, are sets of variable values bounded by conjunctive linear constraints. The idea is to represent a set of states by associating to each control location a finite number of regions corresponding to the variable values that are reachable at that location. Regions are themselves represented by mathematical formulas, or by specific objects such as convex polyhedra. For strongly linear hybrid systems, there exist simple algorithms for computing the effect of time-step and transition-step operations on a set of states represented by a pair (control location, region).

A major drawback of this approach is that state-space exploration will not terminate if there are reachable states that cannot be reached from the initial state by a bounded sequence of time-step and transition-step operations. In particular, this happens when the set of reachable values at some control location cannot be expressed as a finite union of convex regions. An example of such a system is given at Figure 1. Its set of reachable states is given at Figure 2 (solid and dashed lines respectively correspond to values that are reachable at control locations c_0 and c_1).

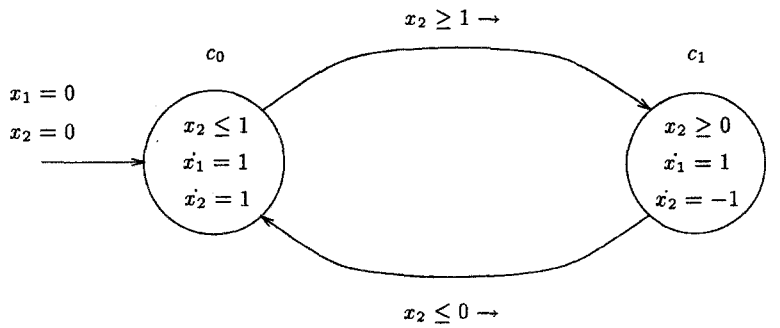

Fig. 1. Example of strongly linear hybrid system.

Let us show how to extend the classical state-space exploration algorithm such as to be able to analyze systems with an infinite number of reachable regions. The main idea, inspired by [BW94] and [BG96], consists of adding a *cycle-step* operation, whose purpose is to capture discrete periodicity. Specifically, given a cycle in the control graph of a hybrid automaton, a cycle-step operation is able to generate all the variable values that could be obtained by performing an unbounded number of times the sequence of time-step and transition-step operations corresponding to the cycle. This makes it possible to generate an infinite number of convex regions in a finite number of steps. As a consequence, cycle-step operations may generate sets of states that are not representable by a finite set of pairs (control location, convex region). It follows that a more powerful representation system is needed. Such a system will be described in Section 4.

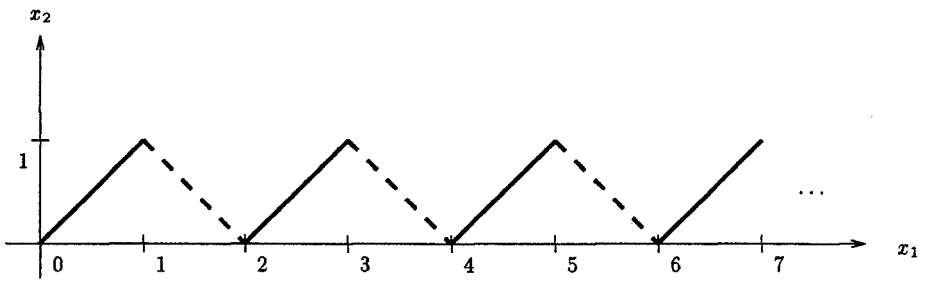

Fig. 2. Reachable states.

3.2 Cycle-step Operations

Let $\mathcal{C} = (c_1, a_1, c_2), (c_2, a_2, c_3), \ldots, (c_{k-1}, a_{k-1}, c_k), (c_k, a_k, c_1)$ be a cycle of transitions. This cycle is said to be *composable* if for all $1 \leq i \leq k$, the conjunction $inv(c_i) \wedge guard(a_i)$ is a predicate of the form $\mathbf{p}_i.\mathbf{x} = q_i$, with $\mathbf{p}_i \in \mathbf{Z}^n$, $q_i \in \mathbf{Z}$, and $\mathbf{p}_i.rates(c_i) \neq 0$. Intuitively, a cycle is composable if for every visited control location and values of the variables at this location, there is exactly one amount of time one can spend at the location that allows to follow the next transition. Determining whether a cycle is composable can be performed by a simple algorithmic check.

Composable cycles have a nice property. If $\mathcal{C}$ is composable, then there exists a guarded assignment $a \equiv P\mathbf{x} \leq \mathbf{q} \rightarrow \mathbf{x} := A\mathbf{x} + \mathbf{b}$, with $P \in \mathbf{Z}^{m \times n}$, $\mathbf{q} \in \mathbf{Z}^m$ $(m \geq 0)$, $A \in \mathbf{Q}^{n \times n}$ and $\mathbf{b} \in \mathbf{Q}^n$, whose effect is equivalent to following $\mathcal{C}$. In other words, executing a from a given set of variable values would yield the same result as performing a time step at the location c_1, followed by a transition step along (c_1, a_1, c_2), then a time step at c_2, and so on until the transition step associated to the last transition (c_k, a_k, c_1) of the cycle. There is a simple algorithm for computing the equivalent guarded command of a composable cycle. It proceeds by first expressing at each control location the only possible time delay at this location as a linear function of the variables, and then by composing the linear transformations undergone by the variables at the control locations and transitions visited by the cycle.

Definition 1 *Let $a \equiv P\mathbf{x} \leq \mathbf{q} \rightarrow \mathbf{x} := A\mathbf{x} + \mathbf{b}$ be the equivalent guarded assignment of some composable cycle $\mathcal{C}$, and $r > 1$ be an integer. The guarded assignment a is said to be* iterable *in base r if there exist $p \in \mathbf{N}_0$ and $m \in \mathbf{N}$ such that the matrix A^p is diagonalizable, and all its eigenvalues belong to $\{0, r^m\}$.*

Iterable guarded assignments have two important properties. First, one can algorithmically check for iterability:

Theorem 2. *There is a decision procedure, based on simple integer arithmetic, for checking whether a guarded assignment is iterable or not.*

Proof The algorithm is left for the full paper. □

Second, one can compute the image of a set of values by the transitive closure of an iterable guarded assignment:

Theorem 3. *There exists a representation system for sets of variable values, such that:*

- *Any finite union of convex regions can be represented.*
- *For every guarded assignment a and represented set V of values, one can compute a representation of the image $V' = a^*(V)$ of V by the transitive closure of a (in other words, V' contains the values obtained by executing repeatedly a any number of times from elements of V).*

Proof A suitable representation system is described in Section 4. The proof that $a^*(V)$ is computable on represented sets is left for the full paper. □

The classical state-space exploration algorithm is extended in the following way. Given a composable cycle $\mathcal{C} = (c_1, a_1, c_2), \ldots, (c_k, a_k, c_1)$ such that its equivalent guarded assignment a is iterable, we simply add to the set of transitions of the system a *meta-transition* (c_1, a^*, c_1), whose effect is to transform a set of values $V \subseteq \mathbf{R}^n$ into $a^*(V)$ without changing the control location c_1. Performing a *cycle-step* operation simply consists of executing a meta-transition. We do not impose an exploration order; however, a breadth-first search will always reach a stable set whenever there is a search order that reaches such a set. Since cycle-step operations generate all the values that could be produced by following repeatedly their underlying cycle, they do not influence the result of a state-space exploration if it terminates. However, they may force the search to terminate, or lower dramatically the number of exploration steps needed before stabilization occurs.

4 Real Vector Automata

In this section, we describe a symbolic representation system well suited to the sets of values that are manipulated by the improved reachability analysis method described in the previous section. The requirements on this system are linked to the operations that are performed during the analysis. Specifically, the representation system has to be able to represent single vectors of integers (such as the set of initial variable values) as well as convex regions. It must be closed over time-step, transition-step, cycle-step, and elementary set-theory operations (union, intersection, ...), and allow an easy computation of their effect on represented sets. Moreover, inclusion of represented sets must be decidable.

4.1 Principles and Definitions

The main idea, inspired by [WB95] and [BG96], consists of representing a set of values by a finite-state automaton accepting encodings of those values as strings of symbols over some alphabet. Since we deal here with sets of vectors with real components, the first step is thus to give an encoding scheme for such vectors.

Let $x \in \mathbf{R}$ be a real number and $r > 1$ be an integer. We encode x in base r, most significant digit first, using r's complement for negative numbers. The result is a word of the form $w = w_i.w_f$, where w_i encodes the integer part of x as a finite word over the alphabet $\{0, \ldots, r-1\}$, the symbol "." is a separator, and w_f encodes

the fractional part of x as a infinite word over the alphabet $\{0,\ldots,r-1\}$. We do not fix the length p of w_i, but only require it to be such that $-r^{p-1}-1 \leq x \leq r^{p-1}+1$. Hence, the most significant digit of a number will be "0" if and only if this number is positive. For simplicity, we require the length of w_f to be infinite (this is not a severe restriction, since an infinite number of "0" symbols can always be appended harmlessly to w_f). The encoding w of x is thus an infinite word over the alphabet $\{0,\ldots,r-1,.\}$. We define its *integer-part length* $|w|_i$ as the number of symbols in w_i. It is noteworthy to remark that for some $x \in \mathbf{R}$ and $p \in \mathbf{N}$, there exist two encodings of x of integer-part length p. For instance, choosing $r = 10$, $x = 11/2$ and $p = 3$ yields the two words $005.5(0)^\omega$ and $005.4(9)^\omega$. Such encodings are said to be *dual*.

To encode a vector of real numbers, we encode each of its components with words of identical integer-part length. This length can be chosen arbitrarily, provided that is sufficient for encoding the vector component with the highest magnitude. It follows that any vector has an infinite number of possible encodings. An encoding of a vector of reals $\mathbf{x} = (x_1,\ldots,x_n)$ can indifferently be viewed either as a tuple $(w_1,\ldots,w_n)$ of words of identical integer-part length over the alphabet $\{0,\ldots,r-1,.\}$, or as a single word w over the alphabet $\{0,\ldots,r-1\}^n \cup \{.\}$.

Since a real vector has several possible encodings, we have to choose which of these the automata we define will recognize. A natural choice is to accept all encodings. This leads to the following definition.

Definition 4 *Let $n \geq 0$ and $r > 1$ be integers. A* Real Vector Automaton (RVA) *$\mathcal{A}$ in base r for vectors in $\mathbf{R}^n$ is a Büchi automaton [Büc62] over the alphabet $\{0,\ldots,r-1\}^n \cup \{.\}$, such that:*

- *Every word w accepted by $\mathcal{A}$ is of the form $w = w_i.w_f$, with $w_i \in (\{0,\ldots,r-1\}^n)^*$ and $w_f \in (\{0,\ldots,r-1\}^n)^\omega$.*
- *For every vector $\mathbf{x} \in \mathbf{R}$, $\mathcal{A}$ accepts either all the encodings of $\mathbf{x}$ in base r, or none of them.*

A RVA is said to *represent* the set of vectors encoded by the words belonging to its accepted language. Remark that the representation is not canonical, for different Büchi automata may accept the same language.

4.2 Elementary RVA

RVA representing sets of real vectors satisfying elementary predicates are easy to obtain. We have the following result.

Theorem 5. *Let $n \geq 0$ and $r > 1$ be integers. There exist RVA for representing in base r the sets:*

- $\mathbf{Z}^n$;
- $\{\mathbf{v}\}$, *for any* $\mathbf{v} \in \mathbf{Q}^n$;
- $\{(x_1,x_2) \in \mathbf{R}^2 \mid x_1 \theta x_2\}$, *for any* $\theta \in \{=,\neq,<,>,\leq,\geq\}$;
- $\{(x_1,x_2,x_3) \in \mathbf{R}^3 \mid x_1 + x_2 = x_3\}$.

Proof The RVA will be given in the full paper. □

4.3 Elementary Operations on RVA

We consider operations on sets of real vectors and study their implementation by operations on the RVA representing those sets. We have the following result.

Theorem 6. *Let $r > 1$ and $m \geq 0$ be integers, V_1, V_2 be sets of real vectors of respective arities (number of components per vector) n_1 and n_2, and A_1, A_2 be RVA representing respectively V_1 and V_2. There exist algorithms for computing RVA representing:*

- *The union $V_1 \cup V_2$ and intersection $V_1 \cap V_2$, provided that $n_1 = n_2$;*
- *The complement $\overline{V_1}$;*
- *The Cartesian product $V_1 \times V_2 = \{(\mathbf{x}_1, \mathbf{x}_2) \mid \mathbf{x}_1 \in V_1 \wedge \mathbf{x}_2 \in V_2\}$;*
- *The projection $\exists x_i V_1 = \{(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{n_1}) \mid (\exists x_i)(x_1, \ldots, x_{n_1}) \in V_1\}$;*
- *The reordering $\pi V_1 = \{(x_{\pi(1)}, \ldots, x_{\pi(n_1)}) \mid (x_1, \ldots, x_{n_1}) \in V_1\}$, where π is a permutation of $\{1, \ldots, n_1\}$;*
- *The expansion $expand(V_1, m) = \{r^{mk}\mathbf{x} \mid \mathbf{x} \in V_1 \wedge k \in \mathbf{N}\}$.*

Moreover, there are algorithms for deciding if V_1 is empty, and if $V_1 \subseteq V_2$.

Proof The algorithms are left for the full paper. □

An important consequence of this result and those of the previous section is that any set of real vectors definable in the structure $\langle \mathbf{R}, +, \leq, Z \rangle$, where Z is the predicate defined as

$$Z(x) \equiv \begin{cases} \text{True if } x \in \mathbf{Z} \\ \text{False if } x \in \mathbf{R} \setminus \mathbf{Z} \end{cases}$$

is representable by a RVA (in any base). Moreover, any set operation expressed in the structure $\langle \mathbf{R}, +, \leq, Z \rangle$ is computable on RVA. Remark that the reciprocal is not true, since the expansion operation is not definable in $\langle \mathbf{R}, +, \leq, Z \rangle$. A similar result appears in [Büc62].

4.4 Performing Step Operations with RVA

Using RVA in the context of our improved reachability analysis method as described in Section 3 requires to be able to perform time-step, transition-step, and cycle-step operations on sets of states represented by a finite union of pairs (control location, RVA). This can be done thanks to the following result:

Theorem 7. *Let $c \in C$ be a control location and $V \subseteq \mathbf{R}^n$ be a set of variable values represented by a RVA $\mathcal{A}$.*

- *One can compute a RVA representing the result*

$$\{\mathbf{v}' \mid (\exists \mathbf{v} \in V, t \in \mathbf{R}^+)(\mathbf{v}' = \mathbf{v} + t.\text{rates}(c) \wedge \text{inv}(c)(\mathbf{v}) \wedge \text{inv}(c)(\mathbf{v}'))\}$$

  of a time-step operation performed at c.
- *Let $t = (c, a, c')$ be a transition. One can compute a RVA representing the result $a(V)$ of a transition-step operation performed along t.*

- *Let $m = (c, a^*, c')$ be a meta-transition. One can compute a RVA representing the result $a^*(V)$ of a cycle-step operation performed along m, provided that a is iterable in the base r of $\mathcal{A}$.*

Proof The idea is to express the results in terms of the elementary sets and operations concerned by Theorems 5 and 6. The complete proof is left for the full paper. □

5 Example of Use

Let us show how the improved reachability analysis method presented in Section 3 can be applied to the very simple system depicted at Figure 1.

The first step consists of adding meta-transitions. The hybrid automaton contains the cycle $\mathcal{C} = (c_0, a_0, c_1), (c_1, a_1, c_0)$, with $a_0 \equiv x_2 \geq 1$ and $a_1 \equiv x_2 \leq 0$. This cycle is composable. Indeed, $(x_2 \leq 1 \wedge x_2 \geq 1) \equiv (x_2 = 1)$ and $(x_2 \geq 0 \wedge x_2 \leq 0) \equiv (x_2 = 0)$ have both the form $\mathbf{p}_i.\mathbf{x} = q_i$, $i = 0, 1$, where each $\mathbf{p}_i$ is such that $\mathbf{p}_i.rates(c_i) \neq 0$. Since $\mathcal{C}$ is composable, there exists a guarded assignment a whose effect is equivalent to following $\mathcal{C}$. In order to compute the components of a, we express at each control location the (unique) time delay that can be spent there as a linear function of the variables. If one spends the time t_0 at location c_0, the evolution law will cause the variable values to undergo the transformation

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} := \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + t_0 \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \tag{1}$$

Since the result of this transformation must satisfy the output condition $x_2 = 1$, we obtain $t_0 = 1 - x_2$. Replacing this value in (1), we obtain that the effect of a time-step operation at c_0 is equivalent to the guarded assignment

$$a_{c_0} \equiv x_2 \leq 1 \rightarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} := \begin{bmatrix} x_1 + 1 - x_2 \\ 1 \end{bmatrix}.$$

Similarly, one obtains for the control location c_1 the guarded assignment

$$a_{c_1} \equiv x_2 \geq 0 \rightarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} := \begin{bmatrix} x_1 + x_2 \\ 0 \end{bmatrix}.$$

Finally, the sequential composition of a_{c_0}, a_0, a_{c_1} and a_1 yields the guarded assignment

$$a \equiv x_2 \leq 1 \rightarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} := \begin{bmatrix} x_1 - x_2 + 2 \\ 0 \end{bmatrix},$$

which therefore captures the effect of $\mathcal{C}$. The last expression can be rewritten in the canonical form $P\mathbf{x} \leq \mathbf{q} \rightarrow \mathbf{x} := A\mathbf{x} + \mathbf{b}$:

$$a \equiv \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq 1 \rightarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} := \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 2 \\ 0 \end{bmatrix}.$$

Since A is diagonalizable and has the eigenvalues 0 and 1, the guarded assignment a is iterable (in any base). We can therefore add the meta-transition (c_0, a^*, c_0) to the system.

We are now ready for exploring the state space of the system. Let us simply show that there exists a single exploration path of finite length that visits all the reachable states (the existence of such a path implies that a breadth-first search would terminate at a depth less or equal to the length of the path). The results are given at Figure 3. For clarity, each computed set of states is prefixed by the operation that produced it, and is expressed as a pair (control location, set of values). In the actual computation, the sets of variable values are represented by RVA.

(initial set of states)	$S_0 = (c_0, \{(0,0)\})$
(cycle step, (c_0, a^*, c_0))	$S_1 = (c_0, \{(2\lambda, 0) \mid \lambda \in \mathbf{N}\})$
(time step, c_0)	$S_2 = (c_0, \{(2\lambda+\delta, \delta) \mid \lambda \in \mathbf{N} \wedge 0 \leq \delta \leq 1\})$
(transition step, (c_0, a_0, c_1))	$S_3 = (c_1, \{(2\lambda+1, 1) \mid \lambda \in \mathbf{N}\})$
(time step, c_1)	$S_4 = (c_1, \{(2\lambda+1+\delta, 1-\delta) \mid \lambda \in \mathbf{N} \wedge 0 \leq \delta \leq 1\})$
(transition step, (c_1, a_1, c_0))	$S_5 = (c_0, \{(2\lambda+2, 0) \mid \lambda \in \mathbf{N}\}) \subseteq S_1$

Fig. 3. State-space exploration path.

6 Conclusions and Comparison with Other Work

We give an algorithm for computing an exact and finite representation of the set of reachable states of a strongly linear hybrid system. Our algorithm can be seen as a strict extension of existing methods [HNSY94, ACH+95, Hen96], which are based on state-space exploration. The improvement consists of a new operation for generating sets of reachable states, that is based on a cycle analysis in the control graph of the system, combined with an original representation system for sets of variable values. Our algorithm considerably broadens the class of systems for which an exact reachability analysis is possible. In particular, it is not limited to systems such that the set of reachable values at each control location can be expressed as a finite union of convex regions. When it terminates, our algorithm allows to decide properties such as reachability of isolated states, or reachability of sets of states expressed as a finite union of pairs (control location, set of values defined in the structure $\langle \mathbf{R}, +, \leq, Z \rangle$).

Of course, since reachability of a given state is undecidable for strongly linear hybrid systems [HKPV95], our algorithm does not necessarily terminate. From a theoretical point of view, this might seem unsatisfactory, but from a practical point of view, this is not at all troublesome. Indeed, our algorithm always terminates whenever existing techniques succeed in producing an exact representation of the reachable part of the state space, and may give out an exact answer when traditional algorithms must resort to approximations methods [HH94]. Moreover, it may produce a faster result for systems having a finite but large number of reachable regions.

Expressing sets of real vectors as finite automata is a very old idea [Büc62], which has originally been introduced as a tool for establishing decidability results in arithmetic. However, the use of finite automata as actual representations of sets of real vectors is original, and generalizes previous results [WB95, BG96] which were obtained for very different systems. The idea of using meta-transitions for speeding up reachability analysis was proposed in [BW94, BG96]. Interesting future work will be to generalize to hybrid systems ongoing work concerning symbolic exploration with meta-transitions (for instance, analyzing a larger class of properties than plain reachability). Another interesting subject will be to study the complexity and practical usefulness of the manipulation algorithms for RVA discussed in Section 4, and evaluating the benefits of the overall method on an actual implementation.

7 Acknowledgments

We wish to thank Pierre Wolper for helpful comments on a preliminary version of this paper.

References

[ACD93] R. Alur, C. Courcoubetis, and D. Dill. Model-checking in dense real-time. *Information and Computation*, 104(1):2–34, May 1993.

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 6 February 1995.

[ACHH93] R. Alur, C. Courcoubetis, T.A. Henzinger, and P.-H. Ho. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems I*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer-Verlag, 1993.

[AD94] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 25 April 1994. Fundamental Study.

[AHH96] R. Alur, T.A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22(3):181–201, 1996.

[BG96] B. Boigelot and P. Godefroid. Symbolic verification of communication protocols with infinite state spaces using QDDs. In *Proc. Computer Aided Verification*, volume 1102 of *Lecture Notes in Computer Science*, pages 1–12, New-Brunswick, NJ, USA, July 1996. Springer-Verlag.

[BLL+95] J. Bengtsson, K. G. Larsen, F. Larsson, P. Pettersson, and W. Yi. UPPAAL - a tool suite for automatic verification of real-time systems. In *Proceedings of the 4th DIMACS Workshop on Verification and Control of Hybrid Systems*, New Brunswick, New Jersey, October 1995.

[Büc62] J. R. Büchi. On a decision method in restricted second order arithmetic. In *Logic, Methodology and Philosophy of Science*, Proceedings of the 1960 International Congress, Stanford, California, 1962. Stanford Univ. Press.

[BW94] B. Boigelot and P. Wolper. Symbolic verification with periodic sets. In *Computer Aided Verification, Proc. 6th Int. Conference*, Stanford, California, June 1994. Lecture Notes in Computer Science, Springer-Verlag.

[DOTY96] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool Kronos. In *Hybrid Systems III, Verification and Control*, volume 1066 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.

[DY95] C. Daws and S. Yovine. Two examples of verification of multirate timed automata with Kronos. In *Proceedings of the 1995 IEEE Real-Time Systems Symposium*, Pisa, Italy, 1995. IEEE Computer Society Press.

[Eve79] S. Even. *Graph Algorithms*. Computer Science Press, 1979.

[Hen96] T. 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.

[HH94] T.A. Henzinger and P.-H. Ho. Model-checking strategies for linear hybrid systems. Technical Report CSD-TR-94-1437, Cornell University, 1994. Presented at the Seventh International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (Austin, TX).

[HH95] T.A. Henzinger and P.-H. Ho. HyTech: The Cornell Hybrid Technology Tool. In P. Antsaklis, A. Nerode, W. Kohn, and S. Sastry, editors, *Hybrid Systems II*, volume 999 of *Lecture Notes in Computer Science*, pages 265–293. Springer-Verlag, 1995.

[HHWT95a] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: the next generation. In *Proceedings of the 16th Annual Real-time Systems Symposium*, pages 56–65. IEEE Computer Society Press, 1995.

[HHWT95b] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. A user guide to HyTech. In E. Brinksma, W.R. Cleaveland, K.G. Larsen, T. Margaria, and B. Steffen, editors, *TACAS 95: Tools and Algorithms for the Construction and Analysis of Systems*, volume 1019 of *Lecture Notes in Computer Science*, pages 41–71. Springer-Verlag, 1995.

[HKPV95] T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *Proceedings of the 27th Annual Symposium on Theory of Computing*, pages 373–382. ACM Press, 1995.

[HNSY94] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994. Special issue for LICS 92.

[KPSY92] Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine. Integration graphs: a class of decidable hybrid systems. In *Proceedings of Workshop on Theory of Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 179–208, Lyngby, Denmark, 1992. Springer-Verlag.

[LPY95] K. G. Larsen, P. Pettersson, and W. Yi. Model-checking for real-time systems. In Horst Reichel, editor, *Proceedings of the 10th International Conference on Fundamentals of Computation Theory*, volume 965 of *Lecture Notes in Computer Science*, pages 62–88, Dresden, Germany, August 1995. Springer-Verlag.

[MY96] O. Maler and S. Yovine. Hardware timing verification using Kronos. In *Proceedings of the IEEE 7th Israeli Conference on Computer Systems and Software Engineering, ICCBSSE'96*. IEEE Computer Society Press, 1996.

[WB95] P. Wolper and B. Boigelot. An automata-theoretic approach to Presburger arithmetic constraints. In *Proc. Static Analysis Symposium*, Lecture Notes in Computer Science, Glasgow, September 1995. Springer-Verlag.