Verification of Hybrid Systems with Linear Differential Inclusions Using Ellipsoidal Approximations [⋆]

Oleg Botchkarev[⋆⋆] and Stavros Tripakis

The University of California at Berkeley
195M Cory Hall Berkeley CA 94720
Phone: (510) 642-5649, Fax: (510) 642-6330
{olegb,stavros}@eecs.berkeley.edu

Abstract. A general verification algorithm is described. It is then shown how ellipsoidal methods developed by A. B. Kurzhanski and P. Varaiya can be adapted to the algorithm. New numerical algorithms that compute approximations of unions of ellipsoids and intersections of ellipsoids and polyhedra were developed. The presented techniques were implemented in the verification tool called VeriSHIFT and some practical results are discussed.

Keywords: hybrid systems, verification, reachability analysis, ellipsoidal approximations.

1 Introduction

A number of application domains, such as car manufacturing, robotics, chemical process control, or avionics, involve *controllers*, consisting of: (a) a set of *sensors* and *actuators*, representing the interface between the controller and its environment; (b) a *control logic* (implemented as one or more circuits or as one or more pieces of software running concurrently), which represents the way the controller should act on the environment.

A promising model for describing such systems is *hybrid automata* [8]. Hybrid automata are finite-state machines equipped with continuous variables. Each discrete state of an automaton has a system of differential equations that govern its continuous variables. Most correctness criteria for such systems can be stated as a *safety* property: the system must never reach an “unsafe” (or a “bad”) state.

Ensuring correctness of the model is often not a trivial task. Simulation of the system is not adequate, since it can only help examine a limited number of trajectories. Analytical methods are often not applicable, considering the complex interaction of continuous and discrete dynamics. An alternative is *reachability analysis*. It consists of computing the set of all reachable states of the system and

[⋆] Research supported by National Science Foundation Grant ECS 9725148 and ONR Contract 11

[⋆⋆] Corresponding author.

then checking that no "bad" state belongs to the reachable set. The reachability problem has been shown to be undecidable, even for models of hybrid automata with simple dynamics (e.g., $\dot{x} \in [a, b]$). Moreover, the so-called *state explosion problem* (the machine representation of the set of reachable states is too large) often limits the applicability of the method, even for decidable sub-classes of the model.

Approximations have been used as a remedy to both the undecidability and the state-explosion problems. Computing an over- or under-approximation (i.e., external or internal approximation) of the exact set of reachable states can be, first, decidable, and second, less expensive, in terms of time and memory. The price to pay is accuracy: what does it mean for a "bad" state to be reachable in the approximative analysis?

This paper presents a new reachability technique for systems of hybrid automata with linear dynamics, expressed as *differential inclusions*: $\dot{x} \in Ax + U$. The basic model of hybrid automaton and its semantics are presented in section 2.

The algorithm performs reachability analysis for bounded time. Reachability for bounded time means that the set of states reachable in Δ time units is computed, where Δ is a parameter supplied by the user. The skeleton of the algorithm, correctness, and trade-offs between accuracy and efficiency are discussed in section 3. A generalization of the algorithm is presented in [1].

The algorithm is based on the ability to approximate: (a) the reachable set of a linear differential inclusion (time propagation); (b) intersections of convex sets; (c) unions of convex sets; (d) linear transformations and geometric sums of convex sets.

Among methods of reachability analysis are those based on ellipsoidal techniques. The presented work is an attempt to use some of the methods described in [5,6].

New methods for computing over-approximations of unions of ellipsoids and intersections of ellipsoids and polyhedrons have also been devised. They are presented in section 4.

These reachability techniques have been implemented in a prototype tool called *VeriSHIFT* (section 5). The tool accepts systems of hybrid automata, communicating by input/output variables and synchronous message passing. Dynamic creation and reconfiguration of automata is also supported.

2 The Model

In this section we present the model of a single hybrid automaton. We consider the extension to systems of communicating hybrid automata in [1].

Preliminaries Let R be the set of real numbers, $\mathsf{R}^{m \times n}$ the set of $m \times n$ real matrices, $\mathcal{C}^n$ the set of convex closed subsets of R^n, and ${\mathcal{C}_b}^n$ the set of convex compact subsets of R^n. $\mathsf{B}^n_\epsilon(x)$, the *ball of dimension n with center x and radius ϵ*, is defined to be the convex set $\{y \in \mathsf{R}^n \mid |x - y| \leq \epsilon\}$.

Given a set $P \in \mathcal{C}^n$ and a matrix $A \in \mathsf{R}^{m \times n}$, AP is the *linear transformation* of P, that is, a set from $\mathcal{C}^m$ defined as $AP = \{Ax \mid x \in P\}$.

Given two sets P_1, $P_2 \in \mathcal{C}^n$, let $P_1 + P_2$ denote the *geometric (Minkowski) sum* of P_1, P_2, defined as:

$$P_1 + P_2 = \{x \mid \exists x_1 \in P_1, x_2 \in P_2, x = x_1 + x_2\}.$$

A *flow* in R^n is defined as a triple $F : (A, I, U)$, where $A \in \mathsf{R}^{n,n}$ and $I \in \mathcal{C}$, $U \in {\mathcal{C}_b}^n$. F defines the following system of differential equations:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + u(t) \\ x(t) &\in I,\ u(t) \in U. \end{aligned}$$

Given points $x_0, x_1 \in \mathsf{R}^n$, we say that x_1 *is F-reachable from x_0 at time t*, denoted $x_0 \overset{t}{\rightsquigarrow}_F x_1$, if there exist functions of time $x(\cdot)$ and $u(\cdot)$ such that $x(0) = x_0$, $x(t) = x_1$, and for all $\tau \in [0,\, t]$, $x(\tau) \in I$, $u(\tau) \in U$ and $\dot{x}(\tau) = Ax(\tau) + u(\tau)$.

Given a set $X_0 \in {\mathcal{C}_b}^n$, the *reachable set of flow F from X_0 at time t*, denoted $\mathcal{X}_F(X_0,\, t)$, is the set of all points that are F-reachable from points of X_0 at time t.

A Hybrid Automaton. We define a hybrid automaton $\mathcal{A}$ with *linear differential inclusions* to be a tuple $(Q,\, X,\, F,\, T,\, G,\, R)$, where:

- Q is a finite set of *discrete states* (or *locations*, or *modes*).
- X is a set of n *continuous variables* taking values in R.
- $F : Q \to \mathsf{R}^{n \times n} \times \mathcal{C}^n \times {\mathcal{C}_b}^n$ associates with each discrete state q a flow (A, I, U). I is called the *invariant* of q and will be denoted as $I(q)$.
- $T \subseteq Q \times Q$ is a set of *discrete transitions*.
- $G : T \to \mathcal{C}^n$ associates with each discrete transition a *guard*.
- $R : T \to \mathsf{R}^{n \times n} \times {\mathcal{C}_b}^n$ associates with each transition a pair (B, P). This pair defines the *reset* of the continuous variables[1]: $x := Bx + P$.

Given a discrete state q, let $\mathsf{out}(q)$ be its set of *out-going transitions*, $\{(q, q') \in T\}$.

We now turn to the semantics of a hybrid automaton like $\mathcal{A}$. A *state* of $\mathcal{A}$ is a pair $(q, x) \in Q \times \mathsf{R}^n$ such that $x \in I(q)$.

Given a state (q, x) and a *delay* $\delta \in \mathsf{R}$, we say that there is a *time transition* from $(q,\, x)$ to a state $(q,\, y)$, denoted $(q,\, x) \overset{\delta}{\rightsquigarrow} (q,\, y)$, if $x \overset{\delta}{\rightsquigarrow}_{F(q)} y$.

Given a state (q, x) and a discrete transition $a = (q,\, q') \in T$, such that $R(a) = (B, P)$, we say that there is a *discrete jump* from $(q,\, x)$ to a state $(q',\, y)$, denoted $(q,\, x) \xrightarrow{a} (q',\, y)$, if $x \in G(a)$, $y \in I(q')$ and $y \in Bx + P$.

[1] If P contains a single point, $P = \{y\}$ the reset is deterministic, that is, each x is mapped to a unique $x' = Bx + y$. If P is not a singleton, then the reset is non-deterministic.

Reachability. Given a set of initial states S_0, we say that a state s is *reachable* from S_0 if there exists $s_0 \in S_0$ and a sequence

$$s_0 \overset{\delta_1}{\rightsquigarrow} s_1' \xrightarrow{a_1} s_1 \overset{\delta_2}{\rightsquigarrow} s_2' \xrightarrow{a_2} \cdots \overset{\delta_k}{\rightsquigarrow} s_k' \tag{1}$$

such that $s_k' = s$. The sequence (1) and the corresponding trajectory of continuous variables $x(\cdot)$ is called an *execution* of the hybrid automaton, and k is called the *length* of the execution. We say that s is reachable from S_0 *in time Δ* if

$$\delta_1 + \delta_2 + \cdots + \delta_k \leq \Delta.$$

Given a discrete state q, we say that q is reachable from S_0 (in time Δ) if there exists a state (q, x) which is reachable from S_0 (in time Δ).

3 Reachability Using Convex Approximations

Given an automaton A and a set S_0 of initial states, we want to verify whether a discrete state q_{bad} is *not* reachable from S_0 in time Δ.

In this section we describe the skeleton of the reachability algorithm. The algorithm is based on the ability to: (a) effectively represent convex compact sets $X \in {\mathcal{C}_b}^n$; (b) compute an over-approximation $\mathcal{X}_F^+(X_0, t) \supseteq \mathcal{X}_F(X_0, t)$ of the reachable set of a linear flow F from a convex compact set X_0 at time t; (c) check whether the intersection of two convex sets is non-empty; (d) compute over-approximations of intersections, unions and geometric sums[2] of convex sets; (e) compute linear transformations of convex sets. Section 4 deals with points (a) – (d) in detail. In this section, we assume that an effective representation of convex sets and the above operations are available. First we present the basic structure of the reachability algorithm. Then we discuss alternatives and their impact on accuracy and efficiency.

3.1 The Basic Algorithm

The algorithm maintains a table $\mathcal{T}$ of tuples of the form: $(q,\ X,\ \tau)$, where $q \in Q$, $X \in {\mathcal{C}_b}^n$ and $\tau \in [0, \Delta]$. $(q,\ X,\ \tau)$ is supposed to represent a set of *unexplored* states $(q, x), x \in X$. A state $s = (q, x)$ is unexplored in the sense that q_{bad} might be reachable from s in time $\Delta - \tau$. An invariant of the algorithm is that if a state (q, y) is reachable in time Δ then at some point the table will contain a tuple $(q,\ X,\ \tau)$, where $\tau \leq \Delta$ and, either $y \in X$, or there exist $x \in X$ and $t \in \mathsf{R}$ such that $x \overset{t}{\rightsquigarrow}_{F(q)} y$.

$\mathcal{T}$ is initialized to S_0 (the set of initial states), such that for all $(q, X, \tau) \in \mathcal{T}$, $\tau = 0$. The algorithm essentially repeats three steps. First, it chooses an unexplored tuple (q, X, τ). Second, it propagates X in time, until time reaches

[2] The geometric sum $P_1 + P_2$ can be computed exactly if at least one of P_1, P_2 is a singleton. Consequently, the reset of a set X, $BX + P$, can be computed exactly if the reset is deterministic (i.e., P is a singleton).

Δ; meanwhile, it computes the intersection of the reachable tube from X with each of the out-going guards of q. Third, for each intersection V with a guard, the algorithm computes the reset of V with respect to the corresponding discrete transition, and adds a new (unexplored) tuple to the table.

The second step involves computing the reachable set of $F(q)$ from X at time t, for $t \in [0, \Delta - \tau]$. Since it is not possible to compute this set for infinitely many time values, we have to discretize time, that is, we compute $\mathcal{X}_F^+(X_0,\, t)$ for $t = k\delta$, where $k = 0, ..., \lceil \frac{\Delta - \tau}{\delta} \rceil$. The *time step* δ is a parameter of the algorithm, given by the user. In order not to "miss" a guard during the propagation of X in discrete time steps, we "enlarge" the reachable set at each time step by a ball of radius ϵ (see step 2, below). ϵ can be effectively computed as a function of $F(q)$ and δ, so that correctness of the over-approximation is ensured:

Lemma 1. *The following estimate is true for all $t \in [0,\, \delta]$:*

$$\mathcal{X}_F(X_0,\, t) \subseteq X_0 + B_\epsilon(0) \tag{2}$$

where $\mathcal{X}_F(X_0,\, t)$ denotes the reachable set of differential inclusion

$$F : \dot{x} \in Ax + U,\ U \in \mathcal{C}_b{}^n,$$

$B_\epsilon(0) \in \mathcal{C}_b{}^n$ is a ball of radius ϵ with the center in 0, and

$$\epsilon = (e^{N_A \delta} - 1)D + e^{N_A \delta} N_U \delta,$$

$$N_A = \|A\| = \max_{\|x\|=1} \|Ax\|,$$

$$D = \max_{x \in X_0} \|x\|,\ N_U = \max_{u \in U} \|u\|.$$

Proof To ensure that inclusion (2) holds it is enough to take ϵ equal to the Hausdorff semidistance between X_0 and $\mathcal{X}_F(X_0,\, t)$:

$$\epsilon = h_+(\mathcal{X}_F(X_0,\, t),\, X_0) = \max_{y \in \mathcal{X}_F(X_0,\, t)} \min_{x \in X_0} \|x - y\|.$$

Then it is not difficult to see that $h_+(\mathcal{X}_F(X_0,\, t),\, X_0 + B_\epsilon(0)) = 0$ which implies $\mathcal{X}_F(X_0,\, t) \subseteq X_0 + B_\epsilon(0)$.

Let us estimate this Hausdorff distance:

$$\begin{aligned} h_+(\mathcal{X}_F(X_0,\, t),\, X_0) &= \max_{y \in \mathcal{X}_F(X_0,\, t)} \min_{x \in X_0} \|y - x\| \\ &= \max_{y \in X_0} \max_{u(\cdot) \in U} \min_{x \in X_0} \|x_F(y,\, t,\, u(\cdot)) - x\| \\ &\le \max_{y \in X_0} \max_{u(\cdot) \in U} \|x_F(y,\, t,\, u(\cdot)) - y\| \\ &= \max_{y \in X_0} \max_{u(\cdot) \in U} \|e^{At} y - y + \int_0^t e^{A(t-s)} u(s) ds\| \\ &\le \max_{y \in X_0} \|(e^{At} - I)y\| + \max_{u(\cdot) \in U} \|\int_0^t e^{A(t-s)} u(s) ds\| \end{aligned}$$

$$\leq \max_{y \in X_0} \|(e^{At} - I)y\| + \int_0^t \max_{u \in U} \|e^{A(t-s)}u\| ds$$
$$\leq (e^{N_A \Delta} - 1)D + e^{N_A \Delta} N_U \Delta.$$

■

Now we are ready to detail the steps of the algorithm:

1. If the table $\mathcal{T}$ is empty, stop and announce that q_{bad} is unreachable in time Δ. Otherwise, if there exists a tuple $(q_{bad}, \_, \_) \in \mathcal{T}$, stop and announce that q_{bad} is *possibly* reachable in time Δ. Otherwise, choose a tuple $(q, X, \tau) \in \mathcal{T}$ with minimal τ, remove it from the table and proceed to step 2.
2. Let (q, X, τ) be the tuple chosen in step 1, $F = F(q)$ and $\mathsf{out}(q) = \{a_1, ..., a_l\}$. Also, for $i = 1, ..., l$, let $a_i = (q, q_i)$, $R(a_i) = (B_i, P_i)$ and $G_i = G(a_i)$.
   (a) Compute $\mathcal{X}_F^+(X_0, k\delta) + \mathsf{B}_\epsilon^n(0)$, for $k = 0, ..., m$, where m is the minimum between $\lceil \frac{\Delta - \tau}{\delta} \rceil$ and the smallest $k \leq \lceil \frac{\Delta - \tau}{\delta} \rceil$ such that $\left(\mathcal{X}_F^+(X_0, k\delta) + \mathsf{B}_\epsilon^n(0)\right) \cap I(q) = \emptyset$.
   (b) For each $i = 1, ..., l$, let k_i be the first time the reachable set intersects the guard G_i, that is, $k_i = \min\{k \mid (0 \leq k \leq m) \wedge ((\mathcal{X}_F^+(X_0, k\delta) + \mathsf{B}_\epsilon^n(0)) \cap G_i \neq \emptyset)\}$. If the reachable set never intersects G_i, we set $k_i = m + 1$. Let $\tau_i = k_i \delta$. If $B_i \neq 0$ (i.e., the reset is not constant), then we compute:
   $$V_i \supseteq \bigcup_{j=k_i}^{m} \left((\mathcal{X}_F^+(X_0, k\delta) + \mathsf{B}_\epsilon^n(0)) \cap G_i\right). \tag{3}$$
   That is, V_i is (an over-approximation of) the union of intersections of the reachable set and the guard at times $t \geq \tau_i$. If $B_i = 0$ the computation of V_i is unnecessary.
3. For each $\tau_i \leq \Delta$, computed at step 2, we add a tuple $(q_i, X_i', \tau + \tau_i)$ to the table $\mathcal{T}$, where,
   $$\begin{array}{l} X_i' = P_i, \text{ if } B_i = 0, \\ X_i' \supseteq B_i V_i + P_i, \text{ otherwise.} \end{array} \tag{4}$$
   Go back to step 1.

We should point out that in step 2(b), we do not need to compute (an over-approximation of) the intersection of the reachable set and the guard at each time step to check whether it is non-empty. Instead, we can have a procedure that *checks*, given two convex sets, whether their intersection is non-empty. If it is, then we can compute it.

Correctness and Termination We now state the main properties of the algorithm.

Lemma 2. *Let $X_0 \in \mathcal{C}^n$, $G \in \mathcal{C}^n$, $F : \dot{x}(t) \in Ax + U$ - some linear flow and $\delta > 0$. If ϵ is choosen accordingly to lemma (1), then*

$$\bigcup_{\tau=0}^{\Delta} (X_F^+(X_0, \tau) \cap G) \subset \bigcup_{i=0}^{k} ((X_F^+(X_0, i\delta) + B_\epsilon(0)) \cap G), \tag{5}$$

where

$$k \in \mathsf{Z},\ (k-1)\delta < \Delta \leq k\delta.$$

The **proof** is a consequence of lemma (1).

Theorem 1. *If the algorithm terminates doing reachability analysis of a hybrid automaton A for time horizon Δ and reports that the state q_{bad} is unreachable (step 1), and a state $(q^*,\, x^*)$ is reachable at time $t \leq \Delta$ as result of a sequence of time and discrete transitions ending with a discrete transition, then at some step of execution the table $\mathcal{T}$ contained a tuple $(q^*,\, X,\, \tau)$ such that $x^* \in X$ and $\tau \leq t$.*

Proof Let us suppose that

$$s_0 \underset{F(q_0)}{\overset{\delta_1}{\rightsquigarrow}} s_1' \xrightarrow{a_1} s_1 \underset{F(q_1)}{\overset{\delta_2}{\rightsquigarrow}} \cdots s_{N-1}' \xrightarrow{a_{N-1}} s_{N-1} \underset{F(q_{N-1})}{\overset{\delta_N}{\rightsquigarrow}} s_N' \xrightarrow{a_N} s_N, \tag{6}$$

$$s_{N-1} = (q',\, x'),\ s_N' = (q',\, x''),\ s_N = (q^*,\, x^*),\ \delta_1 + \delta_2 + \cdots + \delta_N = t,$$

is an execution of length N that leads to the state $(q^*,\, x^*)$ at time t. Let G_i and $R_i = (B_i,\, P_i)$ denote the guard set and the reset relation that correspond to transition $q' \longrightarrow q^*$ in sequence (6).

The theorem is obviously true for the states that can be reached by executions of length 0.

Suppose that the theorem is true for states that can be reached in time Δ by executions of length $N-1$. Let us prove that it is true for executions of length N as well. If the theorem is true for executions of length $N-1$, then at some step of execution the table $\mathcal{T}$ must contain a tuple $r' = (q',\, X',\, \tau')$ such that $x' \in X'$ and $\tau' \leq \delta_1 + \delta_2 + \cdots + \delta_{N-1} = t'$. Since the tuple r' appeared in the table, step 2 of the algorithm was applied to it. Using lemma (2) we can conclude that the set V_i constucted by (3) contains the point x''. Hence, the set X_i' resulting from the reset relation (4) contains the point x^*. That proves the theorem. ■

Theorem 2. *If the algorithm terminates and reports that the state q_{bad} is unreachable in time Δ (step 1), then the state q_{bad} is unreachable in time Δ.*

The **proof** is a direct consequence of theorem (1).

Termination of the algorithm is not guaranteed for systems that may present so-called *zeno* behavior: an infinite number of discrete jumps in a finite amount of time. The following theorem states that termination is guaranteed when the time "consumed" by each loop of discrete transitions of the automaton is bounded from below by a positive number (in our case, at least by δ).

Theorem 3. *If, for any loop $a_1 = (q_1, q_2), a_2 = (q_2, q_3), ..., a_k = (q_k, q_1) \in T$, there exists $i \in [1, k]$ such that the following conditions are satisfied:*

1. $R(a_i) = (0, P)$, that is, the reset of a_i is constant.
2. $\big(P + \mathsf{B}^n_\epsilon(0)\big) \cap G(a_{i+1}) = \emptyset$ (by convention, a_{k+1} is taken to be a_1).

then the algorithm terminates.

The **proof** is quite obvious and is based in the fact that any cycle in the transition graph takes at least one intergration step δ.

3.2 Possible Modifications

Alternative choices could be made at some points in the algorithm. We discuss these possibilities below and comment on their impact on the accuracy and the efficiency of the algorithm.

1. If the table contains two tuples (q, X_1, τ_1) and (q, X_2, τ_2) with the same discrete state, then we replace these tuples by a single tuple $(q, X_1 \cup X_2, \min\{\tau_1, \tau_2\})$. This decreases the size of the table and results in fewer tuples to be explored. On the other hand, since we can only compute an over-approximation of the union $X_1 \cup X_2$, the accuracy of the algorithm might be compromised. Correctness is not affected.
2. At step 1, instead of removing the chosen tuple (q, X, τ) from the table $\mathcal{T}$ we mark the tuple *explored*. Only unexplored tuples are chosen in step 1. Moreover, before adding to $\mathcal{T}$ a new (unexplored) tuple (q, X', τ'), $\mathcal{T}$ is searched for a tuple (q, X'', τ'') such that $X' \subseteq X''$ and $\tau' \geq \tau''$. If such a tuple exists then (q, X', τ') is not added. The status of (q, X'', τ'') (explored or not) is not changed. The correctness of the algorithm is not affected. The size of $\mathcal{T}$ could increase since explored tuples are not removed. On the other hand, new tuples are not added to the table when not necessary, which results in fewer tuples to explore and shorter running time.

4 Ellipsoidal Approximations

The reachability algorithm described in the previous section can work with any representation of convex compact sets, as long as the operations used by the algorithm can be performed effectively on the chosen representation.

The verification tool described here uses ellipsoidal techniques for approximation and reachability analysis. One of the advantages of ellipsoidal methods is that an ellipsoid in ${\mathcal{C}_b}^n$ can be described as a pair $(x, P) \in \mathsf{R}^n \times \mathsf{R}^{n\times n}$, that is, using only $O(n^2)$ space. Time complexity of ellipsoidal operations is also polynomial.[3] The numerical methods that have been used are directly taken from or based on the results described in publications [5,6].

In these works it is shown that ellipsoidal over-approximations of reachable sets can be expressed through ordinary differential equations with coefficients given in explicit analytical form. Other results include parametric representation of ellipsoidal over-approximations of geometric sums and intersections of

[3] As a comparison, the worst-case complexity of polyhedral operations is exponential.

ellipsoids. The reader is referred to the above-mentioned publications for the details of the methods.

Here, we present new techniques that we have developed for operations on ellipsoids and polyhedra and for unions of ellipsoids.

Definition of Ellipsoids Let $\langle l,\, x\rangle$, $l, x \in \mathsf{R}^n$, denote the inner product of l and x. An *ellipsoid* $\mathcal{E}(p, P)$, $p \in \mathsf{R}^n$, $P \in \mathsf{R}^{n\times n}$, $P = P' \geq 0$ [4] is a convex compact set described by the *support function*[5]

$$\rho(l|\,\mathcal{E}(p,\, P)) = \langle l,\, p\rangle + \langle l,\, Pl\rangle^{1/2}.$$

If the matrix P is non-degenerate, the ellipsoid $\mathcal{E}(p,\, P)$ can alternatively be defined as a level set of a quadratic function:

$$\mathcal{E}(p,\, P) = \{x \mid \langle x - p,\, P^{-1}(x-p)\rangle \leq 1\}.$$

Approximation of Unions of Ellipsoids The reachability algorithm of section 3 uses union of convex sets in step 2(b), equation (3). Union is needed also if tuples $(q,\, X_1,\, \tau_1)$, $(q,\, X_2,\, \tau_2) \in \mathcal{T}$ are replaced by a single tuple $(q,\, X_1 \cup X_2,$ $\min\{\tau_1,\, \tau_2\})$, as described in one of the alternative heuristics. Here we describe the algorithm for over-approximating the union of two ellipsoids by an ellipsoid. Such an algorithm should be efficient, since it is likely to be the bottle neck of the basic verification algorithm. It should also exploit the fact that the reachable set changes only slightly in one time-propagation step.

Let us suppose that we want to approximate $\mathcal{E}(p,\, P) \cup \mathcal{E}(r,\, R)$, where P and R are non-degenerate matrices.

The algorithm builds an increasing sequence of ellipsoids:

$$\mathcal{E}(p,\, P) = \mathcal{E}(p_0,\, P_0),\, \mathcal{E}(p_1,\, P_1), ..., \mathcal{E}(p_k,\, P_k), \tag{7}$$

until $\mathcal{E}(p_k,\, P_k) \supseteq \mathcal{E}(r,\, R)$.

$\mathcal{E}(p_{i+1}, P_{i+1})$ is obtained from $\mathcal{E}(p_i, P_i)$ as follows. Given P_i and R, we compute matrices L_i, V_i, D_i and S_i. L_i is a lower triangular matrix that is the result of Cholesky decomposition of the matrix P_i^{-1}: $L_i L_i' = P_i^{-1}$. V_i is a matrix of eigenvectors and D_i is a diagonal matrix of eigenvalues of matrix $C_i = L_i^{-1} R_i^{-1} L_i'^{-1}$ such that $C_i = V_i D_i V_i'$. We denote

$$S_i = L_i V_i \tag{8}$$

and $y_i = S_i'(r - p_i)$.

Then we find a vector x_i^* that is the solution to the non-convex optimization problem

$$J_i(x) = \langle x - y_i,\, D_i(x - y_i)\rangle \rightarrow \max$$

[4] P' is the transpose of matrix P. Similarly, x' is the transpose of vector x.

[5] The support function $\rho(l|\,X)$ of $X \in \mathcal{C}_b{}^n$ is defined as $\rho(l|\,X) = \max\limits_{x\in X}\langle l,\, x\rangle$. Inversely, a support function uniquely defines a convex compact set.

with the constraint $\|x\| \leq 1$. In [12] it is shown how the problem can be solved and it is proved that the result is the global maximum.

We will denote $l_i^* = \frac{S_i^{-1\prime} x_i^*}{\|S_i^{-1\prime} x_i^*\|}$.

If $J_i(x_i^*) \geq 1$, it means that $\mathcal{E}(p_i,\, P_i) \supseteq \mathcal{E}(r,\, R)$ and the algorithm terminates.

If $J_i(x_i^*) < 1$, we compute:

$$p_{i+1} = S_i^{\prime -1} \left(y_i + \frac{d_i}{2} x_i^* \right) + p_i, \tag{9}$$

and

$$P_{i+1} = S_i^{\prime -1} \left((1 + \alpha_i^{-1}) D_i^{-1} + (1 + \alpha_i) \frac{d_i^2}{4} x_i^* x_i^{*\prime} \right) S_i^{-1}, \tag{10}$$

where

$$d_i = 1 - \sqrt{\langle x_i^*,\, D_i^{-1} x_i^* \rangle} - \langle x_i^*,\, y_i \rangle + \epsilon, \tag{11}$$

$$\alpha_i = \frac{2\sqrt{\langle x_i^*,\, D_i^{-1} x_i^* \rangle}}{d_i \, \langle x_i^*,\, x_i^* \rangle^2}, \tag{12}$$

and ϵ is any non-negative number.

Lemma 3. *(See [5]) Let $\mathcal{E}_1 = \mathcal{E}(q_1, Q_1)$ and $\mathcal{E}_2 = \mathcal{E}(q_2, Q_2)$.*

1. *The ellipsoid $\mathcal{E} = \mathcal{E}(q_1 + q_2, Q(\beta))$, where $\beta > 0$ and*

$$Q(\beta) = (1 + \beta^{-1}) Q_1 + (1 + \beta) Q_2,$$

is properly defined and is an external approximation of the geometrical sum $\mathcal{E}_1 + \mathcal{E}_2$, i.e.

$$\mathcal{E}_1 + \mathcal{E}_2 \subseteq \mathcal{E}(q_1 + q_2, Q(\beta))$$

for any $\beta > 0$.

2. *With vector $l \in \mathsf{R}^n, \|l\| = 1$, given, the equality*

$$p = \sqrt{\frac{\langle Q_1 l,\, l \rangle}{\langle Q_2 l,\, l \rangle}}$$

defines a scalar parameter p, such that

$$\rho(l|\, \mathcal{E}(q_1 + q_2, Q(\beta))) = \rho(l|\, \mathcal{E}(q_1, Q_1) + \mathcal{E}(q_2, Q_2)).$$

(The approximation $\mathcal{E}(q_1 + q_2,\, Q(\beta))$ touches the exact sum in direction l.)

Lemma 4. *Given any $\epsilon > 0$, for the ellipsoid $\mathcal{E}(p_{i+1},\, P_{i+1})$ in sequence (7) the following holds:*

1.

$$\mathcal{E}(p_i,\, P_i) \subset \mathcal{E}(p_{i+1},\, P_{i+1}); \tag{13}$$

2.

$$\rho(l_i^* | \mathcal{E}(p_{i+1}, P_{i+1})) = \rho(l_i^* | \mathcal{E}(r, R)) + \epsilon. \tag{14}$$

Proof The linear transformation

$$x' = S_i'(x - p_i),$$

transforms the ellipsoid $\mathcal{E}(p_i, P_i)$ into the unit ball $\mathcal{E}(0, I_n) = k\{x| \langle x, x\rangle \leq 1\}$ [6], and transforms the ellipsoid $\mathcal{E}(r, R)$ into the ellipsoid $\mathcal{E}(y_i, D_i)$. The next ellipsoid in the sequence $\mathcal{E}(p_{i+1}, P_{i+1})$ as specified by (9) and (10) is the result of the reverse transformation applied to the ellipsoid

$$\mathcal{E}' = \mathcal{E}\left(y_i + \frac{d_i}{2}x_i^*,\, (1+\alpha_i^{-1})D_i^{-1} + (1+\alpha_i)\frac{d_i^2}{4}x_i^* x_i^{*\prime}\right)$$

which, according to lemma (3), is an external approximation of the sum

$$\mathcal{E}(y_i, D_i) + \mathcal{E}\left(\frac{d_i}{2}x_i^*,\, \frac{d_i^2}{4}x_i^* x_i^{*\prime}\right)$$

It is not difficult to see that $0 \in \mathcal{E}\left(\frac{d_i}{2}x_i^*,\, \frac{d_i^2}{4}x_i^* x_i^{*\prime}\right)$, which ensures that $\mathcal{E}' \supseteq \mathcal{E}(y_i, D_i)$, and $x_i^* \in \mathcal{E}\left(\frac{d_i}{2}x_i^*,\, \frac{d_i^2}{4}x_i^* x_i^{*\prime}\right)$. Also, the choice of parameters α_i and d_i ensures that

$$\rho(x_i^* | \mathcal{E}(0, I_n)) + \epsilon = \rho\left(x_i^* \,\middle|\, \mathcal{E}(y_i, D_i) + \mathcal{E}\left(\frac{d_i}{2}x_i^*,\, \frac{d_i^2}{4}x_i^* x_i^{*\prime}\right)\right) = \rho(x_i^* | \mathcal{E}').$$

The later implies (14). ■

Theorem 4. *For any $\epsilon > 0$ the algorithm always terminates in a finite number of steps.*

Proof Because the support functions $\rho(l| \mathcal{E}(p_{i+1}, P_{i+1}))$ and $\rho(l| \mathcal{E}(r, R))$ are continuous, for any $\epsilon > 0$ there is $\delta > 0$ such that for any $l : \|l - l_i^*\| \leq \delta$ the following inequality holds

$$\rho(l| \mathcal{E}(p_{i+1}, P_{i+1})) > \rho(l| \mathcal{E}(r, R)),$$

which means that no l_j can belong to the set $\{l| \|l - l_i^*\| \leq \delta\}$. If at some step l_j belongs to this set, $J_j(x_j^*) > 1$ and the algorithm terminates.

There can be only a finite number of $l_i : \|l_i\| = 1$ such that for any i, j: $\|l_i - l_j\| > \delta$. Therefore, the algorithm always terminates in a finite number of steps. ■

In practice, if the ellipsoid $\mathcal{E}(p, P)$ has to be extended just "slightly" in order to contain $\mathcal{E}(r, R)$, which is the case when approximating the union of reachable sets at successive time steps, it is likely that the union algorithm terminates after a single step, i.e., $\mathcal{E}(p_1, P_1) \supseteq \mathcal{E}(r, R)$.

[6] Here $I_n \in \mathsf{R}^{n\times n}$ denotes the identity matrix

Intersections of Ellipsoids and Polyhedra Guards of discrete transitions are usually given in terms of conjunctions of linear inequalities, which define a polyhedron. Here we discuss a method for approximating intersections of ellipsoids and polyhedra.

In order to approximate the intersection of an ellipsoid $\mathcal{E}$ and a polyhedron H, we first compute ellipsoidal over-approximations of the intersection of $\mathcal{E}$ with each of the facets of H. Then, we compute the intersection of the resulting ellipsoids. Since each facet of H is a half-space, we now show how to approximate the intersection of an ellipsoid and a half-space.

Theorem 5. *Suppose that the ellipsoid*

$$\mathcal{E}(q,\, Q) = \{x |\, \langle x - q,\, Q^{-1}(x - q)\rangle \leq 1\},$$

where $Q = Q^T > 0$, and the half-space

$$S = \{x |\, \langle b,\, x\rangle \geq \alpha\}$$

have a non-empty intersection.

1. *Then for any $p \in [0,\, \frac{\alpha'+1}{2})$ the ellipsoid*

$$\mathcal{E}(q_+(p),\, Q_+(p)) = \{x |\, \langle x - q_+(p),\, Q_+^{-1}(p)(x - q_+(p))\rangle \leq 1\}$$

is an external approximation of the intersection $\mathcal{E}(q,\, Q) \cap S$, where

$$q_+(p) = q + pP^{-1}e_1,\; Q_+^{-1}(p) = PCP^T,$$

and

$$P = VD^{1/2}B,\; e_1 = (1,\, 0, ..., 0)^T,$$

$$C = \begin{pmatrix} \beta_1 & 0 & \cdots & 0 \\ 0 & \beta & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \beta \end{pmatrix},$$

$$\beta_1 = \frac{1}{(p-1)^2},\; \beta = \frac{\alpha' + 1 - 2p}{(\alpha' + 1)(p-1)^2},\; \alpha' = \frac{\alpha - \langle b,\, q\rangle}{\sqrt{\langle b,\, b\rangle}}, \tag{15}$$

$$D = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix},\; V = (v_1\; v_2\; ...\; v_n),$$

where $\lambda_1, ..., \lambda_n$ are eigenvalues of matrix Q^{-1} and $v_1, ..., v_n$ are corresponding eigenvectors that are linearly independent and $\|v_i\| = 1$, $i = \overline{1, n}$,

$$B = (b'\; b_2\; b_3 \ldots b_n),\; b' = \frac{1}{\sqrt{\langle b,\, b\rangle}} D^{-1/2} V^T b,$$

$\{b_2,\, b_3, ..., b_n\}$ is an orthonormal basis of subspace $\{x |\, \langle x,\, b'\rangle = 0\}$.

2. The ellipsoid $\mathcal{E}(q_+(p),\, Q_+(p))$ touches $\mathcal{E}(q,\, Q)$ at point

$$x^* = B^T D^{1/2} V^T (e_1 - q). \tag{16}$$

3.

$$\bigcap_{p\in[0,\, \frac{\alpha'+1}{2})} \mathcal{E}(q_+(p),\, Q_+(p)) = \mathcal{E}(q,\, Q) \cap S. \tag{17}$$

Proof The fact that matrix Q (and Q^{-1} as well) is self-adjoint and positive definite gives us the following equalities:

$$V^T V = V V^T = I,\ B^T B = B B^T = I.$$

Also note that

$$Q^{-1} = V D V^T.$$

Let us apply a linear transformation:

$$x = V D^{-1/2} B x' + q, \tag{18}$$

then the following holds:

$$\begin{aligned}&\langle x - q,\, Q^{-1}(x - q)\rangle = \\ &\langle V D^{-1/2} B x',\, Q^{-1} V D^{-1/2} B x'\rangle = \\ &\langle x',\, B^T D^{-1/2} V^T Q^{-1} V D^{-1/2} B x'\rangle = \langle x',\, x'\rangle.\end{aligned}$$

Thus, transformation (18) converts the ellipsoid $\mathcal{E}(q,\, Q)$ to the unit ball

$$B_0 = \{x |\, \langle x,\, x\rangle = 1\}.$$

At the same time transformation (18) converts the half-space S to the half-space

$$S' = \{x' |\, \langle x',\, B^T D^{-1/2} V^T b\rangle \leq \alpha - \langle b,\, q\rangle\} = \{x' |\, \langle x',\, \tfrac{1}{\sqrt{\langle b,\, b\rangle}} B^T D^{-1/2} V^T b\rangle \leq \alpha'\}.$$

Due to the selection of the matrix B:

$$\frac{1}{\sqrt{\langle b,\, b\rangle}} B^T D^{-1/2} V^T b = e_1,$$

thus

$$S' = \{x' |\, \langle x',\, e_1\rangle \geq \alpha'\}.$$

Now we take an ellipsoid that is defined by the matrix

$$C^{-1}(p) = \begin{pmatrix} \beta_1 & 0 & \cdots & 0 \\ 0 & \beta & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \beta \end{pmatrix},$$

where β_1 and β are determined by (15) and center

$$c(p) = (p,\, 0,\, 0, ..., 0).$$

If $p \in [0,\, \frac{\alpha'+1}{2})$ the ellipsoid is defined and it is not difficult to see that

$$\mathcal{E}(c(p),\, C(p)) \supset B_0 \cap S'.$$

Moreover,

$$\mathcal{E}(c(p),\, C(p)) \cap S' = B_0 \cap S'$$

and the ellipsoid $\mathcal{E}(c(p),\, C(p))$ touches B_0 at point $e_1 = (1,\, 0,\, 0, ..., 0)$.

It is not difficult to see also that

$$\mathcal{E}(c(p),\, C(p)) \subset \{(x_1,\, 0,\, 0, ..., 0) |\, x_1 \geq 2p - 1\}.$$

If $p \to \frac{\alpha'+1}{2}$ then $2p - 1 \to \alpha'$. Also notice that if $p = 0$, $\mathcal{E}(c(p),\, C(p)) = B_0$. Therefore

$$\bigcap_{p \in [0,\, \frac{\alpha'+1}{2})} \mathcal{E}(c(p),\, C(p)) = B_0 \cap S'.$$

If we apply the reverse transformation

$$x' = B^T D^{1/2} V^T (x - q),$$

to the ellipsoid $\mathcal{E}(c(p),\, C(p))$, we will get the ellipsoid $\mathcal{E}(q_+(p),\, Q_+(p))$ as defined above. Properties (16) and (17) will be satisfied and point e_1 at which $\mathcal{E}(c(p),\, C(p))$ touches B_0 will be converted to point x^* as defined by (16) and ellipsoids will touch each other at the point x^*. ■

Intersection Check The problem of checking whether two non-degenerate ellipsoids $E(p_1,\, P_1)$ and $E(p_2,\, P_2)$ intersect is equivalent to a convex quadratic optimization problem:

$$J(x) = \big\langle x - p_1,\, P_1^{-1}(x - p_1)\big\rangle \to \min$$

with constraint

$$\big\langle x - p_2,\, P_2^{-1}(x - p_2)\big\rangle \leq 1.$$

If x^* is the solution to the problem and $J(x^*) > 1$, then the ellipsoids do not intersect. Otherwise, they do intersect.

In order to check whether an ellipsoid and a polyhedron intersect, it is possible to check whether the ellipsoid intersects with all half-spaces that form the faces of the polyhedron. If it does not intersect at least with one of them, the ellipsoid does not intersect the polyhedron. Of course, this method is quite coarse but it is simple and effective. If the intersection is actually empty and the above check did not find that out, that still can be discovered during computation of the intersection.

5 Implementaion of the Tool

The techniques are implemented in the verification tool called *VeriSHIFT* [7]. The tool is a C++ library that consists of all necessary numerical algorithms: ellipsoidal and polyhedral [8] representation of convex sets and operations on them, reachability algorithms, verification algorithms described in this paper, etc. The user of the tool writes C++ code in order to describe a model: for each class of hybrid automaton the user writes a definition of a C++ class derived from the special class *HybridObject* provided by the library. The model can be defined in terms of high level notions such as discrete states, transitions, input/output continuous variables, events, bound convex sets, as described in [1]. Each of above notions is defined as a class in the library. Actions taken upon discrete transitions can be described as C++ functions.

The library provides the notion of discrete configuration also implemented as a class. A discrete configuration contains a set of objects with their discrete states, dataflow and configuration connections and is accompanied by a bound convex set containing possible valuations of the continuous variables.

Together with discrete states, continuous variables and events classes describing hybrid automata can contain variables of any possible C++ type including other hybrid automata classes. There is only one requirement: classes have to able to create their copies in other discrete configurations. Objects within the same discrete configuraion can use any mechanism provided by C++ for communicating with each other as long as that does not interfere with the mechanism provided by the *VeriSHIFT* library. In a function called upon a discrete transition of an object the object can modify its private data, can create/destroy other objects or can call methods of other objects.

Execution of a typical program using *VeriSHIFT* starts with creating an initial discrete configuration: creating an empty configuration, creating new objects within the configuration, setting up connections between objects. Once the initial configuration is set up, verification is started by calling a special library function.

In order to verify the properties of the model, the user can observe what discrete states the objects enter or can assign special actions to transitions they are interested in. Also it is possible to examine reachable sets of continuous variables at any phase of the execution.

Acknowledgements

We would like to thank Alexander B. Kurzhanski and Pravin Varaiya for making this work possible.

[7] The source code of the tool along with the documentation and examples can be found at `http://robotics.EECS.Berkeley.EDU/~olegb/VeriSHIFT/`

[8] The Polyhedral Library 2.0 by Doran Wilde and Herve Le Verge has been used.

References

1. Botchkarev O., Ellipsoidal Techniques for Verification of Hybrid Systems, 2000. Available at: `http://robotics.eecs.berkeley.edu/~olegb/VeriSHIFT/`.
2. Dang T. and Maler O., Reachability Analysis via Face Lifting. In T.A. Henzinger and S. Sastry (Eds), Hybrid Systems: Computation and Control , 96-109, LNCS 1386, Springer, 1998.
3. Henzinger T, Ho P. and Wong-Toi H., HyTech: A Model Checker for Hybrid Systems. In *Software Tools for Technology Transfer*, 1, 1997.
4. Kurzhanski A. B. and Filippova T. F., On the Theory of Trajectory Tubes: a Mathematical Formalism for Uncertain Dynamics, Viability and Control, in: *Advances in Nonlinear Dynamics and Control*, ser. PSCT 17, pp.122 - 188, Birkhäuser, Boston, 1993.
5. Kurzhanski A. B. and Vályi I. *Ellipsoidal Calculus for Estimation and Control*, Birkhäuser, Boston, ser.SCFA, 1996.
6. Kurzhanski A. B. and Varaiya P., Ellipsoidal Techniques for Reachability Analysis, 2000. *Proceedings of this conference.*
7. Puri A., Borkar V. and Varaiya P., ϵ-Approximations of Differential Inclusions, in: R.Alur, T.A.Henzinger, and E.D.Sonntag eds., *Hybrid Systems*, pp. 109 – 123, LNCS 1201, Springer, 1996.
8. Puri A. and Varaiya P., Decidability of Hybrid Systems with Rectangular Differential Inclusions, in D.Dill ed., *Proc. CAV'94*, LNCS 1066, Springer, 1966.
9. Rockafellar, R. T., *Convex Analysis,* Princeton University Press, 1970.
10. Varaiya P., Reach Set Computation Using Optimal Control, in *Proc.of KIT Workshop on Verification of Hybrid Systems*, Verimag, Grenoble, 1998.
11. *VeriSHIFT* Home Page. `http://robotics.EECS.Berkeley.EDU/~olegb/VeriSHIFT/`.
12. Ye Y., On Affine Scaling Algorithms for Non-Convex Quadratic Programming. In Mathematical Programming, 56(1992), pp. 285 – 300.