Using Redundant Constraints for Refinement

Eugene Asarin[1], Thao Dang[2], Oded Maler[2], and Romain Testylier[2]

[1] LIAFA,
CNRS and Université Paris Diderot
Case 7014 75205 Paris Cedex 13, France
[2] VERIMAG,
2 avenue de Vignate,
38610 Gières, France

Abstract. This paper is concerned with a method for computing reachable sets of linear continuous systems with uncertain input. Such a method is required for verification of hybrid systems and more generally embedded systems with mixed continuous-discrete dynamics. In general, the reachable sets of such systems (except for some linear systems with special eigenstructures) are hard to compute exactly and are thus often over-approximated. The approximation accuracy is important especially when the computed over-approximations do not allow proving a property. In this paper we address the problem of refining the reachable set approximation by adding redundant constraints which allow bounding the reachable sets in some critical directions. We introduce the notion of directional distance which is appropriate for measuring approximation effectiveness with respect to verifying a safety property. We also describe an implementation of the reachability algorithm which favors the constraint-based representation over the vertex-based one and avoids expensive conversions between them. This implementation allowed us to treat systems of much higher dimensions. We finally report some experimental results showing the performance of the refinement algorithm.

1 Introduction

Hybrid systems, that is, systems exhibiting both continuous and discrete dynamics, have been recognized as a high-level model appropriate for embedded systems, since this model can describe, within a unified framework, the logical part and the continuous part of an embedded system. Due to the safety critical features of many embedded applications, formal analysis is a topic of particular interest. Recently, much effort has been devoted to the development of automatic analysis methods and tools for hybrid systems, based on formal verification.
Reachability analysis is a central problem in formal verification. The importance of this problem can be seen in its growing literature. For hybrid systems with simple continuous dynamics that have piecewise constant derivatives, their reachable sets can be computed using linear algebra. This is a basis for the verification algorithms implemented in a number of tools, such as UPPAAL [19], KRONOS [25], HYTECH [13] and PHAVER [9]. For hybrid systems with more complex

continuous dynamics described by differential or difference equations, the reachability problem is much more difficult, and a major ingredient in designing a reachability analysis algorithm for such hybrid systems is an efficient method to handle their continuous dynamics. Hybrid systems can be seen as a combination of continuous and discrete dynamics; their verification thus requires, in addition to computing reachable sets of continuous dynamics, performing operations specific for discrete behavior, such as Boolean operations over reachable sets. It is worth noting first that *exact computation* of reachable sets of linear continuous systems is in general difficult. It was proved in [21] that this is possible for a restricted class with special eigenstructures.

In this paper we revisit the problem of approximating reachable sets of linear continuous systems, which was investigated in [3, 8]. The approximation accuracy is important especially when the computed over-approximations do not allow proving a property. We propose a method for refining a reachable set approximation by adding redundant constraints to bound it in some critical directions. We also introduce the notion of directional distance which is appropriate for measuring approximation effectiveness with respect to verifying a safety property. Although in this paper we focus only on methods for accurately approximating reachable sets of linear continuous systems, it is important to note that we use convex polyhedra to represent reachable sets, the treatment of discrete dynamics in a hybrid system can thus be done using available algorithms for Boolean operations over these polyhedra. The methods presented in this paper can therefore be readily integrated in any existing polyhedron-based algorithm for verification of hybrid automata with linear continuous dynamics, such as [2].

The reachability problem for linear systems has been a topic of intensive research over the past decade. We defer a discussion on related work to the last section. The paper is organized as follows. We start with a brief description of the technique for approximating reachable sets using optimal control [24, 3]. We then present two methods for refining approximations by using redundant constraints. One method can reduce approximation error globally while the other is more local and guided by the property to verify. We also describe an efficient implementation of these methods and some experimental results.

2 Preliminaries

Notation and Definitions Let $\mathbf{x}$, $\mathbf{y}$ be two points in $\mathbb{R}^n$ and X, Y be two subsets of $\mathbb{R}^n$. We denote by $\langle \mathbf{x}, \mathbf{y} \rangle$ the scalar product of $\mathbf{x}$ and $\mathbf{y}$. The Hausdorff semi-distance from X to Y is defined as $h_+(X,Y) = \sup_{\mathbf{x} \in X} \ \inf_{\mathbf{y} \in Y} \{||\mathbf{x} - \mathbf{y}||\}$ where $|| \cdot ||$ is the Euclidian norm. The Hausdorff distance between X and Y is $h(X,Y) = \sup \ \{h_+(X,Y), h_+(Y,X)\}$.

We now consider a discrete-time linear system described by the following equation

$$\mathbf{x}(k+1) = A\mathbf{x}(k) + \mathbf{u}(k) \tag{1}$$

where $\mathbf{x} \in \mathcal{X}$ is the state of the system and $\mathbf{u} \in U$ is the input. The input set U is a bounded convex polyhedron in $\mathbb{R}^n$. We assume a set $\mathcal{U}$ of admissible

input functions consisting of functions of the form $\boldsymbol{\mu} : \mathbb{Z}^+ \to U$. Note that any system $\mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{v}(k)$ where $\mathbf{v}(k) \in \mathcal{V} \subset \mathbb{R}^m$ can be transformed into a system of the form (1) by letting $\mathbf{u} = B\mathbf{v}$ and defining the input set $U = \{\mathbf{u} \mid \mathbf{u} = B\mathbf{v} \ \wedge \ \mathbf{v} \in \mathcal{V}\}$.
A trajectory of (1) starting from a point $\mathbf{y} \in \mathcal{X}$ under a given input $\boldsymbol{\mu} \in \mathcal{U}$ is the solution $\xi_{\boldsymbol{\mu}}(\mathbf{y}, k)$ of (1) with the initial condition $\mathbf{x}(0) = \mathbf{y}$, that is $\forall k > 0 \ \xi_{\boldsymbol{\mu}}(\mathbf{y}, k+1) = A\xi_{\boldsymbol{\mu}}(\mathbf{y}, k) + \boldsymbol{\mu}(k)$.
Let $\mathcal{I}$ be a subset of $\mathcal{X}$. The set of states reachable from $\mathcal{I}$ at a time point k under a given input $\boldsymbol{\mu} \in \mathcal{U}$, denoted by $Post_{\boldsymbol{\mu}}(\mathcal{I}, k)$, is the set of states visited at time k by all the trajectories starting from points in $\mathcal{I}$ under $\boldsymbol{\mu}$: $Post_{\boldsymbol{\mu}}(\mathcal{I}, k) = \bigcup\limits_{\mathbf{y} \in \mathcal{I}} \xi_{\boldsymbol{\mu}}(\mathbf{y}, k)$. The set of all states reachable from $\mathcal{I}$ at time point k is $Post(\mathcal{I}, k) = \bigcup\limits_{\boldsymbol{\mu} \in \mathcal{U}} Post_{\mathcal{I}, \boldsymbol{\mu}}(k)$.

Reachability Problem The reachability problem we address in this paper is stated as follows. Given a set $\mathcal{I} \subset \mathcal{X}$ of initial states, we want to compute the reachable set $Post(\mathcal{I}, k)$ of the system (1) up to some time $k = k_{max}$.

2.1 Reachability Algorithm

With a view to safety verification, our goal is to obtain conservative approximations. To this end, we make use of the technique, which was first suggested in [24] and then applied in [3, 8, 2] for hybrid systems. This technique is based on the Maximum Principle in optimal control [14]. In the following, we recap the main idea of this technique.
We assume that the initial set is a bounded convex polyhedron described as the intersection of a set $\mathcal{H} = \{H(\mathbf{a}_1, \mathbf{y}_1), \ldots, H(\mathbf{a}_m, \mathbf{y}_m)\}$ of m half-spaces: $\mathcal{I} = \bigcap\limits_{i=0}^{m} H(\mathbf{a}_i, \mathbf{y}_i)$ where each half-space $H(\mathbf{a}_i, \mathbf{y}_i) \in \mathcal{H}$ is defined as $H(\mathbf{a}_i, \mathbf{y}_i) = \{\mathbf{x} \mid \langle \mathbf{a}_i, \mathbf{x} \rangle \leq \langle \mathbf{a}_i, \mathbf{y}_i \rangle = \gamma_i\}$. We say that $\langle \mathbf{a}_i, \mathbf{x} \rangle \leq \langle \mathbf{a}_i, \mathbf{y}_i \rangle$ is the linear constraint associated with $H(\mathbf{a}_i, \mathbf{y}_i)$. For simplicity, we sometimes use $H(\mathbf{a}_i, \mathbf{y}_i)$ to refer to both the half-space and its associated constraint. The polyhedron defined by the intersection of the half-spaces of $\mathcal{H}$ is denoted by $Poly(\mathcal{H})$. We emphasize that in the above description of $\mathcal{I}$, the set of associated constraints may not be minimal, in the sense that it may contain redundant constraints. The vector $\mathbf{a}_i$ is called the *outward normal* of the half-space $H(\mathbf{a}_i, \mathbf{y}_i)$ and $\mathbf{y}_i$ is an arbitrary point of $Poly(\mathcal{H})$ on the boundary of the half-space $H(\mathbf{a}_i, \mathbf{y}_i)$, which we call a *supporting point* of $H(\mathbf{a}_i, \mathbf{y}_i)$.
By the Maximum Principle [14], for every half-space $H(\mathbf{a}, \mathbf{y})$ of $\mathcal{I}$ with the outward normal $\mathbf{a}$ and a supporting point $\mathbf{y}$, there exists an input $\boldsymbol{\mu}^* \in \mathcal{U}$ such that computing the successors of $H(\mathbf{a}, \mathbf{y})$ under $\boldsymbol{\mu}^*$ is sufficient to derive a tight polyhedral approximation of $Post(\mathcal{I}, k)$. It can be proved that the evolution of the normal of each $H(\mathbf{a}, \mathbf{y})$ is governed by the adjoint system whose dynamics is described by the following equation:

$$\boldsymbol{\lambda}(k+1) = -A^T \boldsymbol{\lambda}(k). \tag{2}$$

The solution to (2) with the initial condition $\boldsymbol{\lambda}(0) = \mathbf{a}$ is denoted by $\lambda_{\mathbf{a}}(k)$. The following proposition shows that one can compute *exactly* the reachable set from a half-space.

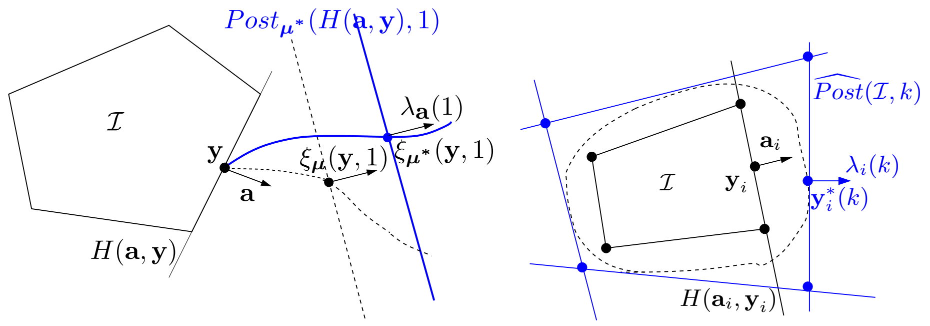
Proposition 1. *Let $H(\mathbf{a},\mathbf{y}) = \{\mathbf{x} \mid \langle \mathbf{a},\mathbf{x}\rangle \leq \langle \mathbf{a},\mathbf{y}\rangle\}$ be a half-space with the normal $\mathbf{a}$ and a supporting point $\mathbf{y}$. Let $\lambda_{\mathbf{a}}(k)$ be the solution to (2) with the initial condition $\boldsymbol{\lambda}(0) = \mathbf{a}$. Let $\boldsymbol{\mu}^*(k) \in \mathcal{U}$ be an input such that for every $k \geq 0$*

$$\boldsymbol{\mu}^*(k) \in \arg\max\{\langle \lambda_{\mathbf{a}}(k), \mathbf{u}\rangle \mid \mathbf{u} \in U\}.$$

Let $\xi_{\boldsymbol{\mu}^*}(\mathbf{y},k)$ be the solution to (1) under the above input $\boldsymbol{\mu}^*$ with the initial condition $\mathbf{x}(0) = \mathbf{y}$. Then, the reachable set from $H(\mathbf{a},\mathbf{y})$ is the half-space with the normal $\lambda_{\mathbf{a}}(k)$ and the supporting point $\xi_{\boldsymbol{\mu}^*}(\mathbf{y},k)$:

$$Post(H(\mathbf{a},\mathbf{y}),k) = Post_{\boldsymbol{\mu}^*}(H(\mathbf{a},\mathbf{y}),k) = \{\mathbf{x} \mid \langle \lambda_{\mathbf{a}}(k),\mathbf{x}\rangle \leq \langle \lambda_{\mathbf{a}}(k), \xi_{\boldsymbol{\mu}^*}(\mathbf{y},k)\rangle\}. \tag{3}$$

We remark that the evolution of the normal $\lambda_{\mathbf{a}}(k)$ is independent of the input; therefore for all $\boldsymbol{\mu} \in \mathcal{U}$ the boundaries of the half-spaces $Post_{\boldsymbol{\mu}}(H(\mathbf{a},\mathbf{y}),k)$ are parallel to each other, as shown in Figure 1. ■



$Post_{\boldsymbol{\mu}^*}(H(\mathbf{a},\mathbf{y}),1)$
$\mathcal{I}$
$\mathbf{y}$
$\mathbf{a}$
$\lambda_{\mathbf{a}}(1)$
$\xi_{\boldsymbol{\mu}}(\mathbf{y},1)$
$\xi_{\boldsymbol{\mu}^*}(\mathbf{y},1)$
$H(\mathbf{a},\mathbf{y})$
$\widehat{Post}(\mathcal{I},k)$
$\mathbf{a}_i$
$\lambda_i(k)$
$\mathcal{I}$
$\mathbf{y}_i$
$\mathbf{y}_i^*(k)$
$H(\mathbf{a}_i,\mathbf{y}_i)$

Fig. 1. Left figure: The solid and the dotted curves are the trajectories $\xi_{\boldsymbol{\mu}^*}(\mathbf{y},k)$ under $\boldsymbol{\mu}^*$ and $\xi_{\boldsymbol{\mu}}(\mathbf{y},k)$ under $\boldsymbol{\mu}$. At the first step, the reachable set $Post_{\boldsymbol{\mu}^*}(H(\mathbf{a},\mathbf{y}),1)$ is the half-space determined by the normal $\lambda_{\mathbf{a}}(1)$ and the supporting point $\xi_{\boldsymbol{\mu}^*}(\mathbf{y},1)$. **Right figure**: Over-approximating the exact reachable set $Post(\mathcal{I},k)$ (shown in dotted line) by the polyhedron $\widehat{Post}(\mathcal{I},k)$.

The following proposition [24, 8] shows that one can *conservatively* approximate the reachable set from the convex polyhedron $\mathcal{I} = \bigcap_{i=0}^{m} H(\mathbf{a}_i,\mathbf{y}_i)$.

Proposition 2. *For every $i \in \{1, \dots, m\}$, let $\boldsymbol{\mu^*}_i(k)$ be an admissible input such that for every $k \geq 0$ $\boldsymbol{\mu^*}_i(k) \in \arg\max\{\langle \lambda_{\mathbf{a}_i}(k), \mathbf{u}\rangle \mid \mathbf{u} \in U\}$. Then,*

$$Post(\mathcal{I}, k) \subseteq \bigcap_{i=0}^{m} \{\mathbf{x} \mid \langle \lambda_{\mathbf{a}_i}(k), \mathbf{x}\rangle \leq \langle \lambda_{\mathbf{a}_i}(k), \xi_{\boldsymbol{\mu^*}_i}(\mathbf{y}_i, k)\rangle\}.$$

Proposition 2 provides the following scheme to over-approximate $Post(\mathcal{I}, k)$ by $\widehat{Post}(\mathcal{I}, k)$. For brevity we denote $\mathbf{y}_i^*(k) = \xi_{\boldsymbol{\mu^*}_i}(\mathbf{y}_i, k)$ and $\lambda_i(k) = \lambda_{\mathbf{a}_i}(k)$.

$$\lambda_i(k+1) = -A^T \lambda_i(k);\ \lambda_i(0) = \mathbf{a}_i, \tag{4}$$

$$\mathbf{y}_i^*(k+1) = A\mathbf{y}_i^*(k) + \boldsymbol{\mu^*}_i(k);\ \mathbf{y}_i^*(0) = \mathbf{y}_i, \tag{5}$$

$$\boldsymbol{\mu^*}_i(k) \in \arg\max\ \{\langle \lambda_i(k), \mathbf{u}\rangle \mid \mathbf{u} \in U\}. \tag{6}$$

Algorithm 1 Over-approximating $Post_{\mathcal{I}}(k)$

```
for $i = 1, \dots, m$ do
    Compute $\lambda_i(k)$ by solving (4).
    Compute $\boldsymbol{\mu^*}_i(k) = \arg\max\ \{\langle \boldsymbol{\lambda}_i(k), \mathbf{u}\rangle \mid \mathbf{u} \in U\}$.
    Compute $\mathbf{y}_i^*(k)$ by solving (5) with $\boldsymbol{\mu^*}_i(k)$ obtained in the previous step.
end for
$\widehat{Post}(\mathcal{I}, k) = \bigcap_{i=1}^{m}\ \{\mathbf{x} \mid \langle \lambda_i(k), \mathbf{x}\rangle \leq \langle \lambda_i(k), \mathbf{y}_i^*(k)\rangle\}$.
```

Note that if the input set U is a bounded convex polyhedron then $\boldsymbol{\mu^*}_i(k)$ can be selected at one of its vertices at every time point k. The last step consists in intersecting all the half-spaces defined by the normal vectors $\lambda_i(k)$ and the points $\mathbf{y}_i^*(k)$ to obtain the convex polyhedron $\widehat{Post}(\mathcal{I}, k)$, which is an over-approximation of $Post(\mathcal{I}, k)$ (see Figure 1 for an illustration of the algorithm).

2.2 Approximation Accuracy

As a metric for our approximations, we use the Hausdorff distance introduced in the beginning of Section 2. This is a good measure for differences between sets. The approximation error is defined as: $\epsilon_k = h(\widehat{Post}(\mathcal{I}, k), Post(\mathcal{I}, k))$.

Proposition 3. *For a linear system $\mathbf{x}(k+1) = A\mathbf{x}(k) + \mathbf{u}(k)$ where the set of input function is a singleton, the equality in Proposition 2 is achieved, that is the set $\widehat{Post}(\mathcal{I}, k) = Post(\mathcal{I}, k)$.*

To prove this result, we notice that the image of the intersection of two sets X and Y by a function g satisfies $g(X \cap Y) \subseteq g(X) \cap g(Y)$. In particular, if g is injective, $g(X \cap Y) = g(X) \cap g(Y)$. Indeed, the solutions of deterministic systems are injective since they do not cross one another. On the contrary, the solutions of non-deterministic systems are generally not injective (since from different initial states, different inputs may lead the system to the same state). This implies that for linear systems with uncertain input, Algorithm 1 produces only an over-approximation of this set. ■

Proposition 4. *If a half-space H supports the initial set $\mathcal{I}$, that is its boundary contains a point in $\mathcal{I}$, then for every $k > 0$ the half-space $Post(H, k)$ computed as shown in the formula (3) supports the exact reachable set $Post(\mathcal{I}, k)$.*

The proof of this can be straightforwardly established from the fact that the supporting point $\mathbf{y}^*(k) = \xi_{\boldsymbol{\mu}^*}(\mathbf{y}, k)$ of each $Post_k(H)$ is indeed a point in the exact reachable set from $\mathcal{I}$ since it is computed as a successor from a point in $\mathcal{I}$ under an admissible input function. ■

From the above results, to improve the approximation accuracy one can use additional half-spaces in the description of the initial set, such that with respect to the initial set their associated constraints are redundant, but under the system's dynamics they evolve into new half-spaces which can be useful for bounding the approximation in some directions. In order to significantly reduce the approximation error which is measured by the Hausdorff distance between the approximate and the exact sets, it is important to find the area when the difference between these sets is large. In the following, we propose two methods for refining the reachable set approximation by dynamically adding constraints.

3 Refinement Using Sharp Angles

For simplicity we use $\mathcal{H} = \{H_1, \ldots, H_m\}$ with $H_i = H(\mathbf{a}_i, \mathbf{y}_i) = \{\mathbf{x} \mid \langle \mathbf{a}_i, \mathbf{x} \rangle \leq \langle \mathbf{a}_i, \mathbf{y}_i \rangle = \gamma_i\}$ to denote the half-spaces of the reachable set at the k^{th} step. As mentioned earlier, the constraints to add should not change the polyhedron and thus must be redundant. Another requirement is that the corresponding half-spaces should be positioned where the approximation error is large. The over-approximation error indeed occurs mostly around the intersections of the half-spaces. In addition, this error is often large when the angle between two adjacent half-spaces, which can be determined from their normal vectors, is smaller than some threshold σ. We call them *adjacent half-spaces with sharp angle.*

Indeed, when two adjacent half-spaces form a sharp angle, the area near their intersection is elongated, which causes a large difference between the polyhedral approximation and the actual reachable set. Hence, in order to better approximate the exact boundary, one needs to use more approximation directions.

A constraint to add can be determined as follows. Its normal vector $\boldsymbol{\lambda}_n$ can be defined by a linear combination of $\boldsymbol{\lambda}_l$ and $\boldsymbol{\lambda}_j$: $\boldsymbol{\lambda}_n = w_1\boldsymbol{\lambda}_l + w_2\boldsymbol{\lambda}_j$ where $w_1 > 0$, $w_2 > 0$. We next determine a supporting point of the constraint. To this end, we find a point on the facet in the intersection between H_l and H_j by solving the following linear programming problem:

$$\begin{aligned} &\min \langle \boldsymbol{\lambda}_n, \mathbf{x} \rangle \\ \text{s.t. } \forall q \in \{1, \ldots, m\} : q \neq l \ \wedge q \neq j \wedge &\langle \boldsymbol{\lambda}_q, \mathbf{x} \rangle \leq \gamma_q \\ &\langle \boldsymbol{\lambda}_l, \mathbf{x} \rangle = \gamma_l \\ &\langle \boldsymbol{\lambda}_j, \mathbf{x} \rangle = \gamma_j \end{aligned} \tag{7}$$

The solution $\mathbf{x}^*$ of the above LP problem yields the new constraint $\langle \lambda_n, \mathbf{x} \rangle \leq \boldsymbol{\lambda}_n \mathbf{x}^*$, which can be used for refinement purposes.

It is important to note that while sharp angles between half-spaces are useful to identify the areas where the approximation error might be large, sharp angles can also be formed when the system converges to some steady state. In this case, "curving" the approximate set does not significantly improve the accuracy, because their normal vectors under the system's dynamics become very close to each other, and we choose not to add constraints in this case.

Dynamical Refinement In the above we showed how to determine redundant constraints for refinement. The refinement process can be done dynamically as follows. During the computation, if at the k^{th} step, the sharp angle criterion alerts that the approximation error might be large, we move $r \leq k$ steps backwards and add constraints for each pair of adjacent half-spaces with sharp angles. The computation is then resumed from the $(k-r)^{th}$ step.
An important remark is that the larger r is, the more significant accuracy improvement is achieved, at the price of more computation effort. Indeed, when we backtrack until the first step, the half-spaces corresponding to the added constraints actually support the boundary of the initial set. Thus, by Proposition 4, their successors also support the exact reachable set from $\mathcal{I}$, which guarantees *approximation tightness*. On the contrary, if $r < k$, it follows from the above LP problem (7) that the added constraints only support the approximate reachable set and their boundaries therefore are not guaranteed to contain a truly reachable point.
Figure 4 illustrates the improvement in accuracy achieved by this method for a 2-dimensional linear system whose matrix is a Jordan block[3]. The colored parts correspond to the approximation error which is reduced by adding constraints using the sharp angle criterion.

4 Refinement Using Critical Directions

A good compromise between accuracy and computation time depends on the problems to solve and the available computation budget. In this section, we discuss a refinement procedure specific for safety verification problems. As we have seen earlier, in order to guarantee a desired approximation error bound, measured using the Hausdorff distance between the approximate and the exact sets, one needs to assure that their difference does not exceed the bound in all directions. Nevertheless, when reachability analysis is used to prove a safety property, this measure is not appropriate for characterizing the potential of reaching a unsafe zone. In this case, one is more interested in computing an approximation that may not be precise (with respect to the Hausdorff distance to the exact set) but enables deciding whether the exact set intersects with the unsafe set. To this end, we use a measure, called *directional distance*.

[3] A Jordan block is a matrix whose main diagonal is filled with a fixed number and all the entries which are directly above and to the right of the main diagonal are 1.

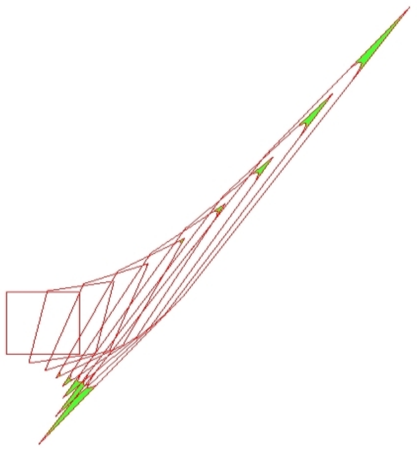


Fig. 2. Adding constraints can reduce approximation error (which is the colored areas in the figure).

4.1 Directional Distance

Given two convex sets A and B in $\mathbb{R}^n$, a Boolean function $contact(A, B) = true$ if and only if the following two conditions are satisfied: (1) $Int(A) \cap Int(B) = \emptyset$; (2) $\delta A \cap \delta B \neq \emptyset$ where $Int(A)$ is the interior of A and δA is its boundary. Intuitively, $contact(A, B)$ is true if and only if A and B intersect with each other and, in addition, they intersect only on their boundaries.
If $A \cap B \neq \emptyset$, the directional distance between A and B is defined as

$$\rho(A, B) = \inf_{\mathbf{p} \in \mathbb{R}^n} \{||\mathbf{p}|| : contact(A, B + \mathbf{p})\}.$$

where $||\mathbf{p}||$ denotes the Euclidian length of the vector $\mathbf{p}$, $B + \mathbf{p} = \{\mathbf{x} + \mathbf{p} \mid \mathbf{x} \in B\}$ is the result of translating the set B by the vector $\mathbf{p}$.
If A and B are in contact, $\rho(A, B) = 0$. Note that if the sets A and B are overlapping, the above $\rho(A, B) > 0$ measures the "depth" of the intersection of the two sets. To generalize this definition to overlapping sets, we need to distinguish this case from the case where A and B do not intersect. To do so, we use a signed version of the directional distance that has positive values if the two sets are non-overlapping and negative values otherwise.

Definition 1. *Given two convex sets A and B in $\mathbb{R}^n$, the signed directional distance between A and B is defined as*

$$\rho_s(A, B) = \begin{cases} \rho(A, B) & \text{if} A \cap B \neq \emptyset, \\ -\rho(A, B) & \text{otherwise.} \end{cases}$$

This directional distance in two and three dimensions has been used in robotics for collision detection [20]. The advantages of using the signed directional distance for the safety verification problem are the following:

- It measures the potential and the degree of collision between two moving objects, which, in the context of reachability computation, provides useful information on the necessity of refining reachable set approximations.
- For convex polyhedral sets, it can be estimated efficiently, as we will show in the sequel.

4.2 Signed Directional Distance Estimation and Refinement Algorithm

The signed directional distance between two convex polyhedra can be computed using existing algorithms (such as in [7]). However, these algorithms are specific for two and three dimensions and require complete polyhedral boundary descriptions (such as their vertices and facets). In high dimensions these descriptions are very expensive to compute, which will be discussed more in Section 5. We therefore focus on the problem of estimating the signed distance using only constraint descriptions of polyhedra. Our solution can be summarized as follows. The underlying idea is based on the relation between the signed directional distance $\rho_s(A,B)$ and the Minkowski difference $A \ominus B$. The latter is defined as follows:

$$A \ominus B = \{\mathbf{b} - \mathbf{a} \; | \mathbf{a} \in A \wedge \mathbf{b} \in B\}.$$

Intuitively, the Minkowski difference contains the translation directions that can bring A into contact with B. Its relation with the signed distance that we exploit is expressed by: $\rho_s(A,B) = \rho_s(\mathbf{0}, A \ominus B)$ where $\mathbf{0}$ is the singleton set that contains the origin of $\mathbb{R}^n$. Again, the vertex description of the Minkowski difference set can be expensive to compute, we resort to a constraint description of this set that can be efficiently computed. To this end, we consider a particular set of translation vectors corresponding to moving the half-space of a face e of A in the direction of its normal $\boldsymbol{\lambda}_e$ by a distance d_e so that the half-space touches B. Then, it can be proved that the half-space $H_e = \{\mathbf{x} \; : \; \langle \boldsymbol{\lambda}_e, \mathbf{x} \rangle \leq d_e\}$ contains at least one face of the exact Minkowski difference $A \ominus B$. Let $\mathcal{H}_e$ be the set of all such half-spaces H_e. This implies that $A \ominus B \subseteq Poly(\mathcal{H}_e) = \bigcap_{\forall e} \{\mathbf{x} \; : \; \langle \boldsymbol{\lambda}_e, \mathbf{x} \rangle \leq d_e\}$. In two or three dimensions, it is possible to obtain the exact constraint description of $A \ominus B$ by considering other translation types (such as an edge of A moves along an edge of B). Nevertheless, this computation requires the vertex descriptions of the polyhedra and we thus omit it.

Since $Poly(\mathcal{H}_e)$ is an over-approximation of $A \ominus B$, it can be proved that

$$\begin{cases} \rho_s(\mathbf{0}, Poly(\mathcal{H}_e)) \leq \rho_s(A,B) & \text{if} A \cap B = \emptyset, \\ \rho_s(\mathbf{0}, Poly(\mathcal{H}_e)) \geq \rho_s(A,B) & \text{otherwise.} \end{cases}$$

The distance $\rho_s(\mathbf{0}, Poly(\mathcal{H}_e))$ can then be easily determined, since it is exactly the largest value of all d_e. We use $\rho_s(\mathbf{0}, Poly(\mathcal{H}_e))$ as an estimate of the signed directional distance between A and B.

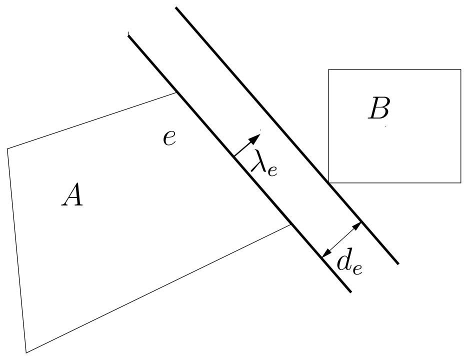


Fig. 3. Estimation of the directional distance.

It is important to note that this estimate is conservative regarding its utility as a critical situation alert. Indeed, if the two sets do not overlap, the result is smaller than the exact separating distance; otherwise, its absolute value is larger than the exact penetration distance. It is important to note again that the above estimation, which does not involve expensive vertex computation, is time-efficient.

In the context of safety verification, the set A plays the role of the reachable set and B the unsafe set. The constraints of A corresponding to the largest values of d_e are called *critical* because they are closest to B with respect to the directional distance measure. Their identification is part of the above computation of the signed directional distance between A and B.

Even if the angle between a critical constraint and one of its adjacent constraints does not satisfy the sharp angle criterion, we still refine around their intersection. The refinement can then be done using the same method for adjacent half-spaces with sharp angles, described in the previous section.

4.3 Refinement Using Constraints from the Safety Specification

Let $\Lambda_{\bar{\mathcal{B}}}$ be the set of normal vectors of the complement of each half-space of the unsafe set $\mathcal{B}$. In many cases, intersection of the reachable set and $\mathcal{B}$ can be tested more easily if the reachable set description contains a constraint whose normal vector coincides with a vector in $\Lambda_{\bar{\mathcal{B}}}$. Hence, for each direction $\boldsymbol{\lambda} \in \Lambda_{\bar{\mathcal{B}}}$, the predecessors of $\boldsymbol{\lambda}$ by the adjoint system can be used to define constraints to add, again by solving the LP problem (7). The refinement using the predecessors of such directions is needed when the reachable set is close to the unsafe set. The refinement procedure using critical directions is summarized in Algorithm 2. In this algorithm, $\mathcal{H}^0$ is the set of constraints of the initial polyhedron $\mathcal{I}$. The function $dir(\mathcal{H}_c^k)$ returns the set of normal vectors of the constrains in $\mathcal{H}_c^k$. The function $AddConstraints(Poly(\mathcal{H}^{k-r}), \Lambda^{k-r})$ adds in $\mathcal{H}^{k-r}$ the new constraints with normal vectors in Λ^{k-r} that support $Poly(\mathcal{H}^{k-r})$.

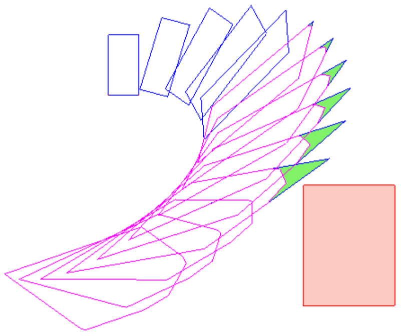


Fig. 4. Refinement using critical directions on a 2-dimensional example.

Algorithm 2 Refinement Using Critical Directions

```
$\mathcal{H}^1 = \widehat{Post}_1(\mathcal{H}^0)$                                    /* One step computation */
$k = 1$
while $k \le k_{max}$ do
  if $\rho_s(\mathcal{H}^k, \mathcal{B}) \le \eta$ then
    $\mathcal{H}_c^k = criticalConstraints(\mathcal{H}^k, \mathcal{B})$      /* $H_c^k$ is the set of critical constraints */
                                        /* $r \le k$ is the number of backtracking steps */
    $\Lambda_c^{k-r} = Pre_r(dir(\mathcal{H}_c^k))$ /* Retrieve $r$-step predecessors of the normal vectors of
    $\mathcal{H}_c^k$
                                        which were computed at the $(k-r)^{th}$ step */
    $\Lambda_b^{k-r} = Pre_r(\Lambda_{\bar{\mathcal{B}}})$                  /* Predecessors of the normal vectors of $\mathcal{B}$ */
    $\Lambda^{k-r} = \Lambda_b^{k-r} \cup \Lambda_c^{k-r}$
    $\mathcal{H}^{k-r} = AddingConstraints(Poly(\mathcal{H}^{k-r}), \Lambda^{k-r})$
    $k = k - r$
  end if
  $\mathcal{H}^{k+1} = \widehat{Post}_1(\mathcal{H}^k)$                               /* One step computation */
  $k = k++$
end while
```

Figure 4 shows the result obtained for a 2-dimensional system in Jordan block form using the above algorithm. The rectangle on the right is the unsafe set. When the reachable set is close to the unsafe set, the algorithm backtracks a number of steps and adds new constraints. This refinement allows approximating more precisely the actual reachable set near the bad set and thus proving that the system does not enter the unsafe set. The colored zones are the parts of the over-approximation error eliminated by the added constraints. It can be seen from the figure that the aprroximation is refined only in the critical zones near the unsafe set.

5 Implementation and Experimental Results

We emphasize that in our development so far the algorithms use the constraint description and do not require the vertex description of polyhedra. Indeed, the transformation from a constraint description to a vertex description is known as vertex enumeration and the inverse transformation is known as facet enumeration. To show the computational complexity of these problems, we mention the algorithm published in [4] which finds m_v vertices of a polyhedron defined by a non-degenerate system of m inequalities in n dimensions (or, dually, the facets of the convex hull of m points in n dimensions, where each facet contains exactly n given points) in time $O(mnm_v)$ and $O(mn)$ space.
From our experience in using polyhedra for reachability computation for continuous and hybrid systems, we noticed that many operations (such as, the convex-hull) are extremely time-consuming, especially in high dimensions. Degeneracy of sets, such as flatness, which occurs frequently in reachable set computation, is also another important factor that limits the scalability of existing polyhedron-based algorithms. It is fair to say that they can handle relatively well systems of dimensions only up to 10. This therefore motivated a lot of research exploiting other set representations, which will be discussed later.
On the other hand, when trying to solve a specific verification problem, it is not always necessary to maintain both the vertex and the constraint descriptions of polyhedra. Indeed, for many tasks in a verification process, vertex enumeration can be avoided, such as in the algorithms we presented so far. We have implemented the above described algorithms of reachability computation with refinement for linear continuous systems and this implementation enabled us to handle continuous systems of dimensions much higher than what can be treated by typical polyhedron-based reachability analysis algorithms, such as [2].
In the following, we present some experimental results obtained using this implementation. To evaluate the performance of our methods, we generated a set of linear systems in Jordan block form in various dimensions up to 100 with the values in the diagonal are all equal to (-0.8). The input set $U = [-0.1, 0.1]^n$ and the initial set $\mathcal{I} = [-1, 1]^n$ are boxes (whose number of constraints equal to $2n$). The threshold for the sharp angle criterion is $\sigma = 60$ degrees.
To show the advantage of the constraint-based implementation, we also tested an implementation using vertex computation on the same examples. In this imple-

mentation with vertex computation, the constraint adjacency information can be directly derived and the constraints to add are easier to compute. However, the cost for vertex computation is high, which limited the application of this implementation to the examples of dimensions only up to 10, as shown in Table 1. For the example in 10 dimensions, we had to fix a smaller maximal number of constraints to add, in order to produce the result in a feasible computation time. The constraint-based implementation is more time-efficient and thus allows us to handle systems of higher dimensions, as shown in Table 2.

dim n	Final number of added constraints	Computation time in seconds
2	32	0.4
5	59	9.14
10	10	150.93
20	–	–
50	–	–
100	–	–

Table 1. Computation time for 100 steps on some linear systems in Jordan block form using the implementation with vertex computation.

dim n	Final number of added constraints	Computation time in seconds
2	32	0.48
5	59	0.76
10	36	2.22
20	38	3.67
50	94	42.07
100	197	335.95

Table 2. Computation time for 100 steps on the same linear systems in Jordan block form using the constraint-based implementation.

6 Summary and Related Work

There have been other works on computing reachable sets of linear continuous systems. The treatment of uncertain inputs can be done by Minkowski sum (such as in the approaches using zonotopes [12, 1], ellipsoids [17, 5, 18], or parallelotopes [15]). This can also be handled by optimization (such as in [16, 6, 22]). On the other hand, various set representations have been investigated. The classes of set representations with special shapes that have been used for reachability computations include oriented hyper-rectangles [23], zonotopes [10, 12,

1], ellipsoids [17, 5, 18], parallelotopes [15]. Compared to general convex polyhedra, their manipulation can be more efficient, but they are less appropriate for approximating sets with complex geometric forms.
To our knowledge, the most scalable algorithm is the recently developed zonotope-based algorithms [12, 1]. The main advantage of zonotopes is that some important operations can be performed very efficiently over zonotopes, such as the Minkowski sum and linear transformations. This allows the zonotope-based algorithms to handle continuous systems of dimensions up to a few hundreds of variables. However, a major difficulty that comes with this set representation is that intersection of zonotopes is hard, which limits the extension of this approach to hybrid systems. Similarly, support functions [11] are a representation for general convex sets, on which the Minkowski sum and linear transformations can be efficiently computed. However, using support functions, set intersection is also difficult, which is an obstacle towards lifting the scalability of the associated algorithms to hybrid systems. In the context of hybrid systems verification, the main advantage of convex polyhedra, in our opinion, is their approximation power. In addition, using well-established techniques for linear programming solving and algorithmic geometry, polyhedral manipulation for specific tasks in verification can be optimized, which was demonstrated in this paper.
Concerning approximation refinement, our method is similar to the well-known counter-example based refinement approaches in the idea of guiding the refinement process using the previously explored behaviors. However, to the best of our knowledge, the idea of using redundant constraints for refinement purposes is new. Another novelty in our results is the use of the directional distance to measure approximation effectiveness in proving safety properties and to guide the refinement process.
The results of this paper open many promising research directions. The application of this method to hybrid systems is straighforward since the polyhedral operations for treating discrete dynamics can be computed using available algorithms. However, a challenge in this task is to go beyond the dimension limits of existing polyhedral computation algorithms, by exploiting the structure and the specificity of hybrid systems as well as of the verification problems. In addition, exploring the Minkowski difference between the reachable set and the unsafe set would allow a better measure of critical proximity of the reachable set under the dynamics of the system.

References

1. M. Althoff, O. Stursberg, and M. Buss, Reachability Analysis of Nonlinear Systems with Uncertain
   Parameters using Conservative Linearization. *CDC'08*, 2008.
2. E. Asarin, T. Dang, and O. Maler. The d/dt tool for verification of hybrid systems. In *Computer Aided Verification*, LNCS 2404, Springer-Verlag, pages 365–370, 2002.
3. E. Asarin, O. Bournez, T. Dang, and O. Maler.
   Approximate reachability analysis of piecewise linear dynamical systems. *HSCC'00*, LNCS 1790, 21-31, 2000.

4. D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete and Computational Geometry* **8**, number 1, pp 295-313, December 1992.
5. O. Botchkarev and S. Tripakis. Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. *Hybrid Systems: Computation and Control*, LNCS 1790, B. Krogh and N. Lynch, Eds. Springer-Verlag, pp 73–88, 2000.
6. A. Chutinan and B.H. Krogh. Computational Techniques for Hybrid System Verification. *IEEE Trans. on Automatic Control* **48**, 64-75, 2003.
7. S. A. Cameron and R. K. Culley. Determining the Minimum Translational Distance between Two Convex Polyhedra. *Proceedings of International Conference on Robotics and Automation* **48**, pp 591–596, 1986.
8. T. Dang. Verification and Synthesis of Hybrid Systems. PhD Thesis, INPG, 2000.
9. G. Frehse. PHAVER: Algorithmic Verification of Hybrid Systems past HYTECH. *International Journal on Software Tools for Technology Transfer* (STTT), Volume 10, Number 3, June, 2008.
10. A. Girard. Reachability of Uncertain Linear Systems using Zonotopes, *HSCC'05*, LNCS 3414, 291-305, 2005.
11. C. Le Guernic and A. Girard. Reachability Analysis of Hybrid Systems Using Support Functions. *CAV 2009*, LNCS 5643, pp 540-554, Springer, 2009.
12. A. Girard, C. Le Guernic and O. Maler. Efficient Computation of Reachable Sets of Linear Time-invariant Systems with Inputs, *HSCC'06*, LNCS 3927, 257-271 2006.
13. T.A. Henzinger, P.-H Ho and H. Wong-Toi. HYTECH: A model checker for hybrid systems. *Software Tools for Technology Transfer*, **1(1-2)**, 110-122, (1997).
14. R.E. Kalman, P.L. Falb, and M.A. Arbib. Topics in Mathematical System Theory. McGraw-Hill Book Company, 1968.
15. E.K. Kostoukova. State Estimation for dynamic systems via parallelotopes: Optimization and Parallel Computations. *Optimization Methods and Software* **9**, pp 269–306, 1999.
16. M. Kvasnica, P. Grieder, M. Baotic, and M. Morari. Multi-Parametric Toolbox (MPT). *HSCC'04*, LNCS 2993, pp 448-462, Springer, 2004.
17. A. Kurzhanski and P. Varaiya. Ellipsoidal techniques for reachability analysis. *Hybrid Systems: Computation and Control*, LNCS 1790, B. Krogh and N. Lynch, Eds, Springer-Verlag, pp. 202–214, 2000.
18. A. Kurzhanskiy and P. Varaiya. Ellipsoidal Techniques for Reachability Analysis of Discrete-time Linear Systems. *IEEE Trans. Automatic Control* **52**, 26-38, 2007.
19. K. Larsen, P. Pettersson, and W. Yi. UPPAAL in a nutshell. *Software Tools for Technology Transfert*, **1(1-2)**, 134-152, 1997.
20. M.C. Lin and D. Manocha. Collision and proximity queries. *Handbook of Discrete and Computational Geometry*, 2003.
21. G. Pappas, G. Lafferriere, and S. Yovine. A new class of decidable hybrid systems. *Hybrid Systems: Computation and Control*, LNCS 1569, F. Vaandrager and J. van Schuppen, Eds. Springer-Verlag, pp. 29–31, 1999.
22. S. Sankaranarayanan, T. Dang, and F. Ivancic. Symbolic model checking of hybrid systems using template polyhedra. In *TACAS'08 - Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2008.
23. O. Stursberg and B. H. Krogh. Efficient Representation and Computation of Reachable Sets for Hybrid Systems. *HSCC 2003*, LNCS, pp 482-497, Springer, 2003.
24. P. Varaiya. Reach Set computation using Optimal Control. *KIT Workshop, Verimag, Grenoble*, 377-383, 1998.
25. S. Yovine. KRONOS: A verification tool for real-time systems. *Software Tools for Technology Transfer*, **1(1-2)**, 123-133, 1997.