Efficient Bounded Reachability Computation for Rectangular Automata

Xin Chen[1], Erika Ábrahám[1], and Goran Frehse[2]

[1] RWTH Aachen University, Germany
[2] Université Grenoble 1 Joseph Fourier - Verimag, France

Abstract. We present a new approach to compute the reachable set with a bounded number of jumps for a rectangular automaton. The reachable set under a flow transition is computed as a polyhedron which is represented by a conjunction of finitely many linear constraints. If the bound is viewed as a constant, the computation time is polynomial in the number of variables.

1 Introduction

Hybrid systems are systems equipped with both continuous dynamics and discrete behavior. A popular modeling formalism for hybrid systems are *hybrid automata*. In this paper, we consider a special class of hybrid automata, called *rectangular automata* [1]. The main restriction is that the derivatives, invariants and guards are defined by lower and upper bounds in each dimension, forming *rectangles* or *boxes* in the value domain. Rectangular automata can be used to model not only simple timed systems but also asymptotically approximate hybrid systems with nonlinear behaviors [2–5].

Since hybrid automata often model safety-critical systems, their *reachability analysis* builds an active research area. The reachability problem is decidable only for *initialized* rectangular automata [1], which can be reduced to timed automata [6]. The main merit of rectangular automata is that the reachable set under a flow is always a (convex) *polyhedron*. It means that the reachable set in a bounded number of jumps can be exactly computed as a set of polyhedra, unlike for general hybrid automata which need approximative methods such as [7–9]. In the past, some geometric methods are proposed for exactly or approximately computing the reachable sets in a bounded number of jumps (see, e.g., [4, 3]). There are also tools like HyTech [10] and PHAVer[11] which can compute bounded reachability for rectangular automata in a geometric way.

However, nearly all of the proposed methods compute the exact reachable set under a flow based on the vertices of the initial set and the derivative rectangle. Since a d-dimensional rectangle has 2^d many vertices, those methods are not able to handle high-dimensional cases. In [3], an approximative method is proposed to over-approximate the reachable set by polyhedra which are represented by conjunctions of linear constraints. Since only $2d$ linear constraints are needed to define a d-dimensional rectangle, the computation time of the method is polynomial in d. However, the accuracy degenerates dramatically when d increases.

In this paper, we compute the reachable set as polyhedra which are represented by finite linear constraint sets [12], where we need only $2d$ linear constraints to define a d-dimensional rectangle. We show that when the number of jumps is bounded by a constant, the computational complexity of our approach is polynomial in d. We also include the cases that some of the rectangles in the definition of a rectangular automaton are not full-dimensional.

The paper is organized as follows. After introducing some basic definitions in Section 2, we describe our efficient approach for computing the bounded reachable set in Section 3. In Section 4, we compare our approach and PHAVer based on a scalable example. Missing proofs can be found in [13].

2 Preliminaries

2.1 Polyhedra and their computation

For a point (or column vector) $v \in \mathbb{R}^d$ in the d-dimensional Euclidean space $\mathbb{R}^d$ we use $v[i]$ to denote its ith component, $1 \leq i \leq d$, and v^T for the row vector being its transpose.

In the following we call linear inequalities $c^T x \leq z$ for some $c \in \mathbb{R}^d$, $z \in \mathbb{R}$ and x a variable, also *constraints*. Given a finite set $\mathcal{L}$ of linear equalities and linear inequalities, we write $S : \mathcal{L}$ for $S = \{x \in \mathbb{R}^d \mid x \textit{ satisfies } \bigwedge_{L \in \mathcal{L}} L\}$, and also write $S : L$ instead of $S : \{L\}$. We say that $L \in \mathcal{L}$ is *redundant* in $\mathcal{L}$ if $S : \mathcal{L} = S' : \mathcal{L} \backslash \{L\}$. Redundant (in)equalities can be detected using linear programming [14].

A finite set $\{v_1, \ldots, v_{d'}\} \subseteq \mathbb{R}^d$ of linearly independent vectors span an $(d'-1)$-dimensional *affine subspace* Π of $\mathbb{R}^d$ by the affine combinations of $v_1, \ldots, v_{d'}$:

$$\Pi = \{ \sum_{1 \leq i \leq d'} \lambda_i v_i \mid \sum_{1 \leq i \leq n} \lambda_i = 1, \lambda_i \in \mathbb{R}\}$$

The *affine hull* $\mathit{aff}(S)$ of a set $S \subseteq \mathbb{R}^d$ is the smallest affine subspace $\Pi \subseteq \mathbb{R}^d$ containing S, and we have that $\mathit{dim}(S) = \mathit{dim}(\mathit{aff}(S))$. We call a subset of a vector space *full-dimensional* if its affine hull is the whole space.

A ($(d-1)$-dimensional) *hyperplane* in $\mathbb{R}^d$ is a $(d-1)$-dimensional affine subspace of $\mathbb{R}^d$. Each hyperplane H can be defined as $H : c^T x = z$ for some $c \in \mathbb{R}^d$ and $z \in \mathbb{R}$. For $d' < d-1$, a d'-dimensional affine subspace H' of $\mathbb{R}^d$ is called a *lower- or d'-dimensional hyperplane* and can be defined as an intersection of $d-d'$ many hyperplanes (see [12]), i.e., as $H' : \bigwedge_{1 \leq i \leq d-d'} c_i^T x = z_i$. Since every linear equation $c^T x = z$ can be expressed by $c^T x \leq z \wedge -c^T x \leq -z$, for $d'' \leq d$, a d''-dimensional hyperplane can be defined by a set of $2(d-d'')$ constraints.

A (d-dimensional) *halfspace* S in $\mathbb{R}^d$ is a d-dimensional set $S : c^T x \leq z$ for some $c \in \mathbb{R}^d$ and $z \in \mathbb{R}$. For $d' < d$, a d'-dimensional set $S' \subseteq \mathbb{R}^d$ is a *lower- or d'-dimensional halfspace* if it is the intersection of a (d-dimensional) halfspace S

and a d'-dimensional hyperplane $H' \not\subseteq S$. Note that for $d'' \leq d$, a d''-dimensional halfspace can be defined by a set of $2(d-d'')+1$ constraints.

Given a constraint $c^T x \leq z$, its *corresponding equation* is $c^T x = z$. The *corresponding hyperplane* of a d'-dimensional halfspace S with $d' \leq d$ is the $(d'-1)$-dimensional hyperplane defined by the set of the corresponding equations of the constraints that define S.

For a finite set $\mathcal{L}$ of constraints we call $P : \mathcal{L}$ a *polyhedron*. Polyhedra can also be understood as the intersection of finitely many halfspaces. *Polytopes* are bounded polyhedra.

A constraint $c^T x \leq z$ is *valid* for a polyhedron P if all $x \in P$ satisfy it. For $c^T x \leq z$ valid for P and for $H_F : c^T x = z$, the set $F : P \cap H_F$ is a *face* of P. If $F \neq \emptyset$ then we call H_F a *support hyperplane* of P, and the vectors λc for $\lambda > 0$ are the *normal vectors* of H_F. The hyperplane $H : c^T x = z$ is a support hyperplane of a polyhedron P if and only if for the *support function* $\rho_P : \mathbb{R}^d \to \mathbb{R} \cup \{\infty\}$, $\rho_P(v) = \sup v^T x \ s.t.\ x \in P$, we have that $\rho_P(c) = z$. We call a face F of a polyhedron P *facet* if $dim(F) = dim(P)-1$, and *vertex* if $dim(F) = 0$. For d'-dimensional faces we simply write *d'-faces*. We use $NF(P)$ to denote the number of P's facets. Given a face F of P, the *outer normals* of F are the vectors $v_F \in \mathbb{R}^d$ such that $\rho_P(v_F) = \sup v_F^T x$ for any $x \in F$. We also define $\mathcal{N}(F, P)$ as the set of the outer normals of F in P.

For a d'-dimensional polyhedron $P : \mathcal{L}_P$, every facet F_P of P can be *determined* by some $\mathcal{L}_{F_P} \subseteq \mathcal{L}_P$, that is $\mathcal{L}_{F_P}$ defines a d'-dimensional halfspace which contains P and the corresponding hyperplane is the affine hull of F_P (see [12]).

Lemma 1. *If a constraint set $\mathcal{L}$ defines a d'-dimensional polyhedron $P \subseteq \mathbb{R}^d$ and there is no redundant constraint in $\mathcal{L}$, then the set $\mathcal{L}$ contains $NF(P)+2(d-d')$ constraints.*

Proof. We need a set $\mathcal{L}'$ of $2(d-d')$ constraints to define $aff(P)$. For every facet F_P of P, we need a constraint L_{F_P} such that $\mathcal{L}' \cup \{L_{F_P}\}$ determines F_P.

For a polyhedron $P : \bigcup_{1 \leq i \leq n} \{c_i^T x \leq z_i\}$ and a scalar $\lambda \geq 0$, the *scaled polyhedron* λP can be computed by $\lambda P : \bigcup_{1 \leq i \leq n} \{c_i^T x \leq \lambda z_i\}$. The *conical hull* of P is the polyhedral cone $cone(P) = \bigcup_{\lambda \geq 0} \lambda P$. If the conical hull of P is d'-dimensional, then P is at least $(d'-1)$-dimensional (see [12]).

Example 1. Figure 1(a) shows a polyhedron $P : x_2 \leq 3 \wedge -x_1 \leq -1 \wedge x_1 - 2x_2 \leq -3$ with three irredundant constraints. The support hyperplanes H_1, H_2, H_3 intersect P at its facets. The fourth hyperplane H_4 is also a support hyperplane of P, but it only intersects P at a vertex, and the related constraint $-x_2 \leq -2$ is redundant. The conical hull of P is shown in Figure 1(b).

Given two polyhedra $P : \mathcal{L}_P$ and $Q : \mathcal{L}_Q$, their *intersection* $P \cap Q$ can be defined by the union of their constraints $P \cap Q : \mathcal{L}_P \cup \mathcal{L}_Q$. The *Minkowski sum* $P \oplus Q$ of P and Q is defined by $P \oplus Q = \{p+q \mid p \in P, q \in Q\}$. It is still a polyhedron, as illustrated in Figure 2. We have the following important theorem for the faces of $P \oplus Q$.

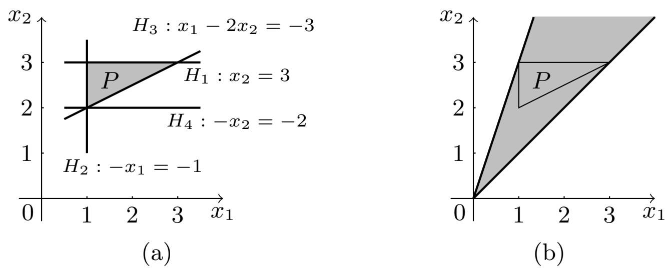


Fig. 1. A 2-dimensional polytope and its conical hull

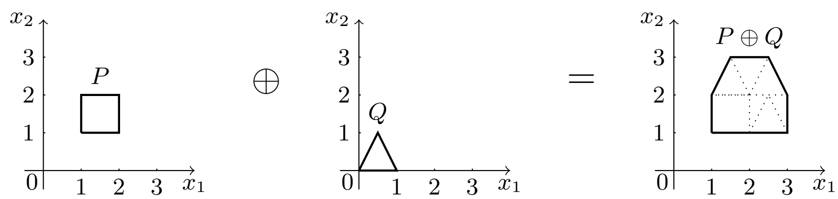


Fig. 2. An example of $P \oplus Q$

Theorem 1 (**[15, 16]**). *For any polytopes P and Q, each face F of $P \oplus Q$ can be decomposed by $F = F_P \oplus F_Q$ for some faces F_P and F_Q of P and Q respectively. Moreover, this decomposition is unique.*

2.2 Rectangular automata

A *box* $B \subseteq \mathbb{R}^d$ is an axis-aligned rectangle which can be defined by a set of constraints of the form $x \leq \overline{a}$ or $-x \leq -\underline{a}$ where x is a variable, and $\overline{a}, \underline{a}$ are rationals. A box $B : \mathcal{L}_B$ is *bounded* if for every variable x there are constraints $x \leq \overline{a}$ and $-x \leq -\underline{a}$ in $\mathcal{L}_B$ for some rationals $\overline{a}, \underline{a}$, otherwise B is *unbounded.* Let $\mathcal{B}^d$ be the set of all boxes in $\mathbb{R}^d$.

Rectangular automata [1] are a special class of hybrid automata [17].

Definition 1. *A d-dimensional* rectangular automaton *is a tuple* $\mathcal{A} = (Loc, Var, Flow, Jump, Inv, Init, Guard, ResetVar, Reset)$ *where*

- *Loc is a finite set of locations, also called discrete states.*
- *$Var = \{x_1, \ldots, x_d\}$ is a set of d ordered real-valued variables. We denote the variables as a column vector $x = (x_1, \ldots, x_d)^T$.*
- *$Flow : Loc \to \mathcal{B}^d$ assigns each location a flow condition which is a box in $\mathbb{R}^d$.*
- *$Jump : Loc \times Loc$ is a set of jumps (or discrete transitions).*
- *$Inv : Loc \to \mathcal{B}^d$ maps to each location an invariant which is a bounded box.*
- *$Init : Loc \to \mathcal{B}^d$ maps to each location a bounded box as initial variable values.*
- *$Guard : Jump \to \mathcal{B}^d$ assigns to each jump a guard which is a box.*
- *$ResetVar : Jump \to 2^{Var}$ assigns to each jump a set of reset variables.*

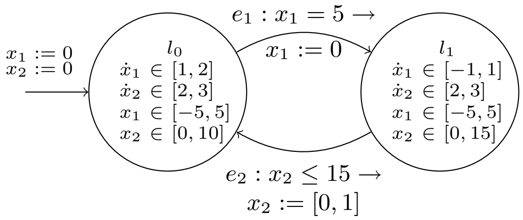


Fig. 3. A rectangular hybrid automaton $\mathcal{A}$

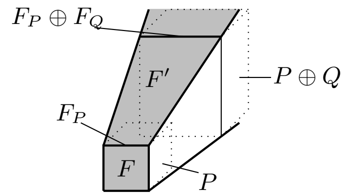


Fig. 4. A 3D example of R

– *Reset* : *Jump* $\rightarrow \mathcal{B}^d$ *maps to each jump a reset box such that for all* $e \in$ *Jump and* $x_i \in$ *ResetVar*(e) *the box Reset*(e) *is bounded in dimension* i.

Example 2. Figure 3 shows an example rectangular automaton. For brevity, we specify boxes by their defining intervals. The location set is $Loc = \{l_0, l_1\}$. The initial states are $Init(l_0) = [0,0] \times [0,0]$ and $Init(l_1) = \emptyset$. The flows are defined by $Flow(l_0) = [1,2] \times [2,3]$ and $Flow(l_1) = [-1,1] \times [2,3]$, and the invariants by $Inv(l_0) = [-5,5] \times [0,10]$ and $Inv(l_1) = [-5,5] \times [0,15]$. There are two jumps $Jump = \{e_1, e_2\}$ with $e_1 = (l_0, l_1)$ and $e_2 = (l_1, l_0)$. The guards are $Guard(e_1) = [5,5] \times (-\infty, +\infty)$ and $Guard(e_2) = (-\infty, +\infty) \times (-\infty, 15]$, the reset variable sets $ResetVar(e_1) = \{x_1\}$ and $ResetVar(e_2) = \{x_2\}$, and the reset boxes $Reset(e_1) = [0,0] \times (-\infty, +\infty)$ and $Reset(e_2) = (-\infty, +\infty) \times [0,1]$.

A *configuration* of $\mathcal{A}$ is a pair (l, u) such that $l \in Loc$ is a location and $u \in Inv(l)$ a vector assigning the value $u[i]$ to x_i for $i = 1, \ldots, d$. There are two kinds of transitions between configurations:

– *Flow:* A transition $(l, u) \xrightarrow{t} (l, u')$ where $t \geq 0$, such that there exists $b \in Flow(l)$ such that $u' = u + tb$ and for all $0 \leq t' \leq t$ we have $u + t'b \in Inv(l)$.
– *Jump:* A transition $(l, u) \xrightarrow{e} (l', u')$ such that $e = (l, l') \in Jump$, $u \in Guard(e)$, $u' \in Inv(l') \cap Reset(e)$, and $u[i] = u'[i]$ for all $x_i \in Var \backslash ResetVar(e)$.

An *execution* of $\mathcal{A}$ is a sequence $(l_0, u_0) \xrightarrow{\alpha_0} (l_1, u_1) \xrightarrow{\alpha_1} \cdots$ where $\xrightarrow{\alpha_i}$ is either a flow or a jump for all i. A configuration is *reachable* if it can be visited by some execution. The *reachability computation* is the task to compute the set of the reachable configurations. In this paper we consider bounded reachability with the number of jumps in the considered executions bounded by a positive integer.

3 A New Approach for Reachability Computation

In this section, we present a new approach to compute the reachable set for a rectangular automaton where the number of jumps is bounded.

3.1 Facets of the reachable set under flow transitions

For a location l of a rectangular automaton with $\mathit{Flow}(l) = Q$ and $\mathit{Init}(l) = P$, the states reachable from P via the flow can be computed in a geometric way:

$$R_l(P) = (P \oplus \mathit{cone}(Q)) \cap \mathit{Inv}(l). \tag{1}$$

As already mentioned, previously proposed methods compute $R_l(P)$ by considering the evolutions of all vertices of P under the flow condition Q. That means, also all vertices of Q must be considered. Since Q is a bounded box, it has 2^d vertices which make the computation intractable for large d. We present an approach to compute $R_l(P)$ exactly based on three constraint sets which define P, Q and $\mathit{Inv}(l)$ respectively. We show that if we are able to compute a constraint set that defines $P \oplus Q$ in PTIME, then a constraint set which defines $R_l(P)$ can also be computed in PTIME.

We firstly investigate the faces of the set $R = P \oplus \mathit{cone}(Q)$ in the general case that P and Q are polytopes in $\mathbb{R}^d$. From the following two lemmata we derive that the number of R's facets is bounded by $(n_P + n_{P\oplus Q})$ where n_P is the number of the facets of P and $n_{P\oplus Q}$ is the number of the faces from dimension $(\mathit{dim}(R){-}2)$ to $(\mathit{dim}(R){-}1)$ in $P \oplus Q$.

Lemma 2. *Given a polytope $Q \subseteq \mathbb{R}^d$ and a positive integer d', a d'-face $F_{\mathit{cone}(Q)}$ of the polyhedron $\mathit{cone}(Q)$ can be expressed by $\mathit{cone}(F_Q)$ where F_Q is a nonempty face of Q and it is at least $(d'{-}1)$-dimensional.*

Proof. The polyhedron $\mathit{cone}(Q)$ can be expressed by $\mathit{cone}(V_Q)$ where V_Q is the vertex set of Q. Then a nonempty face $F_{\mathit{cone}(Q)}$ of $\mathit{cone}(Q)$ can be expressed by $\mathit{cone}(V'_Q)$ where $V'_Q \subseteq V_Q$ is nonempty. Assume S is the halfspace whose corresponding hyperplane is $H = \mathit{aff}(F_{\mathit{cone}(Q)})$. Since $\mathit{cone}(Q) \subseteq S$, we also have that $Q \subseteq S$, moreover, we can infer that H is a support hyperplane of Q and $F_Q = H \cap Q$ is a nonempty face of Q whose vertex set is V'_Q. Therefore, the face $F_{\mathit{cone}(Q)}$ can be expressed by $\mathit{cone}(F_Q)$. From the definition of conical hull, if $F_{\mathit{cone}(Q)}$ is d'-dimensional then F_Q is at least $(d'-1)$-dimensional. □

Lemma 3. *Given two polytopes $P, Q \subseteq \mathbb{R}^d$, any d'-face F_R of the polytope $R = P \oplus \mathit{cone}(Q)$ is either a d'-face of P, or the decomposition $F_R = \bigcup_{\lambda \geq 0}(F_P \oplus \lambda F_Q)$ where F_P, F_Q are some nonempty faces of P, Q respectively and $F_P \oplus F_Q$ is a face of $P \oplus Q$ which is at least $(d'{-}1)$-dimensional.*

Proof. We have two cases for a face F_R of R, (1) F_R is a face of P; (2) F_R can be expressed by $F_P \oplus \mathit{cone}(F_Q)$ where F_Q is a *nonempty* face of Q (from Theorem 1 and Lemma 2). In the case (2), we rewrite R and F_R by

$$R = P \oplus \mathit{cone}(Q) = \bigcup_{\lambda \geq 0}(P \oplus \lambda Q) \quad \text{and} \quad F_R = F_P \oplus \mathit{cone}(F_Q) = \bigcup_{\lambda \geq 0}(F_P \oplus \lambda F_Q)$$

Since F_R is a face of R, i.e., it is on the boundary of R, we infer that for all $\lambda \geq 0$ the set $F_P \oplus \lambda F_Q$ is a face of $P \oplus \lambda Q$. Thus $F_P \oplus F_Q$ is a face of $P \oplus Q$. Since F_R is d'-dimensional, the set $F_P \oplus F_Q$ is at least $(d'-1)$-dimensional. □

Example 3. In Figure 4 on page 5, the set F is a facet of both P and R. In contrast, the facet F' can be expressed by $\bigcup_{\lambda \geq 0}(F_P \oplus \lambda F_Q)$.

The facets of R can be found by enumerating all the facets of P, and all the faces from dimension $(dim(R){-}2)$ to $(dim(R){-}1)$ in $P \oplus Q$.

Lemma 4. *Let $P : \mathcal{L}_P$ and $P \oplus Q : \mathcal{L}_{P \oplus Q}$ be some polytopes with $\mathcal{L}_{P \oplus Q} = \{g_j^T x \leq h_j \mid 1 \leq j \leq m\}$ irredundant. We define the constraint set $\mathcal{L} = \bigcup_{1 \leq i < j \leq m} \mathcal{L}_{i,j}$ such that for each $1 \leq i < j \leq m$, $\mathcal{L}_{i,j} = \{L_{i,j}\}$ if the intersection of $H_i : g_i^T x = h_i$, $H_j : g_j^T x = h_j$ and $P \oplus Q$ is nonempty, and $L_{i,j}$ is a constraint whose corresponding hyperplane $H_{i,j}$ satisfies (1) $H_{i,j}$ is a support hyperplane of P, (2) $H_{i,j}$ is a support hyperplane of $P \oplus Q$ and (3) $H_i \cap H_j \subseteq H_{i,j}$. Otherwise $\mathcal{L}_{i,j} = \emptyset$.*

Suppose that $\mathcal{L}'$ is the set of all constraints in $\mathcal{L}_P$ and $\mathcal{L}_{P \oplus Q}$ that are valid for R. Then the polytope R can be defined by $\mathcal{L} \cup \mathcal{L}'$.

Note that $L_{i,j}$ is not unique for each $1 \leq i < j \leq m$, but we only need one of them. Intuitively, for any facet F_R of R, if F_R is also a facet of P then it can be determined by a subset $\mathcal{L}'_P$ of $\mathcal{L}_P$. Since the constraints in $\mathcal{L}'_P$ are also valid for R, we also have that $\mathcal{L}'_P \subseteq \mathcal{L}'$. Otherwise $F_R = \bigcup_{\lambda \geq 0}(F_P \oplus \lambda F_Q)$ for some nonempty faces F_P, F_Q of P, Q respectively. There are two cases, (a) if $F_P \oplus F_Q$ is $(dim(R){-}1)$-dimensional, then F_R can be determined by a subset $\mathcal{L}'_{P \oplus Q}$ of $\mathcal{L}_{P \oplus Q}$, it is also included by $\mathcal{L}'$; (b) if $F_P \oplus F_Q$ is $(dim(R){-}2)$-dimensional, the facet F_R is determined by a subset of $\mathcal{L}$. Hence, $\mathcal{L} \cup \mathcal{L}'$ defines R.

3.2 Compute the reachable set under flow transitions

In order to compute the constraint set that defines R, we need to find the hyperplanes $H_{i,j}$ stated in Lemma 4. We determine the $H_{i,j} : c^T x = z$ by solving a feasibility problem for the normal vector $c \in \mathbb{R}^d$ and the value $z \in \mathbb{R}$ as follows. Assume $dim(R) = d_R$, $P : \mathcal{L}_P$ and $P \oplus Q$ is defined by the irredundant set $\mathcal{L}_{P \oplus Q} = \{g_j^T x \leq h_j \mid 1 \leq j \leq m\}$. Firstly, we check if the set $H_i \cap H_j \cap (P \oplus Q)$ with $H_i : g_i^T x = h_i$ and $H_j : g_j^T x = h_j$ is nonempty by solving the following linear program:

$$\text{Find } x_I \in \mathbb{R}^d \quad s.t. \quad g_i^T x_I = h_i \wedge g_j^T x_I = h_j \wedge x_I \in P \oplus Q.$$

If such an x_I is found, then the intersection is nonempty, and there must be a $(d_R{-}2)$-face $F_{P \oplus Q}$ of $P \oplus Q$ contained in it since $\mathcal{L}_{P \oplus Q}$ is irredundant. We require that $H_{i,j}$ is a support hyperplane of $P \oplus Q$ and contains $F_{P \oplus Q}$. This can be ensured by finding c in the set $C_{i,j} = \{\alpha g_i + \beta g_j \mid \alpha, \beta \geq 0, \alpha + \beta > 0\}$ and demanding $c^T x_I = z$. An example is shown in Figure 5.

We also require that $H_{i,j}$ is a support hyperplane of P. This can be guaranteed by demanding $\rho_P(c) = z$ and $c = \alpha g_i + \beta g_j$. In order to replace $\rho_P(\alpha g_i + \beta g_j)$ by $\alpha \rho_P(g_i) + \beta \rho_P(g_j)$, we need to ensure their equivalence. This can be done by finding at least one point $p \in P$ such that $g_i^T p = \rho_P(g_i)$ and $g_j^T p = \rho_P(g_j)$. Since the $(d_R{-}2)$-face $F_{P \oplus Q}$ is contained in $H_i \cap H_j$, we have that $g_i^T x = \rho_{P \oplus Q}(g_i)$

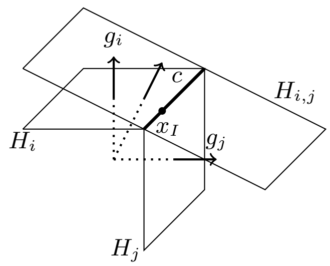


Fig. 5. A 3-dimensional example of the vector c

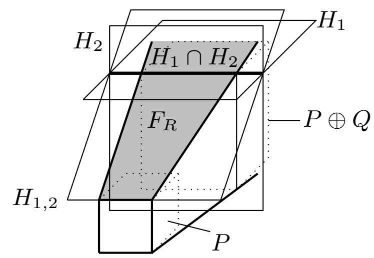


Fig. 6. An example of $H_{i,j}$

and $g_j^T x = \rho_{P\oplus Q}(g_j)$ for all $x \in F_{P\oplus Q}$. From Theorem 1, $F_{P\oplus Q}$ can be decomposed by $F_P \oplus F_Q$ for some faces F_P, F_Q of P, Q respectively, and we can infer that for all $x \in F_P$ it holds that $g_i^T x = \rho_P(g_i)$ and $g_j^T x = \rho_P(g_j)$. Hence we can replace $\rho_P(\alpha g_i + \beta g_j)$ by $\alpha\rho_P(g_i) + \beta\rho_P(g_j)$.

Then the vector c can be computed by solving the following problem:

$$\text{Find } c \in \mathbb{R}^d \; s.t. \begin{cases} c = \alpha g_i + \beta g_j \;\wedge\; \alpha+\beta > 0 \wedge \alpha \geq 0 \wedge \beta \geq 0 \\ c^T x_I = \alpha\rho_P(g_i) + \beta\rho_P(g_j) \end{cases} \tag{2}$$

We set $z = \rho_P(c)$. An example of $H_{i,j}$ is given in Figure 6.

We also need to find the valid constraints for R in $\mathcal{L}_P$ and $\mathcal{L}_{P\oplus Q}$. Given a constraint $L : c^T x \leq z$, L is valid for R if and only if $\rho_R(c) \leq z$. Since

$$\rho_R(c) = \rho_P(c) + \lambda\rho_Q(c) = \sup c^T x + \lambda \sup c^T y \quad s.t. \quad x \in P, y \in Q, \lambda \geq 0,$$

we compute $\rho_P(c)$ and $\rho_Q(c)$ by linear programming. If $\rho_Q(c) \leq 0$ then $\rho_R(c) = \rho_P(c)$, otherwise $\rho_R(c) = \infty$.

If we have the constraints for $P \oplus Q$ then Problem (2) is linear. Algorithm 1 shows the computation of the irredundant constraints of R. Finally, the polytope $\mathcal{L}_{R_l(X)}$ can be defined by the set $\mathcal{L}_R \cup \mathcal{L}_{Inv(l)}$ where $\mathcal{L}_{Inv(l)}$ defines $Inv(l)$.

3.3 Compute the reachable set after a jump

A jump $e = (l, l')$ of a rectangular automaton can update a variable by a value in an interval $[\underline{a}, \overline{a}]$. If the set of the reachable states in l is computed as $(l, R_l(X))$, then the set of states at which e is enabled can be computed by $(l, R_l(X) \cap Guard(e))$. Thus the reachable set after the jump e is $(l', R_e(R_l(X)))$ where

$$\begin{aligned} R_e(R_l(X)) = \{ & u' \in Inv(l') \cap Reset(e) \mid \exists u \in R_l(X) \cap Guard(e). \\ & \forall x_i \in Var \backslash ResetVar(e). u'[i] = u[i] \} \end{aligned} \tag{3}$$

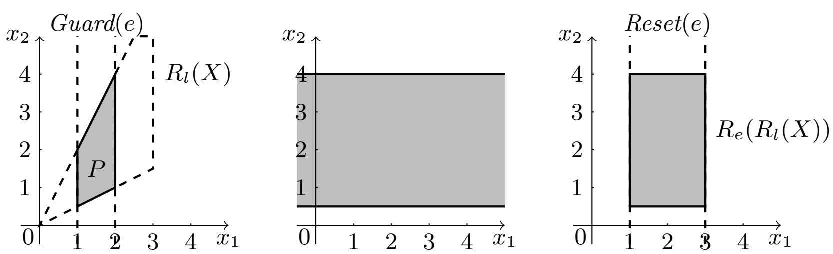
The set $R_e(R_l(X))$ can also be computed in a geometric way. The guard can be considered by defining $R_l(X) \cap Guard(e) : \mathcal{L}_{R_l(X)} \cup \mathcal{L}_{Guard(e)}$. The polytope $R_e(R_l(X))$ can be defined by $\mathcal{L}_e \cup \mathcal{L}_{Reset(e)}$ where $\mathcal{L}_e$ is the set of the constraints computed from $\mathcal{L}_{R_l(X)} \cup \mathcal{L}_{Guard(e)}$ by eliminating all reset variables by Fourier-Motzkin elimination [12], and $\mathcal{L}_{Reset(e)}$ defines the box $Reset(e)$.

Algorithm 1 Algorithm to compute R

Input: $P : \mathcal{L}_P$, $Q : \mathcal{L}_Q$

Output: An irredundant constraint set $\mathcal{L}_R$ of R

```
Compute an irredundant constraint set L_{P⊕Q} of P ⊕ Q;    L_R := ∅;
for all constraints c^T x ≤ z in L_P ∪ L_{P⊕Q} do
    if c^T x ≤ z is valid to R then
        Add the constraint c^T x ≤ z into L_R;
    end if
end for
for all constraints g_i^T x = h_i and g_j^T x = h_j in L_{P⊕Q} do
    Find a hyperplane H_{i,j} : c^T x = z by solving Problem (2);
    if H_{i,j} exists then
        Add the constraint c^T x ≤ z to L_R;
    end if
end for
Remove the redundant constraints from L_R;
return L_R
```



(a) The set P (b) After eliminating x_1 (c) The set $R_e(R_l(X))$

Fig. 7. A 2-dimensional example of resetting x_1 to $[1, 3]$

Example 4. We show an example in Figure 7, where $R_l(X) \cap Guard(e)$ is given by the polytope $P : -2x_1+x_2 \leq 0 \wedge x_1-2x_2 \leq 0 \wedge x_1 \leq 2 \wedge -x_1 \leq -1$. The reset box is $Reset(e) : x_1 \leq 3 \wedge -x_1 \leq -1$, and $Inv(l')$ is the box $[0, 5] \times [0, 5]$. Firstly, we compute the maximum and minimum value of the variable x_1, and we obtain $x_1 \leq 2$ and $-x_1 \leq -1$. By using the constraint $x_1 \leq 2$, we eliminate the variable x_1 from $-2x_1+x_2 \leq 0$ and obtain a new constraint $x_2 \leq 4$. Similarly, we use $-x_1 \leq -1$ to eliminate the variable x_1 from $x_1 - 2x_2 \leq 0$ and get $-x_2 \leq -0.5$. At last, the set $R_e(R_l(X))$ is the polytope

$$R_e(R_l(X)) : \ x_2 \leq 4 \wedge -x_2 \leq -0.5 \wedge x_1 \leq 3 \wedge -x_1 \leq -1$$

Algorithm 2 shows the computation of the reachable set after a jump. Although the Fourier-Motzkin elimination is double-exponential in general, in the next section we show that it is efficient on the reachable sets.

3.4 Complexity of the reachability computation

Algorithm 2 Algorithm to compute the constraints of $R_e(R_l(X))$

Input: The jump $e = (l, l')$, the constraints of $R_l(X) : \mathcal{L}$
Output: The constraints of $R_e(R_l(X))$
Compute the constraint set $\mathcal{L}_P$ of $P = R_l(X) \cap \mathit{Guard}(e)$; $\quad S \leftarrow \mathcal{L}_P$;
for all $x_i \in \mathit{ResetVar}(e)$ **do**
  Eliminate x_i from the constraints in S by Fourier-Motzkin elimination;
end for
return $S \cup \mathcal{L}_{Reset(e)}$

Algorithm 3 Reachability computation for a rectangular automaton

Input: A rectangular hybrid automaton $\mathcal{A}$
Output: The reachable set of $\mathcal{A}$
$R_{\mathcal{A}} \leftarrow \{(l, \mathit{Init}(l)) \mid l \in \mathit{Loc}\}$;
Define a queue Q with elements $(l, X) \in R_{\mathcal{A}}$;
while Q is not empty **do**
  Get (l, X) from Q; $\quad Y \leftarrow R_l(X)$; $\quad R_{\mathcal{A}} \leftarrow R_{\mathcal{A}} \cup \{(l, Y)\}$;
  for all $e = (l, l') \in \mathit{Jump}$ **do**
    $Z \leftarrow R_e(Y)$;
    if $(l', Z) \notin R_{\mathcal{A}}$ **then**
      Insert (l', Z) into Q; $\quad R_{\mathcal{A}} \leftarrow R_{\mathcal{A}} \cup \{(l', Z)\}$;
    end if
  end for
end while
return $R_{\mathcal{A}}$

The reachable set of a rectangular automaton $\mathcal{A}$ can be computed by Algorithm 3. Any reachable set $R_l(X)$ in Algorithm 3 is computed by a sequence

$$X_0 \rightarrow R_{l_0}(X_0) \rightarrow X_1 \rightarrow R_{l_1}(X_1) \rightarrow \cdots \rightarrow X_k \rightarrow R_{l_k}(X_k)$$

where $X_j = R_{e_j}(R_{l_{j-1}}(X_{j-1}))$ for $1 \leq j \leq k$, and $X_0 = \mathit{Init}(l_0)$. Although the termination of Algorithm 3 is not guaranteed, if we lay an upper bound $\overline{k}$ on k then it always stops. We prove that if k is viewed as a constant, then the computation is polynomial in the number of the variables of $\mathcal{A}$.

We prove it by showing that an irredundant constraint set of X_j can be computed from an irredundant constraint set of X_{j-1} in PTIME. Notice that this property is not possessed by any of the methods proposed in the past.

Lemma 5. *For $1 \leq j \leq k$, both $NF(X_j)$ and $NF(X_{j-1} \oplus B_{j-1})$ are polynomial in $NF(X_{j-1})$.*

Proof. By Lemma 1, the size of the irredundant constraint set of X_j is proportional to $NF(X_j)$, then we consider the facets of X_j. We define $G_j = \mathit{Inv}(l_j) \cap \mathit{Guard}(e_{j+1})$ and $B_j = \mathit{Flow}(l_j)$. If the whole space is $\mathbb{R}^d$, in order to maximize the number of X_j's facets, we assume B_i, G_i for $0 \leq i \leq j-1$ are full-dimensional boxes, and X_j is also full-dimensional. Since X_j can be expressed by

$$\bigcup_{a_{j-1}\leq\lambda_{j-1}\leq b_{j-1}} \cdots \bigcup_{a_0\leq\lambda_0\leq b_0} R_{e_j}((\cdots R_{e_1}((X_0\oplus\lambda_0 B_0)\cap G_0)\cdots\oplus\lambda_{j-1}B_{j-1})\cap G_{j-1})$$

a facet F_{X_j} of it can be uniquely expressed by

$$\bigcup_{a'_{j-1}\leq\lambda_{j-1}\leq b'_{j-1}} \cdots \bigcup_{a'_0\leq\lambda_0\leq b'_0} F(\lambda_0,\ldots,\lambda_{j-1}) \tag{4}$$

where $a_i \leq a'_i$ and $b'_i \leq b_i$ for $0 \leq i \leq j-1$, such that

(i) $F(\lambda_0,\ldots,\lambda_{j-1})$ is a face of the box $\Phi(\lambda_0,\ldots,\lambda_{j-1}) = R_{e_j}((\cdots R_{e_1}((X_0 \oplus \lambda_0 B_0)\cap G_0)\cdots\oplus\lambda_{j-1}B_{j-1})\cap G_{j-1})$ and there is no higher dimensional face of $\Phi(\lambda_0,\ldots,\lambda_{j-1})$ can be used to express F_{X_j};
(ii) if the maximum dimension of all those faces $F(\lambda_0,\ldots,\lambda_{j-1})$ is d' where $d-d'-1 \leq j$, then there are exactly $(d-d'-1)$ many λ_i where $0 \leq i \leq j-1$ such that these parameters help to determine $\mathcal{N}(F_{X_j}, X_j)$;
(iii) for any $0 \leq i \leq j-1$, if λ_i helps to determine $\mathcal{N}(F_{X_j}, X_j)$, then the box G_i could also help to determine $\mathcal{N}(F_{X_j}, X_j)$;
(iv) for any $0 \leq i \leq j-1$, any γ_i, γ'_i where

$$\begin{cases} a'_i < \gamma_i, \gamma'_i < b'_i, & \textit{if } a'_i < b'_i \\ \gamma_i = \gamma'_i = a'_i, & \textit{otherwise} \end{cases}$$

we have that $F(\gamma_0,\ldots,\gamma_{j-1}), F(\gamma'_0,\ldots,\gamma'_{j-1})$ have the maximum dimension among all the faces $F(\lambda_0,\ldots,\lambda_{j-1})$, and $\mathcal{N}(F(\gamma_0,\ldots,\gamma_{j-1}), \Phi(\gamma_0,\ldots,\gamma_{j-1}))$ $= \mathcal{N}(F(\gamma'_0,\ldots,\gamma'_{j-1}), \Phi(\gamma'_0,\ldots,\gamma'_{j-1}))$.

In brief, the above properties tell that $\mathcal{N}(F_{X_j}, X_j)$ depends on (a) the set $\mathcal{N}(F(\gamma_0,\ldots,\gamma_{j-1}), \Phi(\gamma_0,\ldots,\gamma_{j-1}))$ in the property (iv), i.e., the outer normals of a bounded box face (we call those faces related), (b) the $(d-d'-1)$ parameters in the property (ii), and (c) the dependence of $\mathcal{N}(F_{X_j}, X_j)$ and G_i for every $0 \leq i \leq j-1$ such that λ_i helps to determine $\mathcal{N}(F_{X_j}, X_j)$. Thereby if F_B is a d'-face of a bounded box, it has at most $2^{d-d'-1}\binom{j}{d-d'-1}$ related facets in X_j.

Given a dimension d' where $d-d'-1 \leq j$, as we said, if a d'-face F_B of a bounded box B is related to some facet F_{X_j} then there are exactly $(d-d'-1)$ many λ_i's help to determine the outer normals of F_{X_j}. Thus there are $(d-d'-1)$ steps to determine $\mathcal{N}(F_{X_j}, X_j)$. We define $\mathcal{P}_i$ as the set of the d'-faces in B which possibly have related facets in X_j after the ith step. Obviously, $\mathcal{P}_0$ contains all the d'-faces in B. In every $(i+1)$th step, at least half of the faces in $\mathcal{P}_i$ lose the possibility to have related facets in X_j since X_i is a union of boxes and every box is centrally symmetric. Hence, there are at most

$$2^{-(d-d'-1)}\mathcal{F}^d_{d'} = 2^{-(d-d'-1)}\left(2^{d-d'}\binom{d}{d'}\right) = 2\binom{d}{d'}$$

d'-faces of B could have related facets in X_j, where $\mathcal{F}^d_{d'}$ is the number of the d'-faces in B. Therefore, there are at most $2^{d-d'}\binom{j}{d-d'-1}\binom{d}{d'}$ facets in X_j which are related to some d'-faces of B. By considering all $\max(d-j-1,0) \le d' \le d-1$, we can conclude that $NF(X_j)$ is polynomial in $NF(X_{j-1})$ for $j \ge 1$. Similarly, we can also prove that $NF(X_{j-1} \oplus B_{j-1})$ is polynomial in $NF(X_{j-1})$ for $j \ge 1$. $\square$

Now we give our method to compute X_j from X_{j-1}. The most expensive part in the computation is computing $X_{j-1} \oplus B_{j-1}$. We decompose B_{j-1} by $B_{j-1} = [\underline{a}_1, \overline{a}_1]_1 \oplus [\underline{a}_2, \overline{a}_2]_2 \oplus \cdots \oplus [\underline{a}_d, \overline{a}_d]_d$, such that for $1 \le i \le d$, $x[i] \le \overline{a}_i$, $-x[i] \le -\underline{a}_i$ are irredundant constraints for B_{j-1} and $[\underline{a}_i, \overline{a}_i]_i$ is a line segment (1-dimensional box) defined by the following constraint set:

$$\{x[i] \le \overline{a}_i, -x[i] \le -\underline{a}_i\} \cup \{x[i'] \le 0 \mid i' \ne i\} \cup \{-x[i'] \le 0 \mid i' \ne i\}$$

We denote the polytope resulting from adding the first m line segments onto X_{j-1} by X_{j-1}^m, then for all $1 \le m \le d$, $NF(X_{j-1}^m)$ is polynomial in $NF(X_{j-1})$. Since an irredundant constraint set for X_{j-1}^m can be computed in PTIME based on an irredundant constraint set of X_{j-1}^{m-1}, we conclude that an irredundant constraint set which defines $X_{j-1} \oplus B_{j-1}$ can be computed in a time polynomial in d if j is viewed as a constant.

Next we consider the complexity of the Fourier-Motzkin elimination on the set $R_{l_j}(X_j)$. Since $NF(X_{j+1})$ is polynomial in $NF(X_j)$, the polyhedron resulting from the elimination of each reset variable has a number of facets which is polynomial in $NF(X_j)$. Since eliminating one variable is PTIME, we conclude that the Fourier-Motzkin elimination on $R_{l_j}(X_j)$ is polynomial in d if j is viewed as a constant. If we use *interior point methods* [14] to solve linear programs then the bounded reachability computation is polynomial in d.

Theorem 2. *The computational complexity of $R_{l_j}(X_j)$ is polynomial in d if j is viewed as a constant.*

Theorem 3. *The computational complexity of the reachable set with a bounded number of jumps is polynomial in d if the bound is viewed as a constant.*

Unfortunately, the worst-case complexity is exponential in j. However, it only happens in extreme cases. The exact complexity of our approach mainly depends on the complexity of solving linear programs.

4 Experimental Results

We implemented our method in MATLAB using the CDD tool [18] for linear programming. We compared our implementation with PHAVer (embedded in SpaceEx [19]) on a scalable artificial example. Since there are rare high dimensional examples published, we design a scalable example which is given in Figure 8, where d is the (user-defined) dimension of the automaton and i denotes all the integers from 1 to d. The automaton $\mathcal{A}_d$ helps to generate reachable sets with large numbers of vertices and facets, and for each jump, nearly half of the variables are reset.

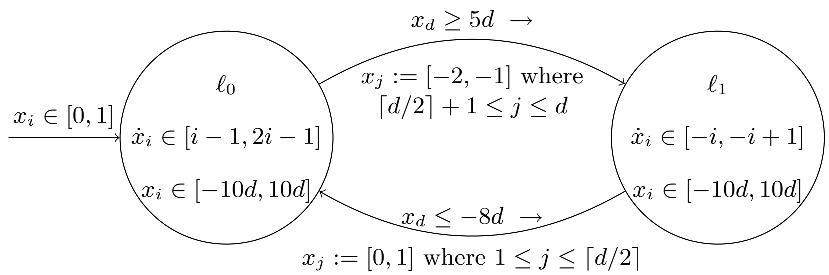


Fig. 8. Rectangular automaton $\mathcal{A}_d$

Dimension	MaxJmp	PHAVer		Our method				
		Memory	Time	Memory	Time	ToLP	LPs	Constraints
5	2	9.9	0.81	< 10	2.36	2.20	1837	81
6	2	48.1	21.69	< 10	4.96	4.68	3127	112
7	2	235.7	529.01	< 10	15.95	15.28	7214	163
8	2	n.a.	t.o.	< 10	27.42	26.48	10517	209
9	2	n.a.	t.o.	< 10	107.99	105.59	23639	287
10	2	n.a.	t.o.	< 10	218.66	215.45	32252	354
5	4	10.2	1.51	< 10	4.82	4.50	3734	167
6	4	51.1	35.52	< 10	11.25	10.64	7307	240
7	4	248.1	1191.64	< 10	32.93	31.60	16101	352
8	4	n.a.	t.o.	< 10	72.04	69.81	27375	466
9	4	n.a.	t.o.	< 10	240.51	235.61	64863	641
10	4	n.a.	t.o.	< 10	543.05	535.77	86633	816

Table 1. Experimental results. Time is in seconds, memory in MBs. "MaxJmp" is the bound on the number of jumps, "ToLP" is the total linear programming time, "LPs" is the number of linear programs solved (including the detection of redundant constraints), "Constraints" is the number of irredundant constraints computed, "n.a." means not available, "t.o." means that the running time was greater than one hour.

The experiments were run on a computer with a 2.8 GHz CPU and 4GB memory, the operating system is Linux. The experimental results are given by Table 1. Since MATLAB does not provide a build-in function to monitor the memory usage of its programs on Linux, the listed memory usage is the total memory usage minus MATLAB memory usage before the experiment. Our method can handle $\mathcal{A}_{10}$ efficiently, however PHAVer stops at $\mathcal{A}_7$. Our implementation is a prototype and the running times can even be improved by a C++ implementation and a faster LP solver.

5 Conclusion

We introduced our efficient approach for the bounded reachability computation of rectangular automata. However, the method of computing the reachable set under a flow transition can also be applied to linear hybrid automata. With some more effort this approach can also be adapted for the approximative analysis of hybrid systems with nonlinear behavior.

References

1. T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *J. Comput. Syst. Sci.*, 57(1):94–124, 1998.
2. T.A. Henzinger, P. Ho, and H. Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. *Automatic Control, IEEE Transactions on*, 43(4):540–554, apr 1998.
3. J. Preußig, S. Kowalewski, H. Wong-Toi, and T. A. Henzinger. An algorithm for the approximative analysis of rectangular automata. In *Proc. of FTRTFT'98*, volume 1486 of *LNCS*, pages 228–240. Springer, 1998.
4. H. Wong-Toi and J. Preußig. A procedure for reachability analysis of rectangular automata. In *Proc. of American Control Conference*, volume 3, pages 1674 –1678 vol.3, 2000.
5. L. Doyen, T. A. Henzinger, and J. Raskin. Automatic rectangular refinement of affine hybrid systems. In *Proc. of FORMATS'05*, volume 3829 of *LNCS*, pages 144–161. Springer, 2005.
6. R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.
7. A. Chutinan and B. H. Krogh. Computing polyhedral approximations to flow pipes for dynamic systems. In *Proc. of CDC'98*. IEEE Press, 1998.
8. O. Stursberg and B. H. Krogh. Efficient representation and computation of reachable sets for hybrid systems. In *Proc. of HSCC'03*, volume 2623 of *LNCS*, pages 482–497. Springer, 2003.
9. A. Girard. Reachability of uncertain linear systems using zonotopes. In *Proc. of HSCC'05*, volume 3414 of *LNCS*, pages 291–305. Springer, 2005.
10. T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. Hytech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, (1):110–122, 1997.
11. G. Frehse. Phaver: Algorithmic verification of hybrid systems past hytech. In *Proc. of HSCC'05*, volume 3414 of *LNCS*, pages 258–273. Springer, 2005.
12. G. M. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer, 1995.
13. X. Chen, E. Abraham, and G. Frehse. Efficient bounded reachability computation for rectangular automata. Technical report, RWTH Aachen University, 2011. `http://www-i2.informatik.rwth-aachen.de/i2/hybrid_research_pub0/`.
14. S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
15. K. Fukuda. From the zonotope construction to the minkowski addition of convex polytopes. *J. Symb. Comput.*, 38(4):1261–1272, 2004.
16. C. Weibel and K. Fukuda. Computing faces up to k dimensions of a minkowski sum of polytopes. In *Proc. of CCCG'05*, pages 256–259, 2005.
17. T. A. Henzinger. The theory of hybrid automata. In *Proc. of LICS'96*, pages 278–292, 1996.
18. K. Fukuda. cdd, cddplus and cddlib homepage. `http://www.ifor.math.ethz.ch/~fukuda/cdd_home/`.
19. G. Frehse, C. Le Guernic, A. Donzé, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. Spaceex: Scalable verification of hybrid systems. In *Proc. of CAV'11*, LNCS. Springer, 2011.

Appendix

Lemma 6. *Given a finite set of vectors $\{g_1, \ldots, g_m\} \subseteq \mathbb{R}^d$ which contains at least n linearly independent vectors where $n \leq d$ and $n \leq m$. For any set $V = \{g_{i,j} \mid 1 \leq i < j \leq n\}$ such that $g_{i,j}$ is a nontrivial linear combination of g_i and g_j, we have that there are at least $(n-1)$ linearly independent vectors in V.*

Proof. Without loss of generality, the first n vectors in $g_1, \ldots, g_m$ are assumed to be linearly independent, and we define $G_n = (G_{n-1}, A_n)$ where

$$G_{n-1} = (g_{1,2}, \ldots, g_{1,n-1}, g_{2,3}, g_{2,n-1}, \ldots, g_{n-2,n-1}) \text{ and } A_n = (g_{1,n}, \ldots, g_{n-1,n}).$$

We prove by induction on n that the rank of G_n is at least $(n-1)$, i.e., $r(G_n) \geq n-1$. The base case where $n = 1$ is trivial. When $n > 1$, by induction, there are at least $(n-2)$ linearly independent vectors in the set $V' = \{g_{i,j} \mid 1 \leq i < j \leq n-1\}$, in other words, $r(G_{n-1}) \geq n-2$. We investigate the matrix A_n. It can be written by

$$A_n = (\alpha_{1,n} g_1 + \beta_{1,n} g_n \ , \ \ldots \ , \ \alpha_{n-1,n} g_{n-1} + \beta_{n-1,n} g_n).$$

We assume that at least one of $\beta_{1,n}, \ldots, \beta_{n-1,n}$ is not zero, otherwise $r(A_n) = n-1$ and then $r(G_n) \geq n-1$. Without loss of generality, we assume $\beta_{1,n} \neq 0$. Since g_n is linearly independent of the vectors $g_1, \ldots, g_{n-1}$, we infer that the vector $\alpha_{1,n} g_1 + \beta_{1,n} g_n$ is linearly independent of the vectors $g_{i,j}$ for $1 \leq i < j \leq n-1$. Therefore, the rank of (G_{n-1}, A_n) is at least $(n-1)$. Since all the columns of G_n are vectors in V, we conclude that there are at least $(n-1)$ linearly independent vectors in V. □

Proof of Lemma 4

Proof. We show that for any facet F_R of R there is a subset of $\mathcal{L} \cup \mathcal{L}'$ which determines F_R. We assume that the whole space is $\mathbb{R}^d$ and $dim(R) = d_R$. For any facet F_R of R, we have two cases.

- **Case 1:** The set F_R is also a facet of P, i.e., $F_R = F_P$ where F_P is a facet of P. Then the constraint set $\mathcal{L}_{F_P} \subseteq \mathcal{L}_P$ which determines F_P can also be used to determine F_R, and therefore the constraints in $\mathcal{L}_{F_P}$ are also valid for R. So they are included by $\mathcal{L}'$.
- **Case 2:** The set F_R can be expressed by $F_R = \bigcup_{\lambda \geq 0} (F_P \oplus \lambda F_Q)$ where F_P and F_Q are some nonempty faces of P and Q respectively. By Lemma 3, $F_P \oplus F_Q$ is a face of $P \oplus Q$ and it is at least $(d_R - 2)$-dimensional.
  $(*)$ If $F_{P \oplus Q} = F_P \oplus F_Q$ is $(d_R - 1)$-dimensional, then $F_{P \oplus Q}$ and F_R are contained in the same $(d_R - 1)$-dimensional affine subspace of $\mathbb{R}^d$, i.e., $\mathit{aff}(F_P \oplus F_Q) = \mathit{aff}(F_R)$. Assume the set $\mathcal{L}_{F_P \oplus F_Q} \subseteq \mathcal{L}_{P \oplus Q}$ determines $F_{P \oplus Q}$. For any $L \in \mathcal{L}_{F_P \oplus F_Q}$, the halfspace $S_L : L$ contains $P \oplus Q$ and F_R is contained in its corresponding hyperplane H_L, thereafter H_L is a support hyperplane of R and L is a valid constraint for R. Hence we infer that $\mathcal{L}_{F_P \oplus F_Q}$ also determines F_R.

(∗∗) If $F_{P\oplus Q} = F_P \oplus F_Q$ is $(d_R - 2)$-dimensional, we assume it is determined by some $\mathcal{L}_{F_P \oplus F_Q} \subseteq \mathcal{L}_{P\oplus Q}$. We define

$$K = \{k \mid L_k : g_k^T x \leq h_k \text{ is contained in } \mathcal{L}_{F_P \oplus F_Q}\},$$

then *aff*$(F_{P\oplus Q})$ can be defined by the intersection $I_K = \bigcap_{k\in K} H_k$. Note that for any $i, j \in K \wedge i < j$, there always exists a hyperplane $H_{i,j}$ which intersects $P \oplus Q$ at $F_{P\oplus Q}$ and intersects P at F_P, since $H_i \cap H_j$ contains $F_{P\oplus Q}$. From the properties (1)-(3), we have that the unit normal vector of $H_{i,j}$ can be expressed by $\alpha g_i + \beta g_j$ where $\alpha, \beta \geq 0$, at least one of α, β is not zero, and g_i, g_j are the unit normal vectors of H_i, H_j respectively. We define U_K as the set of the unit normal vectors of $H_{i,j}$ for $i, j \in K \wedge i < j$. Since I_K is $(d_R - 2)$-dimensional, if V_K is the set of the unit normal vectors of H_k for $k \in K$, then there are at least $(d - d_R + 2)$ linearly independent vectors in V_K. By Lemma 6, there are at least $(d - d_R + 1)$ linearly independent vectors in U_K. Therefore, the intersection $\bigcap_{i,j\in K\wedge i<j} H_{i,j}$ is at most $(d_R - 1)$-dimensional. From the properties (1)-(3), if $H_{i,j} : c^T x = z$ exists for some $1 \leq i < j \leq m$, then

$$\rho_P(c) = \rho_{P\oplus Q}(c) = \rho_P(c) + \rho_Q(c) = z$$

thus $\rho_R(c) = z$, which means $H_{i,j}$ is a support hyperplane of R and $L_{i,j}$ is valid for R. For $i, j \in K \wedge i < j$ we can further derive that F_R is contained in $H_{i,j}$. Therefore we can conclude that the constraint set $\{L_{i,j} \mid i, j \in K \wedge i \neq j\}$ determines F_R.

The constraints for the facets in Case 1 and Case 2 (∗) can be found by searching the valid constraints in $\mathcal{L}_P$ and $\mathcal{L}_{P\oplus Q}$ for R. For the facets in Case 2 (∗∗), the constraint set can be found by enumerating the hyperplanes $H_{i,j}$ for $1 \leq i < j \leq m$. Therefore, the set $\mathcal{L} \cup \mathcal{L}'$ defines R. □