Reachability Analysis of Hybrid Systems Using Symbolic Orthogonal Projections*

Willem Hagemann

Carl von Ossietzky Universität Oldenburg,
Ammerländer Heerstraße 114–118, 26111 Oldenburg, Germany
willem.hagemann@informatik.uni-oldenburg.de

Abstract. This paper deals with reachability analysis of hybrid systems with continuous dynamics described by linear differential inclusions and arbitrary linear maps for discrete updates. The invariants, guards, and sets of reachable states are given as convex polyhedra. Our reachability algorithm is based on a novel representation class for convex polyhedra, the symbolic orthogonal projections (sops), on which various geometric operations, including convex hulls, Minkowski sums, linear maps, and intersections, can be performed efficiently and exactly. The capability to represent intersections of convex polyhedra exactly is superior to support function-based approaches like the LGG-algorithm (Le Guernic and Girard [21]).

Accompanied by some simple examples, we address the problem of the monotonic growth of the exact representation and propose a combination of our reachability algorithm with the LGG-algorithm. This results in an efficient method of better accuracy than the LGG-algorithm and its productive implementation in SpaceEx [13].

1 Introduction

Reachability analysis of hybrid systems has to deal with two problems: The first one is a systematic representation of the reachable states. Aside from nonconvex approaches like [4,5,22], the reachable states are usually represented as unions of convex sets, for which different representations, including polyhedra [7], template polyhedra [23], zonotopes [15,16], ellipsoids [19], and support functions [20], are used. The choice of the representation has a wide influence on the approximations of the underlying sets and on the efficiency of the operations required for the reachability analysis, e. g. zonotopes, ellipsoids, and support functions are challenging for intersections with guard sets [1,16]. The second problem is to tackle the dynamics of the system. Typical classes of admissible dynamics vary from constant derivatives [9,18], linear differential equations or inclusions [13,15,19] to nonlinear differential equations [23,6]. However, in order to approximate complexer dynamics, the classes should allow differential inclusions [2,3,12]. In turn,

* This work was partly supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS, http://www.avacs.org/).

A. Biere and R. Bloem (Eds.): CAV 2014, LNCS 8559, pp. 407–423, 2014.

the choice of the admissible dynamics has an impact on the required operations for the post image computation. Hence, both problems are highly related.

In this paper we focus on the reachability analysis of hybrid systems with continuous dynamics described by linear differential inclusions and arbitrary linear maps for discrete updates. The invariants, guards, and sets of reachable states are given as convex polyhedra, where we assume that the polyhedra are given as intersections of half-spaces (*$\mathcal{H}$-representation*) and not as convex hulls of vertices and rays (*$\mathcal{V}$-representation*). Our reachability algorithm is based on a novel representation class for convex polyhedra, the symbolic orthogonal projections (sops), on which various geometric operations, including convex hulls, Minkowski sums, linear maps, and intersections, can be performed efficiently and exactly. The capability to represent intersections of convex polyhedra exactly is superior to support function-based approaches.

Due to space limitations we omit the proofs. The interested reader will find the proofs and some additional materials in [17].

2 Template Polyhedra and Support Functions

In their 2009 article [21], Le Guernic and Girard proposed an algorithm for reachability analysis of hybrid systems based on the usage of support functions. This reachability algorithm, we call it the LGG-algorithm for short, as been implemented in the verification tool box SpaceEx [13]. The efficiency of the LGG-algorithm is achieved by a clever combination of support functions and template polyhedra. We briefly restate their representations of convex sets.

Template polyhedra are $\mathcal{H}$-polyhedra $\mathbf{P}(A_{\text{fix}}, \mathbf{a}) = \{\mathbf{x} \mid A_{\text{fix}}\mathbf{x} \leq \mathbf{a}\}$ where the template matrix A_{fix} is fixed a priori. For a – not necessarily convex – set $\mathbf{S} \subseteq \mathbb{R}^d$ and a direction $\mathbf{n} \in \mathbb{R}^d$ the value of the *support function* is defined as $h_{\mathbf{S}}(\mathbf{n}) = \sup_{\mathbf{x}\in\mathbf{S}} \mathbf{n}^T\mathbf{x}$. For an $\mathcal{H}$-polyhedron $\mathbf{P}(A, \mathbf{a})$ the value of the support function is given by the linear program "maximize $\mathbf{n}^T\mathbf{x}$ subject to $A\mathbf{x} \leq \mathbf{a}$". Support functions behaves nicely under most geometric operations; in detail, for any two compact convex sets $\mathbf{P}$, $\mathbf{Q}$ in $\mathbb{R}^d$, and any $(d \times d)$-matrix M the following equations are easily computable:

$$\begin{aligned} h_{M(\mathbf{P})}(\mathbf{n}) &= h_{\mathbf{P}}(M^T\mathbf{n}), && \text{(linear map)} \\ h_{\mathbf{P}+\mathbf{Q}}(\mathbf{n}) &= h_{\mathbf{P}}(\mathbf{n}) + h_{\mathbf{Q}}(\mathbf{n}), && \text{(Minkowski sum)} \\ h_{\text{conv}(\mathbf{P}\cup\mathbf{Q})} &= \max(h_{\mathbf{P}}(\mathbf{n}), h_{\mathbf{Q}}(\mathbf{n})), && \text{(closed convex hull)} \end{aligned}$$

while the intersection is not easily computable

$$h_{\mathbf{P}\cap\mathbf{Q}}(\mathbf{n}) = \inf_{\mathbf{m}\in\mathbb{R}^d} h_{\mathbf{P}}(\mathbf{n} - \mathbf{m}) + h_{\mathbf{Q}}(\mathbf{m}).$$

Given the template matrix A_{fix} and the support function $h_{\mathbf{S}}$ of some set $\mathbf{S}$, one easily obtains a closed convex over-approximation $\mathbf{P}(A_{\text{fix}}, \mathbf{a}_{\mathbf{S}})$ of $\mathbf{S}$: The ith coefficient of the vector $\mathbf{a}_{\mathbf{S}}$ is given by $h_{\mathbf{S}}(\mathbf{n}_i)$, where $\mathbf{n}_i^T$ is the ith row of the template matrix. We will use the notation $\mathbf{a}_{\mathbf{S}} = \mathbf{h}_{\mathbf{S}}(A_{\text{fix}})$ for such a row-wise computation of the vector $\mathbf{a}_{\mathbf{S}}$.

3 Symbolic Orthogonal Projections

We introduce a novel representation class for polyhedral sets which we call *symbolic orthogonal projections*, or *sops*, for short. Sops can be realized in any vector space $\mathbb{K}^d$ over an ordered field $\mathbb{K}$. Any sop $\mathbf{P} = \mathbf{P}(A, L, \mathbf{a}) \subseteq \mathbb{K}^d$, where A is an $(m \times d)$-matrix, L is an $(m \times k)$-matrix, and $\mathbf{a}$ is a column vector in $\mathbb{K}^m$, is the orthogonal projection of an $\mathcal{H}$-polyhedron $\mathbf{P}\left(\begin{pmatrix} A & L \end{pmatrix}, \mathbf{a}\right) \subseteq \mathbb{K}^{d+k}$ onto $\mathbb{K}^d$, where k is the number of columns in L, i. e.,

$$\mathbf{P} = \mathbf{P}(A, L, \mathbf{a}) = \left\{ \mathbf{x} \in \mathbb{K}^d \,\middle|\, \exists \mathbf{z} \in \mathbb{K}^k,\ A\mathbf{x} + L\mathbf{z} \leq \mathbf{a} \right\}.$$

Obviously, the sop $\mathbf{P}(A, L, \mathbf{a})$ is empty if and only if $\mathbf{P}\left(\begin{pmatrix} A & L \end{pmatrix}, \mathbf{a}\right)$ is empty, and any $\mathcal{H}$-polyhedron $\mathbf{P} = \mathbf{P}(A, \mathbf{a}) \in \mathbb{K}^d$ may be represented by the sop $\mathbf{P}(A, \emptyset, \mathbf{a})$, where $\emptyset$ denotes an empty matrix. Furthermore, for any sop $\mathbf{P}(A, L, \mathbf{a})$ in $\mathbb{K}^d$ and any given direction $\mathbf{n} \in \mathbb{K}^d$ the optimal value of the linear program "maximize $\mathbf{n}^T\mathbf{x}$ subject to $A\mathbf{x} + L\mathbf{z} \leq \mathbf{a}$" provides the value of the support function $h_{\mathbf{P}}(\mathbf{n})$. Hence, sops can easily be over-approximated by template polyhedra.

As a rather technical notion, we call a sop $\mathbf{P}(A, L, \mathbf{a})$ *complete* if there exists some $\mathbf{u} \geq 0$ with $\mathbf{0} = A^T\mathbf{u}$, $\mathbf{0} = L^T\mathbf{u}$, and $1 = \mathbf{a}^T\mathbf{u}$. Any sop can be completed by adding the redundant row $(\mathbf{0}^T, \mathbf{0}^T, 1)$ to its representation $(A, L, \mathbf{a})$.[1]

Convex Hull, Minkowski Sum, and Intersection. We show that symbolic orthogonal projections allow to efficiently represent closed convex hulls, Minkowski sums, and intersections of polyhedra. All these operations are realized as block matrices over the original matrices. The zero matrix is denoted by O.

Proposition 1. *Let* $\mathbf{P}_1 = \mathbf{P}(A_1, L_1, \mathbf{a}_1)$ *and* $\mathbf{P}_2 = \mathbf{P}(A_2, L_2, \mathbf{a}_2)$ *be two non-empty sops in* $\mathbb{K}^d$*. Then the following equations hold:*

$$\operatorname{conv}(\mathbf{P}_1 \cup \mathbf{P}_2) = \mathbf{P}\left(\begin{pmatrix} A_1 \\ \mathrm{O} \end{pmatrix}, \begin{pmatrix} A_1 & L_1 & \mathrm{O} & \mathbf{a}_1 \\ -A_2 & \mathrm{O} & L_2 & -\mathbf{a}_2 \end{pmatrix}, \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{0} \end{pmatrix}\right),$$

if $\mathbf{P}_1$ and $\mathbf{P}_2$ are complete;

$$\mathbf{P}_1 + \mathbf{P}_2 = \mathbf{P}\left(\begin{pmatrix} A_1 \\ \mathrm{O} \end{pmatrix}, \begin{pmatrix} A_1 & L_1 & \mathrm{O} \\ -A_2 & \mathrm{O} & L_2 \end{pmatrix}, \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{pmatrix}\right);$$

$$\mathbf{P}_1 \cap \mathbf{P}_2 = \mathbf{P}\left(\begin{pmatrix} A_1 \\ A_2 \end{pmatrix}, \begin{pmatrix} L_1 & \mathrm{O} \\ \mathrm{O} & L_2 \end{pmatrix}, \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{pmatrix}\right).$$

Linear Mappings. Any linear mapping ϕ is uniquely determined by its *transformation matrix* $M \in \mathbb{K}^{n \times m}$, i. e., $\phi(\mathbf{x}) = M\mathbf{x}$. We are interested in three types of linear mappings, where the $(n \times n)$-identity matrix is denoted by I_n: (i) *automorphisms*, having invertible transformation matrices; (ii) *orthogonal projections* proj_k, for $0 \leq k \leq d$, having $(k \times d)$-matrices of the form $(I_k \ \mathrm{O})$; and (iii) *elementary embeddings* embed_l, for $l \geq d$, having $(l \times d)$-matrices of the form $\binom{I_d}{\mathrm{O}}$.

[1] One can show that any sop $\mathbf{P}$ which represents a fully dimensional polytope in $\mathbb{K}^d$ with $d \geq 1$ is complete. For a geometric interpretation of completeness, see [17].

Proposition 2. *Every transformation matrix M can be written as the product $M = S^{-1}EPT^{-1}$, where S and T are invertible, E is the matrix of an elementary embedding, and P is the matrix of an orthogonal projection.*

Proposition 3. *Let $\mathbf{P}_1 = \mathbf{P}(A_1, L_1, \mathbf{a}_1)$ be a sop in $\mathbb{K}^d$, S an invertible $(d \times d)$-transformation matrix of the linear mapping ϕ, proj_k an orthogonal projection with $0 \le k \le d$, and embed_l an elementary embedding with $l \ge d$. Then*

$$\begin{aligned}
\phi(\mathbf{P}_1) &= \mathbf{P}\left(A_1 S^{-1}, L_1, \mathbf{a}_1\right), \\
\mathrm{embed}_l(\mathbf{P}_1) &= \mathbf{P}\left(\begin{pmatrix} A_1 & \mathrm{O} \\ \mathrm{O} & I_{l-d} \\ \mathrm{O} & -I_{l-d} \end{pmatrix}, \begin{pmatrix} L_1 \\ \mathrm{O} \\ \mathrm{O} \end{pmatrix}, \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}\right), \\
\mathrm{proj}_k(\mathbf{P}_1) &= \mathbf{P}(A, L, \mathbf{a}_1), \\
&\quad \text{where } \begin{pmatrix} A & L \end{pmatrix} = \begin{pmatrix} A_1 & L_1 \end{pmatrix} \text{ and } A \text{ has } k \text{ columns.}
\end{aligned}$$

Problem of Set Entailment. We should address an open issue: Up to now, there is – to the author's best knowledge – no efficient method to decide subset relations for polyhedra represented as support functions or sops, and it is questionable whether such efficient methods exists.

Overview. The adjacent table provides an overview on the hardness of performing linear transformations, Minkowski sums, closed convex hulls, intersections, and deciding subset relations on polyhedra in the respective representation. The tick indicates computability in (weakly) polynomial time and a minus-sign indicates that the enumeration problem is either NP-hard or its complexity is unknown, see [25].

Representation	$M(\cdot)$	$\cdot + \cdot$	conv $(\cdot \cup \cdot)$	$\cdot \cap \cdot$	$\cdot \subseteq \cdot$
$\mathcal{V}$-representation	✓	✓	✓	–	✓
$\mathcal{H}$-representation	✓[a]	–	–	✓	✓
support function	✓[b]	✓	✓	–	–
sop	✓	✓	✓	✓	–

[a]for automorphism, [b]for endomorphism

3.1 Beyond Template Polyhedra

Sops profit from the underlying $\mathcal{H}$-representation, i. e., we may solve linear programs to test for emptiness or to find relative interior points. Additionally, we may switch from the primal system of linear inequalities to its dual. In this section we shall make use of these techniques and present a method which allows to find the facet-defining half-space of a sop $\mathbf{P}$ in some given direction. This method is then extended to an interpolation method which improves existing over-approximations. The needed geometrical concepts are shortly introduced in the following. A comprehensive introduction can be found in [26]. For the theory of linear programming, see [24].

Let $\mathbf{P} = \mathbf{P}(A, \mathbf{a})$ be an $\mathcal{H}$-polyhedron. The points of $\mathbf{P}$ are those vectors $\mathbf{x}$ which satisfy the system $A\mathbf{x} \le \mathbf{a}$. A point $\mathbf{x}$ of $\mathbf{P}$ is an *interior point* if there exists

a ball $\mathbf{B}_\epsilon = \{\mathbf{x} \,|\, |\mathbf{x}| \leq \epsilon\}$ with $\epsilon > 0$ such that $\mathbf{x} + \mathbf{B}_\epsilon \subseteq \mathbf{P}$. Only full-dimensional polyhedra have interior points. However, any polyhedron $\mathbf{P} = \mathbf{P}(A, \mathbf{a})$ is full-dimensional relatively to its affine hull $\mathrm{aff}(\mathbf{P})$. Hence, we call a point $\mathbf{x}$ of $\mathbf{P}$ a *relative interior point* relatively to $\mathrm{aff}(\mathbf{P})$ if there exists a ball $\mathbf{B}_\epsilon$ with $\epsilon > 0$ such that $(\mathbf{x} + \mathbf{B}_\epsilon) \cap \mathrm{aff}(\mathbf{P}) \subseteq \mathbf{P}$. A *facet-defining* half-space $\mathbf{H}$ of $\mathbf{P}$ is a half-space $\mathbf{H} = \left\{\mathbf{x} \,\middle|\, \mathbf{n}^T\mathbf{x} \leq b\right\}$, $\mathbf{P} \subseteq \mathbf{H}$, such that $\mathbf{P} \cap \left\{\mathbf{x} \,\middle|\, \mathbf{n}^T\mathbf{x} = b\right\}$ has a relative interior point relatively to $\mathrm{aff}(\mathbf{P}) \cap \left\{\mathbf{x} \,\middle|\, \mathbf{n}^T\mathbf{x} = b\right\}$.

The topological concept of a relative interior point can equivalently be defined on the system $A\mathbf{x} \leq \mathbf{a}$ of the polyhedron $\mathbf{P} = \mathbf{P}(A, \mathbf{a})$. Every solution $\mathbf{x}$ of the system of strict linear inequalities $A\mathbf{x} < \mathbf{a}$ is an interior point of $\mathbf{P}$. If $\mathbf{n}^T\mathbf{x} \leq b$ is an inequality of the system $A\mathbf{x} \leq \mathbf{a}$, and if all solutions $\mathbf{x}$ of the system $A\mathbf{x} \leq \mathbf{a}$ satisfy $\mathbf{n}^T\mathbf{x} = b$, then $\mathbf{n}^T\mathbf{x} = b$ is called an *implicit equality* of the system. For any set I of row indices of $A\mathbf{x} \leq \mathbf{a}$ we denote the corresponding subsystem by $A_I\mathbf{x} \leq \mathbf{a}_I$. The linear equalities representing the affine hull are linear combinations of the implicit equalities of the system $A\mathbf{x} \leq \mathbf{a}$ and vice versa. Let I be the set of indices of the implicit equalities in $A\mathbf{x} \leq \mathbf{a}$ and S be the set of the remaining indices. Each solution $\mathbf{x}$ of the system $A_I\mathbf{x} = \mathbf{a}_I$, $A_S\mathbf{x} < \mathbf{a}_S$ is a *relative interior point* of $\mathbf{P}$.

Relative interior points and implicit equalities can be found by means of linear programming: The optimal solution $(\mathbf{x}_0, \lambda_0)$ of the linear program "maximize λ subject to $A_I\mathbf{x} = \mathbf{a}_I$, $A_S + \mathbf{1}\lambda \leq \mathbf{a}_S$, $0 \leq \lambda \leq 1$" determines a relative interior point $\mathbf{x}_0$ if $\lambda_0 > 0$. For $\lambda_0 = 0$ one obtains sufficient hints to find further implicit equalities, see [14]. Let I be the set of indices of all implicit equalities of $\mathbf{P}$ and S the set of the remaining indices. The *facet-defining* inequalities of $\mathbf{P}$ are exactly those inequalities whose index j is in S and the linear program "maximize λ subject to $A_{I\cup\{j\}}\mathbf{x} = \mathbf{a}_{I\cup\{j\}}$, $A_{S\setminus\{j} + \mathbf{1}\lambda \leq \mathbf{a}_{S\setminus\{j\}}$, $0 \leq \lambda \leq 1$" has a positive optimal solution. The corresponding half-spaces to a facet-defining inequality are also facet-defining.

The orthogonal projection of a relative interior point is a relative interior point of the projected set. Let $\mathbf{P} = \mathbf{P}(A, L, \mathbf{a}) \subseteq \mathbb{K}^d$ be a sop, $\mathbf{z}$ be a relative interior point of $\mathbf{P}((A\ L), \mathbf{a})$, and $\mathbf{z}_d$ the vector of the first d coefficients of $\mathbf{z}$. Then $\mathbf{z}_d$ is a relative interior point of $\mathbf{P}$ and $\mathbf{P'} = \mathbf{P}(A, L, \mathbf{a} - (A\ L)\mathbf{z})$ is a sop representing the translated polyhedron $\mathbf{P'} = \mathbf{P} - \mathbf{z}_d$, which contains the origin $\mathbf{0}$ as a relative interior point.

Proposition 4 (Ray Shooting). *Let $\mathbf{P} = \mathbf{P}(A, L, \mathbf{a})$ be a nonempty and complete sop in $\mathbb{K}^d$ which contains the origin $\mathbf{0}$ as a relative interior point. Then the following linear program is feasible for any vector $\mathbf{r} \in \mathbb{K}^d$:*

$$\textit{maximize } \mathbf{r}^T A^T\mathbf{u} \textit{ subject to } L^T\mathbf{u} = \mathbf{0},\ \ \mathbf{a}^T\mathbf{u} = 1,\ \ \mathbf{u} \geq \mathbf{0},$$

and exactly one of the following statements holds:

1. *The linear program is unbounded and $\mathbf{r}$ is not in $\mathrm{aff}(\mathbf{P})$.*
2. *The optimal value equals zero and $\mathbf{P}$ is unbounded in direction $\mathbf{r}$.*
3. *The optimal value $\mathbf{r}^T A^T\mathbf{u}_0$ is positive and $\mathbf{P}$ is bounded in direction $\mathbf{r}$. Let $\lambda = \frac{1}{\mathbf{r}^T A^T\mathbf{u}_0}$, $\mathbf{n} = A^T\mathbf{u}_0$, and $\mathbf{H} = \mathbf{H}(\mathbf{n}, 1)$ be a half-space. Then $\lambda\mathbf{r}$ is a boundary point of $\mathbf{P}$ and $\mathbf{H}$ is a supporting half-space of $\mathbf{P}$ in $\lambda\mathbf{r}$.*

Hence, for any given ray $\mathbf{r}$ we find the maximal length $\lambda = \frac{1}{\mathbf{r}^T A^T \mathbf{u}_0}$ such that $\lambda\mathbf{r}$ is on the boundary of $\mathbf{P}$, and we obtain a supporting half-space of $\mathbf{P}$ in $\lambda\mathbf{r}$. If $\lambda\mathbf{r}$ is a relative interior point of a facet, – which is most likely the case if $\mathbf{r}$ was chosen randomly – then ray shooting returns a facet-defining half-space.[2]

A sop $\mathbf{P}$ and an over-approximating template polyhedron $\mathbf{P}'$ have, in general, none or only a few facet-defining half-spaces in common. Hence, we may use Proposition 4 to find facet-defining half-spaces of $\mathbf{P}$ and add them to $\mathbf{P}'$ yielding a better over-approximation $\mathbf{P}''$ of $\mathbf{P}$. We call $\mathbf{P}''$ an *interpolation* of $\mathbf{P}$ and $\mathbf{P}'$. Throughout this paper we use the following simple interpolation strategy: Initially, $\mathbf{P}''$ is set to the affine hull of $\mathbf{P}$. For an arbitrary inequality of $\mathbf{P}'$ we decide whether it is face-defining in $\mathbf{P}' \cap \mathbf{P}''$. If not, it is removed from $\mathbf{P}'$. Otherwise, we choose $\mathbf{r}$ as a relative interior point of the defined facet, and apply Proposition 4 on $\mathbf{P}$ and $\mathbf{r}$. The resulting half-space is then added to the representation of $\mathbf{P}''$. Now, we proceed with any inequality of $\mathbf{P}'$ until all inequalities of $\mathbf{P}'$ are removed.

More sophisticated interpolations are possible but not investigated here.

4 Reachability Analysis Using Sops

In this section we first give a short outline of the reachability analysis for linear hybrid systems. Then we discuss the usage of sops as a novel exact data structure for the reachability analysis. We will observe the monotonic growth of the assembled sops. While the assembly can be done efficiently, any evaluation of the assembled sops by means of linear programs gets increasingly harder. Finally, we analyze why the LGG-algorithm is that much faster and show how to combine both approaches to obtain a fast and improved reachability algorithm.

Hybrid Systems. A hybrid system $H = (\mathrm{Var}, \mathrm{Mod}, \mathrm{Init}, \mathrm{LDE}, \mathrm{Inv}, \mathrm{Trans})$ encodes the nondeterministic evolution of some initial states over time. A *state* of the hybrid system is uniquely determined by the pair $(\mathbf{x}, m)$ of a real-valued vector $\mathbf{x} \in \mathbb{R}^d$ and a mode m of the finite set Mod of *modes*. A *symbolic state* is a pair $(\mathbf{P}, m)$ of a polyhedron $\mathbf{P} \subseteq \mathbb{R}^d$ and a mode $m \in \mathrm{Mod}$. Each dimension of $\mathbb{R}^d$ is associated with a variable in Var. Init is a designated set of *initial states*. Each mode m is associated with a linear differential equation in LDE of the form $\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + \mathbf{u}(t)$ describing the time derivative of the evolution of the continuous variables Var during the mode m. Here, A is a real-valued $(d \times d)$-matrix, and $\mathbf{u}(t) \in \mathbf{U} \subseteq \mathbb{R}^d$ is given as a bounded polyhedron which models the set of disturbances or admissible inputs of the continuous flow. The system may only remain in a mode m as long as the state $(\mathbf{x}, m)$ is inside the associated

[2] In any case, it is possible to test whether the resulting half-space $\mathbf{H}$ is face-defining: Let d be the dimension of the affine hull of the sop $\mathbf{P}$ and $\mathbf{H}_=$ be the bounding hyperplane of $\mathbf{H}$. Then $\mathbf{H}$ is a facet-defining half-space if and only if $\mathrm{aff}(\mathbf{P} \cap \mathbf{H}_=)$ has the dimension $d-1$. In practice, one has to solve several linear programs which makes this test costly.

invariant $(\mathbf{I}, m) \in \text{Inv}$, i. e., $\mathbf{x}$ is in the polyhedron $\mathbf{I} \subseteq \mathbb{R}^d$. Further, for each mode m there is a finite number of associated discrete transitions. A discrete transition $(\mathbf{G}, m, \text{Asgn}, m') \in \text{Trans}$ is enabled if the state $(\mathbf{x}, m)$ satisfies the guard $(\mathbf{G}, m)$, i. e., $\mathbf{x}$ is in the polyhedron $\mathbf{G} \subseteq \mathbb{R}^d$. If a transition is enabled, the state $(\mathbf{x}, m)$ may jump to $(\mathbf{x}', m')$, where $\mathbf{x}'$ is in the image of $\mathbf{x}$ under the affine transformation Asgn.

For safety checks, we additionally use specialized transitions $(\mathbf{U}, m, \emptyset, \emptyset)$. Here, $(\mathbf{U}, m)$ represents designated states for which we want to decide whether the system can reach these states. As soon as such a reachability is established we may stop the reachability analysis and return an appropriate message.

Reachability Analysis of Linear Hybrid Systems. As in [13], we define the discrete post-operator $\text{post}_d(\mathbf{P}, m)$ as the set of states which are reachable by a discrete transition of $(\mathbf{P}, m)$ and the continuous post-operator $\text{post}_c(\mathbf{P}, m)$ as the set of states which are reachable from $(\mathbf{P}, m)$ by letting an arbitrary amount of time elapse. The set $\mathcal{R}$ of reachable states is then the fix-point of the sequence

$$\mathcal{R}_0 = \text{post}_c(\text{Init}), \quad \mathcal{R}_{k+1} = \mathcal{R}_k \cup \text{post}_c(\text{post}_d(\mathcal{R}_k)).$$

The fix-point computation needs an efficient method to decide set entailment. We already mentioned that for support functions and sops we do not have such an efficient method at hand. While this deficiency might be compensated by using template polyhedra, we restrict this paper to bounded model checking, i. e., we do not compute the actual fix-point, but compute a restricted sequence until a given time bound is exceeded.

The discrete post-operator simply comprises the individual application of the affine transformations to the symbolic states and can be done efficiently using sops. Hence, we dedicate our attention to the continuous post-operator.

Reachability Analysis of Linear Systems. For each symbolic state $(\mathbf{X}_0, m)$, the continuous post-operator boils down to a reachability analysis of a single linear system, as it can be found in [13,20]: Given some initial state set $\mathbf{X}_0 \subseteq \mathbb{R}^d$, we want to compute all reachable states $\mathcal{R}_{[0,t]}$ within the time interval $[0, t]$ under the linear differential equation

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + \mathbf{u}(t), \qquad \mathbf{x}(0) \in \mathbf{X}_0,\ \mathbf{u}(t) \in \mathbf{U}.$$

The computation of the reachable set $\mathcal{R}_{[0,t]}(\mathbf{X}_0)$ is done by a step-wise computation of flow segments $\mathcal{R}_{[k\delta,(k+1)\delta]}(\mathbf{X}_0)$ over a time interval of length δ:

$$\mathcal{R}_{[0,t]}(\mathbf{X}_0) = \bigcup_{k=0,\ldots,N} \mathcal{R}_{[k\delta,(k+1)\delta]}(\mathbf{X}_0).$$

The computation is based on two important ingredients:

- the *initial segment* $\mathcal{R}_{[0,\delta]}(\mathbf{X}_0)$ and a set which collects the influence of the bounded inputs $\mathcal{R}_\delta(\{\mathbf{0}\}) := \mathcal{R}_{[\delta,\delta]}(\{\mathbf{0}\})$;
- the *exact* recurrence relation for $k > 0$

$$\mathcal{R}_{[k\delta,(k+1)\delta]}(\mathbf{X}_0) = \mathrm{e}^{\delta A}\mathcal{R}_{[(k-1)\delta,k\delta]}(\mathbf{X}_0) + \mathcal{R}_\delta(\{\mathbf{0}\}). \tag{1}$$

In general, the sets in the first item are not convex. Hence, one has to compute convex over-approximations of these sets, which is called the *initial bloating* procedure. The recurrence relation (1) is then applied to the bloated sets resulting in an over-approximation of the reachable sets.

Initial Bloating. In this section we discuss how to compute convex over-approximations $\mathbf{R}_0$ of $\mathcal{R}_{[0,\delta]}(\mathbf{X}_0)$ and $\mathbf{V}$ of $\mathcal{R}_\delta(\{\mathbf{0}\})$. There are several ways to compute such over-approximations, varying from conservative over-approximation [15] to an accurate bloating of the convex hull $\mathrm{clconv}(\mathbf{X}_0 \cup \mathrm{e}^{\delta A}(\mathbf{X}_0))$ [8]. The novel method given below is inspired by the method proposed by Le Guernic [20], which has been slightly improved and implemented in SpaceEx [13]. Anyhow, we cannot apply that bloating method directly, since we are dealing with polyhedral sets only, while the method of Le Guernic involves piecewise quadratic functions to describe the support function of the bloated sets. Although we made no effort to give an precise comparison of both bloating methods, we expect the support functions based method to provide better results in general. However, since sops also have support functions, the following bloating procedure can also be applied to reachability analysis using support functions. A detailed comparison of both bloating methods is considered as future work.

We use the superposition principle to decompose $\mathcal{R}_{[0,\delta]}(\mathbf{X}_0)$ into the sum

$$\mathcal{R}_{[0,\delta]}(\mathbf{X}_0) = \bigcup_{t\in[0,\delta]} \mathrm{e}^{tA}(\mathbf{X}_0) + \bigcup_{t\in[0,\delta]} \mathcal{R}_t(\{\mathbf{0}\}).$$

The first summand $\bigcup_{t\in[0,\delta]} \mathrm{e}^{tA}(\mathbf{X}_0)$ is exactly the set of reachable states of the related autonomous system $\dot{\mathbf{x}} = A\mathbf{x}$ within the time interval $[0,\delta]$ and the latter summand $\mathcal{R}_{[0,\delta]}(\{\mathbf{0}\}) = \bigcup_{t\in[0,\delta]} \mathcal{R}_t(\{\mathbf{0}\})$ accounts for the accumulated influences of all admissible inputs. We over-approximate both summands separately and add them afterwards to obtain an over-approximation of the reachable states of the nonautonomous system.

For the following let $\boxdot(\mathbf{X})$ be the symmetric interval hull of $\mathbf{X}$, that is, $\boxdot(\mathbf{X}) = [-z_1, z_1] \times \cdots \times [-z_d, z_d]$ where $z_i = \max\left(\left|\inf_{\mathbf{x}\in\mathbf{X}} \mathbf{e}_i^T\mathbf{x}\right|, \left|\sup_{\mathbf{x}\in\mathbf{X}} \mathbf{e}_i^T\mathbf{x}\right|\right)$. Further, let $|\mathbf{x}|$ and $|A|$ be the vector and the matrix where all coefficients are replaced by their absolute values. Hence, for any vector $\mathbf{x}$ and any set $\mathbf{X}$ we have $\mathbf{x} \le |\mathbf{x}|$ and, if $\mathbf{x} \in \mathbf{X}$, then $|\mathbf{x}| \in \boxdot(\mathbf{X})$. We define the abbreviation $[\![\mathrm{e}^{\delta A}\mathbf{X}]\!] = \boxdot\left(\mathrm{e}^{\delta|A|}(\boxdot(\mathbf{X}))\right)$ and obtain the following over-approximation of $\mathrm{e}^{tA}(\mathbf{X}_0)$.

Lemma 1. *For all $t \in [0,\delta]$ the set inclusion $\mathrm{e}^{tA}(\mathbf{X}) \subseteq [\![\mathrm{e}^{\delta A}\mathbf{X}]\!]$ holds.*

The next lemma is based on a Taylor approximation of mth order and an over-approximation of the Lagrange form of the remainder involving Lemma 1.

Lemma 2. *For any nonnegative integer m and any $t \in [0, \delta]$ the following set inclusions hold*

$$\begin{aligned} \mathrm{e}^{tA}(\mathbf{X}_0) &\subseteq \sum_{k=0}^{m} \frac{t^k}{k!} A^k(\mathbf{X}_0) + \frac{t^{m+1}}{(m+1)!} A^{m+1}([\![\mathrm{e}^{\delta A}\mathbf{X}_0]\!]), \\ \mathcal{R}_t(\{\mathbf{0}\}) &\subseteq \sum_{k=0}^{m} \frac{t^{k+1}}{(k+1)!} A^k(\mathbf{U}) + \frac{t^{m+2}}{(m+2)!} A^{(m+1)}([\![\mathrm{e}^{\delta A}\mathbf{U}]\!]). \end{aligned} \tag{2}$$

For $t = \delta$, (2) already provides an over-approximation of $\mathcal{R}_\delta(\{\mathbf{0}\})$. We choose $m = 0$ and obtain the first-order approximation

$$\mathbf{V} = \delta A^k(\mathbf{U}) + \frac{t^2}{2} A([\![\mathrm{e}^{\delta A}\mathbf{U}]\!]) \supseteq \mathcal{R}_\delta(\{\mathbf{0}\}).$$

We use the fact, that for any $\mathbf{x}$, $k \geq 0$, and $t \in [0, \delta]$ the term $\frac{t^k}{k!} A^k\mathbf{x}$ can be written as the convex combination $(1-\lambda)\mathbf{0} + \lambda \frac{\delta^k}{k!} A^k\mathbf{x}$ with $\lambda = \frac{t^k}{\delta^k}$, and hence, as stipulated, $0 \leq \lambda \leq 1$. We introduce the notion $\lhd(\mathbf{X}) = \mathrm{clconv}(\{\mathbf{0}\} \cup \mathbf{X})$ and obtain a first-order approximations of $\bigcup_{t\in[0,\delta]} \mathrm{e}^{tA}(\mathbf{X}_0)$ and $\bigcup_{t\in[0,\delta]} \mathcal{R}_t(\{\mathbf{0}\})$.

Lemma 3. *The following set inclusions hold*

$$\begin{aligned} \bigcup_{t\in[0,\delta]} \mathrm{e}^{tA}(\mathbf{X}_0) &\subseteq \mathbf{X}_0 + \lhd(\delta A(\mathbf{X}_0)) + \lhd\left(\frac{\delta^2}{2} A^2([\![\mathrm{e}^{\delta A}\mathbf{X}_0]\!])\right), \\ \bigcup_{t\in[0,\delta]} \mathcal{R}_t(\{\mathbf{0}\}) &\subseteq \lhd(\delta\mathbf{U}) + \lhd\left(\frac{\delta^2}{2} A([\![\mathrm{e}^{\delta A}\mathbf{U}]\!])\right). \end{aligned}$$

The first inclusion in Lemma 3 provides an over-approximation of the reachable states in forward direction. We may also compute an over-approximation in backward direction starting from $\mathrm{e}^{\delta A}(\mathbf{X}_0)$. Finally we obtain the proposition:

Proposition 5 (Over-Approximation of $\mathcal{R}_{[0,\delta]}(\mathbf{X}_0)$). *Let $\mathbf{X}_1 = \mathrm{e}^{\delta A}(\mathbf{X}_0)$. Then the following set inclusion holds:*

$$\begin{aligned} \mathcal{R}_{[0,\delta]}(\mathbf{X}_0) \subseteq &\left(\mathbf{X}_0 + \lhd(\delta A(\mathbf{X}_0)) + \lhd\left(\frac{\delta^2}{2} A^2([\![e^{\delta A}(\mathbf{X}_0)]\!])\right)\right) \\ &\cap \left(\mathbf{X}_1 + \lhd(-\delta A(\mathbf{X}_1)) + \lhd\left(\frac{\delta^2}{2}(-A)^2([\![e^{-\delta A}(\mathbf{X}_1)]\!])\right)\right) \\ &+ \lhd(\delta\mathbf{U}) + \lhd\left(\frac{\delta^2}{2} A([\![\mathrm{e}^{\delta A}(\mathbf{U})]\!])\right) = \mathbf{R}_0. \end{aligned}$$

4.1 A Reachability Algorithm for Linear Systems with Invariants

We compute the reachable states of a linear system according to Algorithm 1 [20,21]. The inputs of the algorithm are the first flow segment $\mathbf{R}_0$, the set $\mathbf{V}$ – both obtained by the initial bloating procedure –, the invariant $\mathbf{I}$, and the set

Algorithm 1. Reachability Algorithm for a Linear System (SOP)

Input: the matrix A of the linear differential equation, an invariant $\mathbf{I}$, the set $\mathcal{G}$ of guards, an over-approximation $\mathbf{R}_0 \subseteq \mathbf{I}$ of $\mathcal{R}_{[0,\delta]}(\mathbf{X}_0)$, an over-approximation $\mathbf{V}$ of $\mathcal{R}_\delta(\{\mathbf{0}\})$, and an integer $N = \lfloor \frac{t}{\delta} \rfloor$.

Output: A collection of the intersections of $\mathcal{R}_{[0,t]}(\mathbf{X}_0)$ and the guards in $\mathcal{G}$.

1. **for** $k \leftarrow 0, \ldots, N$ **do**
2. **if** $\mathbf{R}_k = \emptyset$ **then break;**
3. **for** each guard $\mathbf{G}_j \in \mathcal{G}$ **do**
4. **if** $\mathbf{R}_k \cap \mathbf{G}_j \neq \emptyset$ **then** collect the intersection $\mathbf{R}_k \cap \mathbf{G}_j$**;**
5. **end for;**
6. $\mathbf{R}_{k+1} \leftarrow (\mathrm{e}^{\delta A}\mathbf{R}_k + \mathbf{V}) \cap \mathbf{I}$**;**
7. **end for;**
8. **return** collected intersections with the guards**;**

$\mathcal{G} = \{\mathbf{G}_1, \ldots, \mathbf{G}_g\}$ of guards. The computation of the next flow segment in Line 6 is based on (1). Additionally, this computation fully respects the influences of the invariant. Hence, the sequence $(\mathbf{R}_k)$ of flow segments computed by the algorithm is exact provided $\mathbf{R}_0$, $\mathbf{V}$, and $\mathrm{e}^{\delta A}$ are exact. In Lines 3-5 the intersections of the current flow segment with the guards are computed and collected. Yet, we did not specify how the actual collection is performed. There are several possibilities varying from returning each single intersection, which potentially leads to a multiplication of the symbolic state sets, to returning the convex hull of all intersections. We implemented a collection strategy where an individual convex hull is assembled of all intersections for each guard traversal.

The following observations were made on a implementation of Algorithm 1.

1. The algorithm provides a new degree of exactness. The only theoretical source of inexactness are the computation of the matrix exponential $\mathrm{e}^{\delta A}$, the over-approximations due to the initial bloating procedure, and the over-approximations in the collection step (Line 4). In practice, we also have to care for numerical issues due to the usage of floats.
2. The main drawback of a pure sop-based approach is the monotonic growth of the representation matrices of the involved sops. While the assembly of huge sops (Line 6) can be done efficiently, the evaluation of such sops gets increasingly harder. Based on our experiences we assess the following parts of the reachability analysis in order of increasing influence on the growth of the sops:
   (a) The initial bloating procedure has the mildest influence on the growth, since it is only applied once for each symbolic state.
   (b) The intersection with the invariant can be efficiently combined with a redundancy removal to avoid unneeded growth.
   (c) While the implemented collection strategy keeps the number of symbolic states small, the representation matrices of such collections can be quite large. The size is highly dependent on the time step parameter δ.
   (d) The Minkowski sum in Line 6 has the highest influence: A nonempty set $\mathbf{V}$ leads to a linear growth of the representation matrices.

4.2 Le Guernic and Girard's Reachability Algorithm

Compared in run-time, our prototype of Algorithm 1 is clearly behind the reachability algorithm of Le Guernic and Girard [21]. Algorithm 2 restates their algorithm in our context. The algorithm is based on a clever combination of support functions and template polyhedra and profits from the weaker handling of the invariant. In fact, the influence of the invariant only accounts for the current flow segment and is not carried over to the next flow segment. This leads to an efficient computation of the next flow segment in Lines 7–9, where the invariant is completely ignored. The influences of the bounded input are accumulated in the sequence $(\mathbf{s}_k)$, and, instead of updating the state set $\mathbf{R}_0$, only an updated template matrix T_{k+1} is computed; based on the fact that the optimal values of the two linear programs "maximize $\mathbf{n}^T\mathbf{x}$ subject to $\mathbf{x} \in \mathrm{e}^{k\delta A}(\mathbf{R}_0)$" and "maximize $(\mathbf{n}^T\mathrm{e}^{k\delta A})\mathbf{x}$ subject to $\mathbf{x} \in \mathbf{R}_0$" agree. In every step the computation of the template polyhedra $\mathbf{P}(T_0, \mathbf{b}_{k+1})$, which over-approximate the flow segments $\mathbf{R}_{k+1}$, can be done in constant time[3]. The quality of the over-approximation highly depends on the template matrix T_0. In order to improve the handling of the invariant, the facet normals of the invariant should be added to the template directions [13].

Algorithm 2. Reachability Algorithm for a Linear System (LGG)

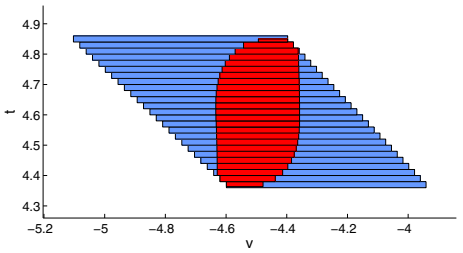
Input: A, I, $\mathcal{G}$, $\mathbf{R}_0$, $\mathbf{V}$, N as specified in Algorithm 1 and an additional template polyhedron $\mathbf{P}(T_0, \mathbf{b}_0)$ over-approximating $\mathbf{R}_0$.

Output: A collection of the intersections of $\mathcal{R}_{[0,t]}(\mathbf{X}_0)$ and the guards in $\mathcal{G}$.

1. $\mathbf{s}_0 \leftarrow \mathbf{0}$;
2. **for** $k \leftarrow 0, \ldots, N$ **do**
3. **if** $\mathbf{P}(T_0, \mathbf{b}_k) \cap \mathbf{I} = \emptyset$ **then break;**
4. **for** each guard $\mathbf{G}_j \in \mathcal{G}$ **do**
5. **if** $\mathbf{P}(T_0, \mathbf{b}_k) \cap \mathbf{I} \cap \mathbf{G}_j \neq \emptyset$ **then** collect the intersection $\mathbf{P}(T_0, \mathbf{b}_k) \cap \mathbf{I} \cap \mathbf{G}_j$;
6. **end for;**
7. $\mathbf{s}_{k+1} \leftarrow \mathbf{s}_k + \mathbf{h}_{\mathbf{V}}(T_k)$;
8. $T_{k+1} \leftarrow T_k\mathrm{e}^{\delta A}$;
9. $\mathbf{b}_k \leftarrow \mathbf{h}_{\mathbf{R}_0}(T_{k+1}) + \mathbf{s}_{k+1}$;
10. **end for;**
11. **return** collected intersections with the guards;

Combining Algorithm 1 and Algorithm 2. Algorithm 3 is a combination of Algorithm 1 and Algorithm 2. While it preserves the exactness of the sop-based algorithm, all involved linear programs have a constant number of variables and constraints (Lines 3, 5, and 12). Also the assembly of the sop in Line 10 and

[3] This is more a practical observation than a theoretical result. Although there exists an algorithm which solves *rational* linear programs of fixed dimension and m constraints in $\mathcal{O}(m)$ elementary arithmetic operations on numbers of polynomial size, the complexity of linear programs is usually given by a polynomial bound which also depends on the maximum bit size of the coefficients [24].



This figure shows the first intersection of a bouncing ball with the guard (the floor). For the model description see Sect. 5. We used the time step $\delta = 0.02$. The outer slices show the intersections computed by Algorithm 2 using a rectangular template matrix. Each inner slice shows a tight rectangular over-approximation of the sops computed by Algorithm 1. Their representation matrices reach a size of about 3500 rows and 2000 columns with 9000 nonzero coefficients. The convex hull of all intersections has a size of 82652 rows, 46999 columns and 283006 nonzero coefficients.

Fig. 1. Comparison of Algorithm 1 and 2

Line 12 can be done in constant time. Lines 10–13 are an equivalent replacement for the assignment $\mathbf{R}_{k+1} \leftarrow (\mathrm{e}^{\delta A}\mathbf{R}_k + \mathbf{V}) \cap \mathbf{I}$ with an additional redundancy removal, see also Item 2b in Sect. 4.1.

Algorithm 3. Reachability Algorithm for a Linear System (SOP + LGG)

Input: A, I, $\mathcal{G}$, $\mathbf{R}_0$, $\mathbf{V}$, N as specified in Algorithm 1 and an additional template polyhedron $\mathbf{P}(T_0, \mathbf{b}_0)$ over-approximating $\mathbf{R}_0$.

Output: A collection of the intersections of $\mathcal{R}_{[0,t]}(\mathbf{X}_0)$ and the guards in $\mathcal{G}$.

1. $\mathbf{s}_0 \leftarrow \mathbf{0}$;
2. **for** $k \leftarrow 0, \ldots, N$ **do**
3. **if** $\mathbf{P}(T_0, \mathbf{b}_k) \cap \mathbf{I} = \emptyset$ **then break**;
4. **for** each guard $\mathbf{G}_j \in \mathcal{G}$ **do**
5. **if** $\mathbf{P}(T_0, \mathbf{b}_k) \cap \mathbf{I} \cap \mathbf{G}_j \neq \emptyset$ **then** collect the intersection $\mathbf{R}_k \cap \mathbf{G}_j$;
6. **end for**;
7. $\mathbf{s}_{k+1} \leftarrow \mathbf{s}_k + \mathbf{h}_{\mathbf{V}}(T_k)$;
8. $T_{k+1} \leftarrow T_k \mathrm{e}^{\delta A}$;
9. $\mathbf{b}_{k+1} \leftarrow \mathbf{h}_{\mathbf{R}_0}(T_{k+1}) + \mathbf{s}_{k+1}$;
10. $\mathbf{R}_{k+1} \leftarrow \mathrm{e}^{\delta A}\mathbf{R}_k + \mathbf{V}$;
11. **for** each constraint $\mathbf{c}_i$ of $\mathbf{I}$ **do**
12. **if** $\mathbf{c}_i$ is not redundant in $\mathbf{P}(T_0, \mathbf{b}_{k+1})$ **then** $\mathbf{R}_{k+1} \leftarrow \mathbf{R}_{k+1} \cap \mathbf{c}_i$;
13. **end for**;
14. **end for**;
15. **return** collected intersections with the guards;

Fighting the Monotonic Growth by Interpolation. In practice, the step computation of Algorithm 3 is done in constant time. But the size of the sops still grows monotonically. While this growth can still be handled during the collection and the discrete updates, latest in the next continuous iteration, when the discrete post-image of a symbolic state is passed to Algorithm 3 again, the enormous size of the sop has an effect: All involved linear programs of Algorithm 3 have

to be solved over systems of linear inequalities of enormous size.[4] To overcome this problem, we use the ray shooting based interpolation as described in Sect. 3.1. In every step, we have two representations of the current flow segment: the template polyhedron $\mathbf{P}(T_0, \mathbf{b}_k) \cap \mathbf{I}$ and the set $\mathbf{R}_k$ represented by a sop. Hence, we compute an interpolating $\mathcal{H}$-polyhedron $\mathbf{Q}$ with $\mathbf{R}_k \subseteq \mathbf{Q} \subseteq \mathbf{P}(T_0, \mathbf{b}_k) \cap \mathbf{I}$. This interpolating polyhedron is a tight over-approximation of $\mathbf{R}_k$ and is at least as good as the template polyhedron computed by the LGG-algorithm 2. Then we replace $\mathbf{R}_k$ by the interpolating polyhedron and still achieve results which are at least as good as the results we would achieve by the pure LGG-algorithm. In our prototype the interpolation and replacement of $\mathbf{R}_k$ is applied after `interpolate_after` step computations. The interpolation can be disabled by setting `interpolate_after` $= 0$.

We use a similar strategy to confine the growth of the collected intersections. Instead of building the convex hull of an arbitrary sequence, we apply the convex hull and the template hull on at most `max_conv_hull` consecutive elements of the sequence. Then we compute the interpolation between the template hull and the convex hull. The resulting interpolations form a new sequence for which we proceed as before. We iterate this process until only one element remains. Again, this interpolation strategy can be disabled. The resulting set is at least as good as the result one would achieve with template polyhedra only.

5 Experimental Results

We compare our prototypical implementations of Algorithm 3 and Algorithm 2 against the productive implementation of the LGG-algorithm in SpaceEx, where we used the SpaceEx Virtual Machine Server v0.9.8b for the comparison. The prototype, called SoapBox, is implemented in Matlab and uses Gurobi Optimizer 5.6[5] for the linear programming tasks. Furthermore, it has successfully been applied in a case study [10,11].

Bouncing Ball. For our benchmarks we have chosen a simple model of a bouncing ball. The dynamics of the model are given by $\dot{x} = v$, $\dot{v} = -1 \pm 0.05$, and $\dot{t} = 1$. The ball bounces as soon as it reaches the floor which is modeled by the invariant $x \geq 0$ and the transition $v \leftarrow -\frac{3}{4}v$, guarded by $x \leq 0$ and $v \leq 0$. The initial states are given by the interval hull of $10 \leq x \leq 10.2$, $0 \leq v \leq 0.2$, and $t = 0$.

Table 1 shows the run-times in seconds for different time steps δ and different numbers of iterations. Throughout all computations we used a rectangular template matrix, and the sop-specific configuration parameters were set as follows: `interpolate_after` $= 20$ and `max_conv_hull` $= 4$. Clearly, a C++-implementation of the LGG-algorithm, as it can be found in SpaceEx, outperforms our Matlab-implementation. The run-times of our LGG-algorithm and

[4] Actually, the enormous size of the sops already effects the initial bloating procedure.
[5] `http://www.gurobi.com`

Table 1. Run-Time Comparison of Algorithm 3, Algorithm 2, and SpaceEx (SPX)

# It:	4			5			6		
δ	Alg. 3	Alg. 2	SPX	Alg. 3	Alg. 2	SPX	Alg. 3	Alg. 2	SPX
0.08	21.16	15.87	1.88	27.25	29.32	3.90	31.55	56.47	5.78
0.04	46.08	29.74	4.33	58.59	56.92	8.28	68.48	106.69	11.51
0.02	89.24	58.45	7.65	116.59	109.69	14.33	137.16	209.93	25.69
0.01	178.85	114.91	17.03	227.56	218.89	28.88	270.91	424.64	48.47

SpaceEx differ by a factor of 6.9 to 9.8. We have to bear in mind that SpaceEx additionally performs fix-point checks. Anyhow, the comparison with SpaceEx might give a hint what speed-up could be expected for a C++-implementation of our algorithms. We also should note that SpaceEx uses nonpolyhedral bloating.

More significant is a comparison of our prototypes of the LGG-algorithm 2 and the combined Algorithm 3, since they are embedded in the same overall reachability algorithm. For an increasing number of iterations, we observe that Algorithm 3 outperforms Algorithm 2 despite the computational overhead. Figure 2 shows the reachable positions x over the time t for 6 iterations. The left hand side diagram shows the reachable states computed by Algorithm 3 and the right hand side diagram shows the reachable states computed by the LGG-algorithm[6]. The reachable states computed by Algorithm 3 lie within the time interval $[0, 35]$, while the reachable states computed by the LGG-algorithm extend to nearly $t \in [0, 90]$ due to the poor handling of intersections and invariants. Hence, the LGG-algorithm has to perform much more flow-segment computations.

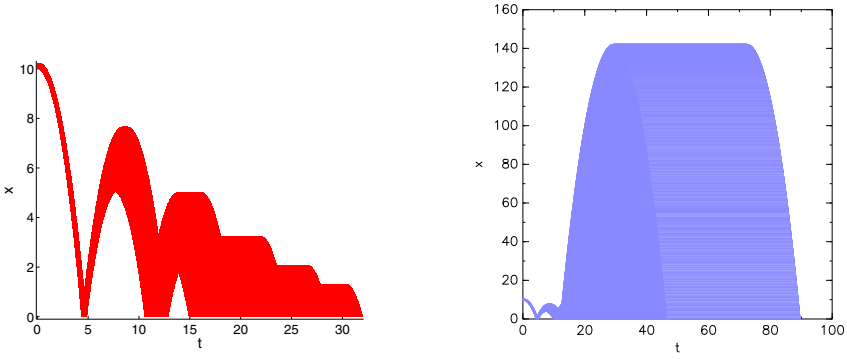


Fig. 2. Comparison of Algorithm 3 and SpaceEx

Approach Velocity Controller. In Fig. 3 we compare the reachable states computed by Algorithm 3 on the left and SpaceEx on the right. We used a rectangular template matrix and the parameters $\delta = 0.5$ and `interpolate_after` =

[6] The figure actually show the SpaceEx output. The output of Algorithm 2 looks quite the same, but we think it is more impressive to compare with SpaceEx here.

40. The underlying model is a single mode approach velocity controller (AVC). The AVC controls the velocity v of a following car in order to establish the desired distance d_{des} to the leading car which has the velocity v_a. The current distance of the cars can be read off the variable d. The dynamics are

$$\begin{aligned} \dot{d} &= v_a - v, & \dot{v} &= 0.29(v_a - v) + 0.01(d - d_{\text{des}}), & \dot{t} &= 1, \\ -0.5 \leq \dot{v}_a &\leq 0.5, & 0 \leq v_a &\leq 20. \end{aligned} \tag{3}$$

The inequalities (3) restrict the allowed velocity of the leading car: While the differential inclusion allows some restricted change of the velocity, the invariant restricts the velocity to a bounded interval. Initially, both cars have a velocity of $20\frac{m}{s}$ and a distance of $450m$. By the invariant, the leader is not allowed to drive backward or exceed some maximal velocity. Clearly, one should expect that this behavior carries over to the following car, i. e., that the velocity of the follower is asymptotically bounded by some interval. A comparison of the right and the left figure shows that SpaceEx (right) is not able to establish any bound on the velocity of the follower while Algorithm 3 (left) shows the desired behavior.

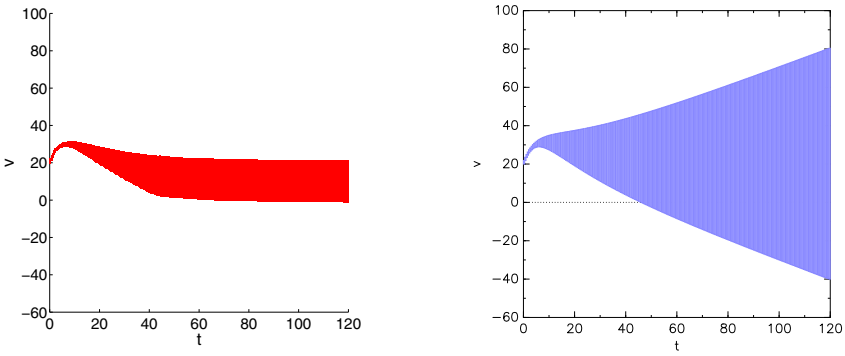


Fig. 3. Comparison of Algorithm 3 and SpaceEx

6 Conclusion

We introduced a novel representation class for polyhedra, the symbolic orthogonal projections (sops). Various geometric operations can efficiently be performed on this representation class. Together with linear programming, sops can be used to implement an reachability algorithm where all polyhedral operations are done exactly (Algorithm 1). Due to the monotonic growth of the representation size, this algorithm is not suitable for practical applications. After combining Algorithm 1 with the LGG-algorithm we achieve an efficient reachability algorithm (Algorithm 3). The applicability, accuracy, and efficiency of the resulting algorithm is demonstrated on some simple examples.

Acknowledgements. I would like to thank Uwe Waldmann for the useful discussion on this paper. Also, I am thankful for the anonymous reviewers for their constructive comments.

References

1. Althoff, M., Krogh, B.H.: Avoiding geometric intersection operations in reachability analysis of hybrid systems. In: Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2012, pp. 45–54. ACM, New York (2012)
2. Asarin, E., Dang, T., Girard, A.: Reachability analysis of nonlinear systems using conservative approximation. In: Maler, O., Pnueli, A. (eds.) HSCC 2003. LNCS, vol. 2623, pp. 20–35. Springer, Heidelberg (2003)
3. Asarin, E., Dang, T., Girard, A.: Hybridization methods for the analysis of nonlinear systems. Acta Informatica 43(7), 451–476 (2007)
4. Asarin, E., Dang, T., Maler, O.: The d/dt tool for verification of hybrid systems. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, pp. 365–370. Springer, Heidelberg (2002)
5. Bournez, O., Maler, O., Pnueli, A.: Orthogonal polyhedra: Representation and computation. In: Vaandrager, F.W., van Schuppen, J.H. (eds.) HSCC 1999. LNCS, vol. 1569, pp. 46–60. Springer, Heidelberg (1999)
6. Chen, X., Ábrahám, E., Sankaranarayanan, S.: Flow*: An analyzer for non-linear hybrid systems. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 258–263. Springer, Heidelberg (2013)
7. Chutinan, A., Krogh, B.: Computational techniques for hybrid system verification. IEEE Transactions on Automatic Control 48(1), 64–75 (2003)
8. Chutinan, A., Krogh, B.H.: Computing polyhedral approximations to flow pipes for dynamic systems. In: Proceedings of the 37th IEEE Conference on Decision and Control, 1998, vol. 2, pp. 2089–2094 (December 1998)
9. Damm, W., Dierks, H., Disch, S., Hagemann, W., Pigorsch, F., Scholl, C., Waldmann, U., Wirtz, B.: Exact and fully symbolic verification of linear hybrid automata with large discrete state spaces. Science of Computer Programming 77(10-11), 1122–1150 (2012), aVoCS 2009
10. Damm, W., Hagemann, W., Möhlmann, E., Rakow, A.: Component based design of hybrid systems: A case study on concurrency and coupling. Reports of SFB/TR 14 AVACS 95, SFB/TR 14 AVACS (2014), `http://www.avacs.org`, ISSN: 1860-9821
11. Damm, W., Möhlmann, E., Rakow, A.: Component based design of hybrid systems: A case study on concurrency and coupling. In: Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control, HSCC 2014, pp. 145–150. ACM, New York (2014)
12. Dang, T., Maler, O., Testylier, R.: Accurate hybridization of nonlinear systems. In: Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2010, pp. 11–20. ACM, New York (2010)
13. Frehse, G., et al.: SpaceEx: Scalable verification of hybrid systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 379–395. Springer, Heidelberg (2011)
14. Fukuda, K.: Lecture: Polyhedral computation, spring 2011 (2011), `http://stat.ethz.ch/ifor/teaching/lectures/poly_comp_ss11/lecture_notes`

15. Girard, A.: Reachability of uncertain linear systems using zonotopes. In: Morari, M., Thiele, L. (eds.) HSCC 2005. LNCS, vol. 3414, pp. 291–305. Springer, Heidelberg (2005)
16. Girard, A., Le Guernic, C.: Zonotope/hyperplane intersection for hybrid systems reachability analysis. In: Egerstedt, M., Mishra, B. (eds.) HSCC 2008. LNCS, vol. 4981, pp. 215–228. Springer, Heidelberg (2008)
17. Hagemann, W.: Reachability analysis of hybrid systems using symbolic orthogonal projections. Reports of SFB/TR 14 AVACS 98, SFB/TR 14 AVACS (2014), http://www.avacs.org, ISSN: 1860-9821
18. Henzinger, T.A., Ho, P.H., Wong-Toi, H.: Hytech: Amodel checker for hybrid systems. International Journal on Software Tools for Technology Transfer 1(1-2), 110–122 (1997)
19. Kurzhanski, A.B., Varaiya, P.: Ellipsoidal techniques for reachability analysis. In: Lynch, N.A., Krogh, B.H. (eds.) HSCC 2000. LNCS, vol. 1790, pp. 202–214. Springer, Heidelberg (2000)
20. Le Guernic, C.: Reachability analysis of hybrid systems with linear continuous dynamics. Ph.D. thesis, Université Grenoble 1 - Joseph Fourier (2009)
21. Le Guernic, C., Girard, A.: Reachability analysis of hybrid systems using support functions. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 540–554. Springer, Heidelberg (2009)
22. Prabhakar, P., Viswanathan, M.: A dynamic algorithm for approximate flow computations. In: Proceedings of the 14th International Conference on Hybrid Systems: Computation and Control, HSCC 2011, pp. 133–142. ACM, New York (2011)
23. Sankaranarayanan, S., Dang, T., Ivančić, F.: Symbolic model checking of hybrid systems using template polyhedra. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 188–202. Springer, Heidelberg (2008)
24. Schrijver, A.: Theory of Linear and Integer Programming. John Wiley & Sons, Chichester (1986)
25. Tiwary, H.R.: On the hardness of computing intersection, union and Minkowski sum of polytopes. Discrete & Computational Geometry 40(3), 469–479 (2008)
26. Ziegler, G.M.: Lectures on polytopes, vol. 152. Springer (1995)