Efficient Geometric Operations on Polyhedra

Willem Hagemann

Max Planck Institute for Informatics

Nanning, December 12, 2013

Overview

- motivation: reachability analysis of hybrid systems
- polyhedra
- geometrical operations: convex hull, Minkowski sum, intersection, linear transformations,...
- support functions and template polyhedra
- symbolic orthogonal projections *(new)*

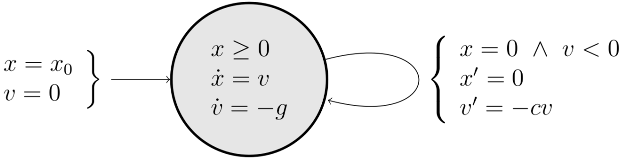
Motivation / Reachability Analysis of Hybrid Systems

A linear hybrid system consists of several linear systems (*modes*)

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

which are connected by *discrete transitions* (jumps).

- A *state* of the hybrid system is the pair $(m, \mathbf{x})$ of a mode m and a vector $\mathbf{x}$ of values for the variables.
- In each mode the variables can only take values within the mode specific *invariant*.
- Discrete transitions are triggered by conjunctions of linear constraints. A transition assigns a new value to the mode variable m, and the vector $\mathbf{x}$ is updated by a linear transformation.



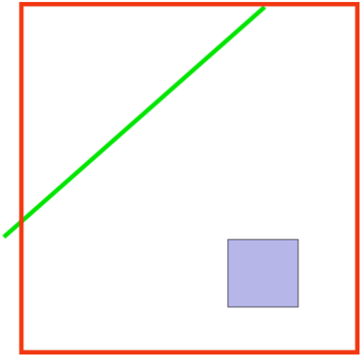
A simple hybrid system: the bouncing ball

Motivation / Reachability Analysis of Hybrid Systems

Given an initial set, an interesting region, an mode invariant and the ODE

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

We compute the reachable states step-wise.
Initial set



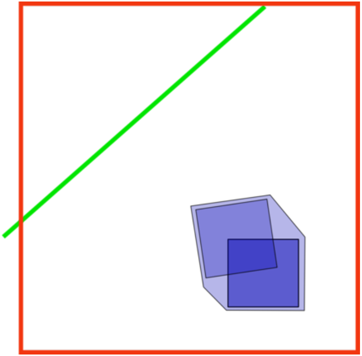
Motivation / Reachability Analysis of Hybrid Systems

Given an initial set, an interesting region, an mode invariant and the ODE

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

We compute the reachable states step-wise.

Initial bloating



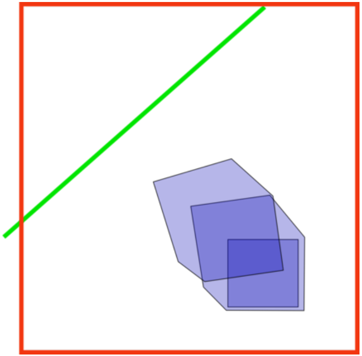
Motivation / Reachability Analysis of Hybrid Systems

Given an initial set, an interesting region, an mode invariant and the ODE

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

We compute the reachable states step-wise.

Next segment



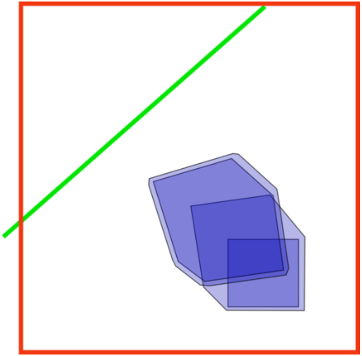
Motivation / Reachability Analysis of Hybrid Systems

Given an initial set, an interesting region, an mode invariant and the ODE

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

We compute the reachable states step-wise.

Adding bounded input

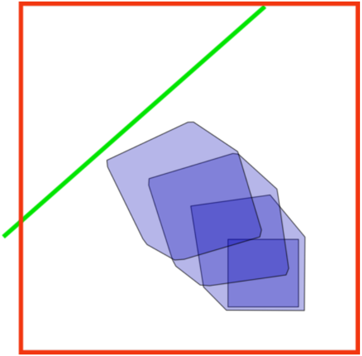


Motivation / Reachability Analysis of Hybrid Systems

Given an initial set, an interesting region, an mode invariant and the ODE

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

We compute the reachable states step-wise.
Next segment



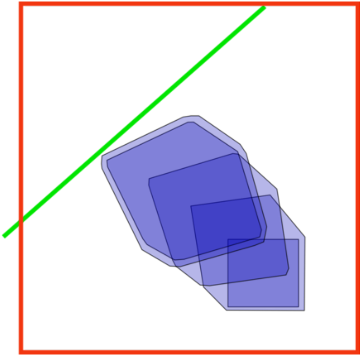
Motivation / Reachability Analysis of Hybrid Systems

Given an initial set, an interesting region, an mode invariant and the ODE

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

We compute the reachable states step-wise.

Adding bounded input



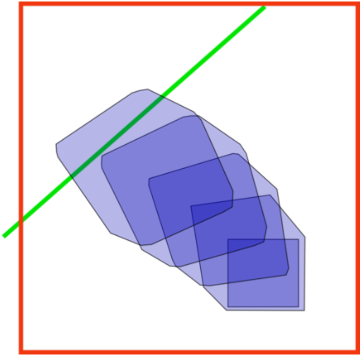
Motivation / Reachability Analysis of Hybrid Systems

Given an initial set, an interesting region, an mode invariant and the ODE

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

We compute the reachable states step-wise.

Next segment



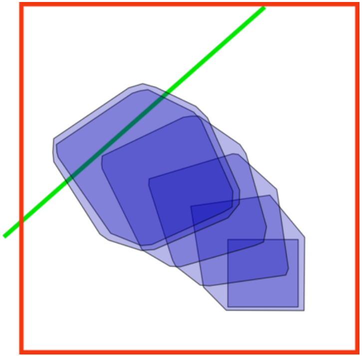
Motivation / Reachability Analysis of Hybrid Systems

Given an initial set, an interesting region, an mode invariant and the ODE

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

We compute the reachable states step-wise.

Adding bounded input



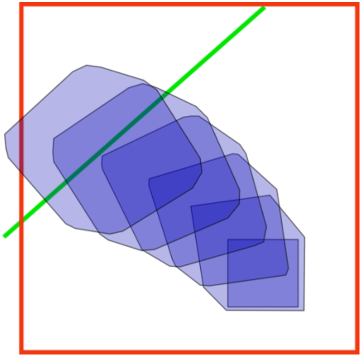
Motivation / Reachability Analysis of Hybrid Systems

Given an initial set, an interesting region, an mode invariant and the ODE

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

We compute the reachable states step-wise.

Next segment



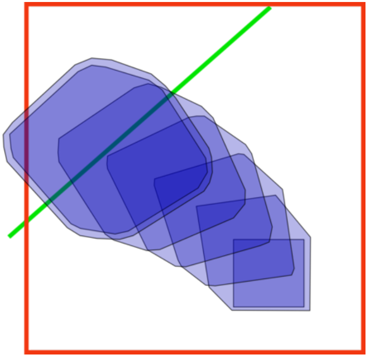
Motivation / Reachability Analysis of Hybrid Systems

Given an initial set, an interesting region, an mode invariant and the ODE

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

We compute the reachable states step-wise.

Adding bounded input

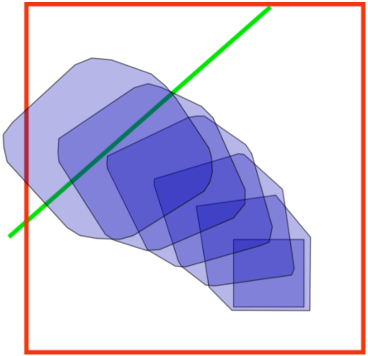


Motivation / Reachability Analysis of Hybrid Systems

Given an initial set, an interesting region, an mode invariant and the ODE

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

We compute the reachable states step-wise.
Restrict to invariant



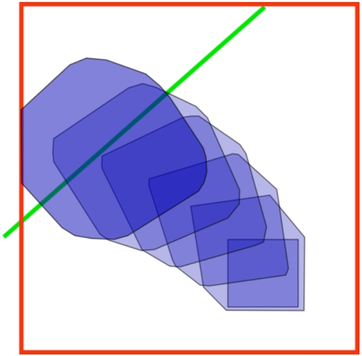
Motivation / Reachability Analysis of Hybrid Systems

Given an initial set, an interesting region, an mode invariant and the ODE

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

We compute the reachable states step-wise.

Restrict to invariant



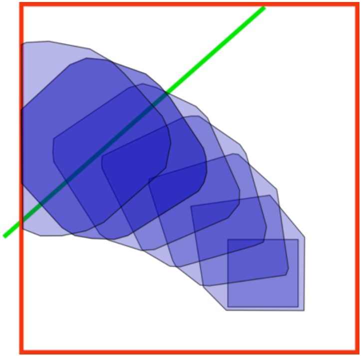
Motivation / Reachability Analysis of Hybrid Systems

Given an initial set, an interesting region, an mode invariant and the ODE

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

We compute the reachable states step-wise.

Restrict to invariant

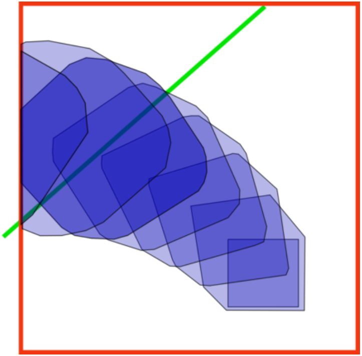


Motivation / Reachability Analysis of Hybrid Systems

Given an initial set, an interesting region, an mode invariant and the ODE

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

We compute the reachable states step-wise.
Restrict to invariant



Motivation / Reachability Analysis of Hybrid Systems

Given an initial set, an interesting region, an mode invariant and the ODE

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

We compute the reachable states step-wise.

Input: ODE A, invariant $\mathbf{I}$, $\mathcal{G}$ set of guards, over-approx. $\mathbf{R}_0 \subseteq \mathbf{I}$ of initial set over-approx. $\mathbf{V}$ of bounded input, and an integer N.

Output: A collection of intersections of the reachable states with guards in $\mathcal{G}$.

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

Motivation

Various geometrical operations are used in the reachability analysis of hybrid systems.

How can we implement these operations efficiently?

The state of the art verification tool *SpaceEx* uses support functions and template polyhedra.

References:
Le Guernic, Girard. *Reachability analysis of hybrid systems using support functions*, CAV 2009
Frehse, et al. *SpaceEx: Scalable verification of hybrid systems*, CAV 2011

Motivation

Various geometrical operations are used in the reachability analysis of hybrid systems.

How can we implement these operations efficiently?

The state of the art verification tool *SpaceEx* uses support functions and template polyhedra.

References:

Le Guernic, Girard. *Reachability analysis of hybrid systems using support functions*, CAV 2009

Frehse, et al. *SpaceEx: Scalable verification of hybrid systems*, CAV 2011

Motivation

Various geometrical operations are used in the reachability analysis of hybrid systems.

How can we implement these operations efficiently?

The state of the art verification tool *SpaceEx* uses support functions and template polyhedra.

References:

Le Guernic, Girard. *Reachability analysis of hybrid systems using support functions*, CAV 2009

Frehse, et al. *SpaceEx: Scalable verification of hybrid systems*, CAV 2011

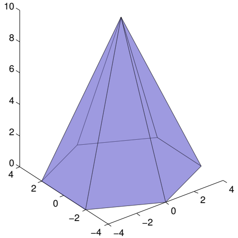
Polyhedra

A polyhedron $\mathbf{P}$ is a convex set with planar facets.

- Typical representations:

  $\mathcal{H}$-representation $\mathbf{P} = \mathbf{P}(A, \mathbf{a}) = \{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{a}\}$,
  $(A, \mathbf{a})$ is a system of linear ineq.

  $\mathcal{V}$-representation $\mathbf{P} = \mathrm{cone}(\mathbf{U}) + \mathrm{conv}(\mathbf{V})$,
  $\mathbf{u} \in \mathbf{U}$ are the rays,
  $\mathbf{v} \in \mathbf{V}$ are the vertices of $\mathbf{P}$



- Conversion between both representation is known as *vertex enumeration* and *facet enumeration problem*.

- Conversion is *expensive*.

Geometrical Operations

We are interested in the following geometrical operations:

convex hull	$\operatorname{conv}(\mathbf{P} \cup \mathbf{Q})$, smallest *closed* convex set including $\mathbf{P}$ and $\mathbf{Q}$
Minkowski sum	$\mathbf{P} + \mathbf{Q}$, adding each vector in $\mathbf{P}$ to each vector in $\mathbf{Q}$
intersection	$\mathbf{P} \cap \mathbf{Q}$, all vectors in $\mathbf{P}$ and $\mathbf{Q}$
linear map	$M(\mathbf{P})$, applying M to each vector in $\mathbf{P}$

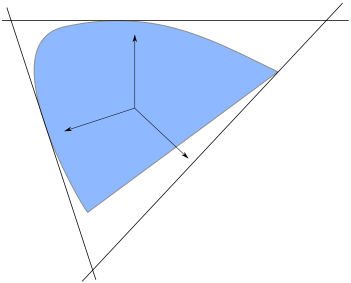
Efficiency of these operations depends on the representation, e. g.

- convex hull and Minkowski sum:
  easy for $\mathcal{V}$-representation, but hard for $\mathcal{H}$-representation
- intersection:
  easy for $\mathcal{H}$-representation, but hard for $\mathcal{V}$-representation

Support Functions

The value of the *support function* of a convex set $\mathbf{S}$ in the direction $\mathbf{n}$ is defined as

$$h_{\mathbf{S}}(\mathbf{n}) := \sup_{\mathbf{x}\in\mathbf{S}} \mathbf{n}^T\mathbf{x}, \quad h_{\mathbf{S}}(\mathbf{n}) \in \mathbb{R} \cup \{-\infty, \infty\}$$



For a polyhedron $\mathbf{P} = \mathbf{P}(A, \mathbf{a})$ this agrees with the optimal value of the LP

$$\text{maximize } \mathbf{n}^T\mathbf{x} \text{ subject to } A\mathbf{x} \leq \mathbf{a}.$$

Geometrical Operations and Support Functions

Let $\mathbf{P}$ and $\mathbf{Q}$ be polyhedra in $\mathbb{R}^d$ and M be the matrix of a linear map $\mathbb{R}^d \to \mathbb{R}^d$.

Support functions behave nicely under most geometrical operations:

convex hull $\quad h_{\mathrm{conv}(\mathbf{P}\cup\mathbf{Q})} = \max(h_{\mathbf{P}}(\mathbf{n}), h_{\mathbf{Q}}(\mathbf{n}))$

Minkowski sum $\quad h_{\mathbf{P}+\mathbf{Q}}(\mathbf{n}) = h_{\mathbf{P}}(\mathbf{n}) + h_{\mathbf{Q}}(\mathbf{n})$

linear map $\quad h_{M(\mathbf{P})}(\mathbf{n}) = h_{\mathbf{P}}(M^T\mathbf{n})$

but intersection is not easy to compute:

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

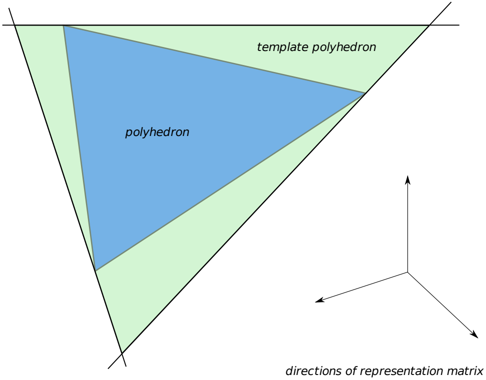
hence, one might use the over-approximation

$$h_{\mathbf{P}\cap\mathbf{Q}}(\mathbf{n}) \leq \min(h_{\mathbf{P}}(\mathbf{n}), h_{\mathbf{Q}}(\mathbf{n}))$$

Template Polyhedra

A *template polyhedron* $\mathbf{P} = \mathbf{P}(A_{\mathrm{fix}}, \mathbf{a})$ has a representation matrix A_{fix} which is fixed a priori.

- Template polyhedra are used to sample support functions,
- where each row of A_{fix} is a sampling direction.



Symbolic Orthogonal Projections

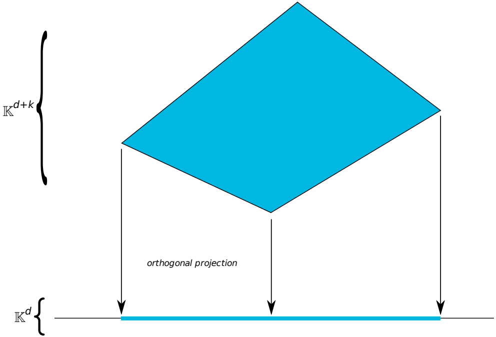
A *symbolic orthogonal projection* (sop) $\mathbf{P} \subseteq \mathbb{K}^d$ is a polyhedron given by

$$\mathbf{P}(A, L, \mathbf{a}) = \{\mathbf{x} \mid \exists \mathbf{z},\ A\mathbf{x} + L\mathbf{z} \leq \mathbf{a}\},$$

where A is a $(m \times d)$-matrix, L is a $(m \times k)$-matrix, $k \geq 0$, and $\mathbf{a}$ is a column vector with m entries.

The idea:

- A sop $\mathbf{P}(A, L, \mathbf{a}) \subseteq \mathbb{K}^d$ is represented by the orthogonal projection of an $\mathcal{H}$-polyhedron $\mathbf{P}((A\ L), \mathbf{a}) \subseteq \mathbb{K}^{d+k}$.
- Any $\mathcal{H}$-polyhedron $\mathbf{P}(A, \mathbf{a})$ can be seen as a sop $\mathbf{P}(A, \emptyset, \mathbf{a})$.



Note, that we *do not* compute the actual $\mathcal{H}$-representation of $\mathbf{P}$ (which would be hard for a non-trivial matrix L). We treat the orthogonal projection symbolically.

Some Technical Details

Complete sop:

- A sop $\mathbf{P}(A, L, \mathbf{a})$ is *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 $(A, L, \mathbf{a})$.
- Any sop $\mathbf{P}$ representing a full-dimensional polytope (i. e. bounded in every direction) is complete.

Decomposition of linear maps:

- The representation matrix M of any linear map $\phi : \mathbb{K}^d \to \mathbb{K}^l$ can be written as $M = S^{-1}EPT^{-1}$, where S and T are invertible, E is the matrix of an embedding, and P is the matrix of an orthogonal projection.

Geometric Operations and Sops

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 sops in $\mathbb{K}^d$ and M be the invertible matrix of a linear map $\mathbb{R}^d \to \mathbb{R}^d$.
Sops behave nicely under the geometric operations.

$$\begin{array}{lrl}
\text{convex hull} & \operatorname{conv}(\mathbf{P}_1 \cup \mathbf{P}_2) & = \mathbf{P}\left(\begin{pmatrix} A_1 \\ \mathrm{O} \end{pmatrix}, \begin{pmatrix} A_1 & L_1 & \mathrm{O} & \mathbf{a}_1 \\ -A_2 & \mathrm{O} & L_2 & -\mathbf{a}_2 \end{pmatrix}, \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{0} \end{pmatrix}\right) \\
\text{Minkowski sum} & \mathbf{P}_1 + \mathbf{P}_2 & = \mathbf{P}\left(\begin{pmatrix} A_1 \\ \mathrm{O} \end{pmatrix}, \begin{pmatrix} A_1 & L_1 & \mathrm{O} \\ -A_2 & \mathrm{O} & L_2 \end{pmatrix}, \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{pmatrix}\right) \\
\text{intersection} & \mathbf{P}_1 \cap \mathbf{P}_2 & = \mathbf{P}\left(\begin{pmatrix} A_1 \\ A_2 \end{pmatrix}, \begin{pmatrix} L_1 & \mathrm{O} \\ \mathrm{O} & L_2 \end{pmatrix}, \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{pmatrix}\right)
\end{array}$$

Note: For the convex hull the sops have to be complete.

Geometric Operations and Sops

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 sops in $\mathbb{K}^d$ and M be the invertible matrix of a linear map $\mathbb{R}^d \to \mathbb{R}^d$.
Sops behave nicely under the geometric operations.

$$
\begin{array}{lrl}
\text{automorphism} & M(\mathbf{P}_1) & = \mathbf{P}(A_1 M^{-1}, L_1, \mathbf{a}_1) \\
\text{embedding} & \text{embed}_{d+l}(\mathbf{P}_1) & = \mathbf{P}\left(\begin{pmatrix} A_1 & O \\ O & I_k \\ O & -I_k \end{pmatrix}, \begin{pmatrix} L_1 \\ O \\ O \end{pmatrix}, \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}\right) \\
\text{projection} & \text{proj}_{d-k}(\mathbf{P}_1) & = \mathbf{P}(A, L, \mathbf{a}_1)
\end{array}
$$

Note: For the projection the matrices A and L are uniquely determined by the stipulation $(A\ L) = (A_1\ L_1)$ and the demand that A has $d - k$ columns.

Properties of Sops I

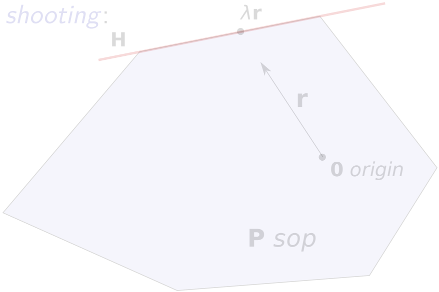
Further geometrical operations are possible: cylindrification, shadows, etc.

Sops benefit from the underlying $\mathcal{H}$-representation and LP, i. e.

- support function of an sop (and hence template polyhedra)
- redundancy removal applicable
- relative interior points
- polar polyhedra

These techniques can be combined to *ray shooting*:

- *Given* a non-empty sop $\mathbf{P} = \mathbf{P}(A, L, \mathbf{a})$ which contains $\mathbf{0}$ as a relative interior point
- and a ray $\mathbf{r}$.
- *Find* a scalar λ such that $\lambda\mathbf{r}$ is a boundary point of $\mathbf{P}$, and
- a supporting half-space H of $\mathbf{P}$ in the boundary point $\lambda\mathbf{r}$



Properties of Sops I

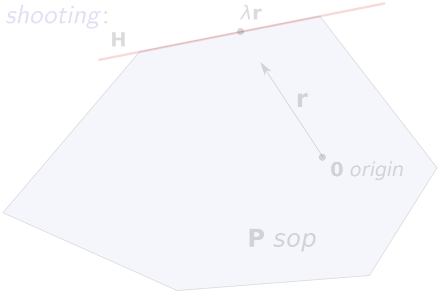
Further geometrical operations are possible: cylindrification, shadows, etc.

Sops benefit from the underlying $\mathcal{H}$-representation and LP, i. e.

- support function of an sop (and hence template polyhedra)
- redundancy removal applicable
- relative interior points
- polar polyhedra

These techniques can be combined to *ray shooting*:

- *Given* a non-empty sop $\mathbf{P} = \mathbf{P}(A, L, \mathbf{a})$ which contains $\mathbf{0}$ as a relative interior point
- and a ray $\mathbf{r}$.
- *Find* a scalar λ such that $\lambda\mathbf{r}$ is a boundary point of $\mathbf{P}$, and
- a supporting half-space H of $\mathbf{P}$ in the boundary point $\lambda\mathbf{r}$



Properties of Sops I

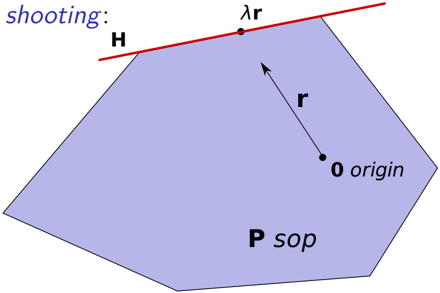
Further geometrical operations are possible: cylindrification, shadows, etc.

Sops benefit from the underlying $\mathcal{H}$-representation and LP, i. e.

- support function of an sop (and hence template polyhedra)
- redundancy removal applicable
- relative interior points
- polar polyhedra

These techniques can be combined to *ray shooting*:

- *Given* a non-empty sop $\mathbf{P} = \mathbf{P}(A, L, \mathbf{a})$ which contains $\mathbf{0}$ as a relative interior point
- and a ray $\mathbf{r}$.
- *Find* a scalar λ such that $\lambda\mathbf{r}$ is a boundary point of $\mathbf{P}$, and
- a supporting half-space H of $\mathbf{P}$ in the boundary point $\lambda\mathbf{r}$



Theorem (Ray Shooting)

Let $\mathbf{P} = \mathbf{P}(A, L, \mathbf{a})$ *be a non-empty and complete sop in* $\mathbb{K}^d$ *which contains the origin* $\mathbf{0}$ *as a relative interior point. Then the following LP is feasible for any vector* $\mathbf{r} \in \mathbb{K}^d$*:*

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

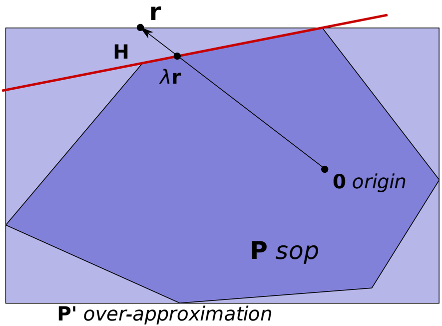
Further, the following statements hold:

1. *The linear program is unbounded if and only if* $\mathbf{r} \notin \mathrm{aff}(\mathbf{P})$.
2. *The optimal value equals zero if and only if* $\mathbf{P}$ *is unbounded in direction* $\mathbf{r}$,
3. *Otherwise, the optimal value is positive and for the optimal solution* $\mathbf{u}_0$ *we have: The half-space* $\mathbf{H} = \mathbf{H}(\mathbf{n}, 1)$, *with* $\mathbf{n} = A^T \mathbf{u}_0$, *is an optimal supporting half-space of* $\mathbf{P}$, *i. e.* $\frac{1}{\mathbf{r}^T A^T \mathbf{u}_0} \mathbf{r}$ *is a boundary point of* $\mathbf{H}$.

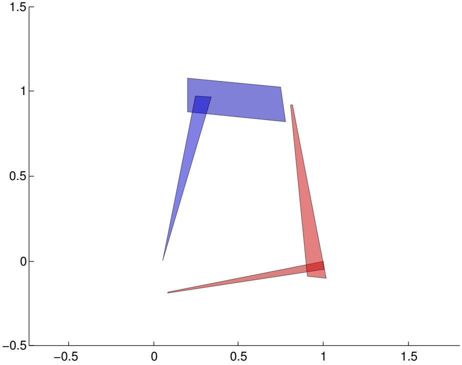
From Ray Shooting to Interpolation

Ray shooting allows to compute interpolations (even with *exact facets*) between a sop $\mathbf{P}$ and an over-approximating template polyhedron $\mathbf{P}'$.

1. Choose $\mathbf{r}$ as relative interior point of a facet of $\mathbf{P}'$
2. ray shooting in direction $\mathbf{r}$ provides an supporting half-space $\mathbf{H} = \mathbf{H}(\mathbf{n}, 1)$
3. Add $\mathbf{H}$ to representation of $\mathbf{P}'$ to improve over-approximation.

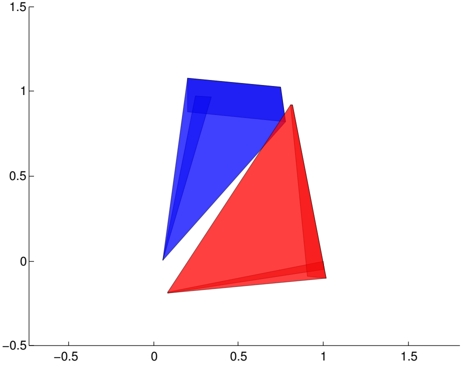


Example



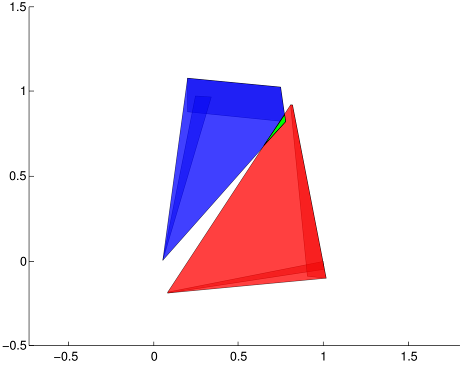
Given four polytopes, two blue and two red,

Example



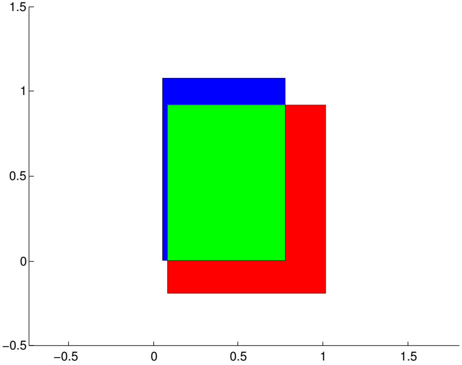
we compute the convex hulls of the red and the blues ones

Example



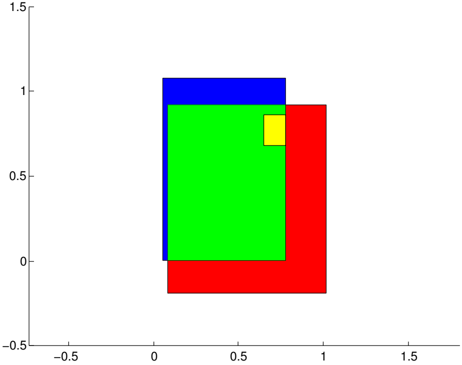
and the resulting intersection.

Example



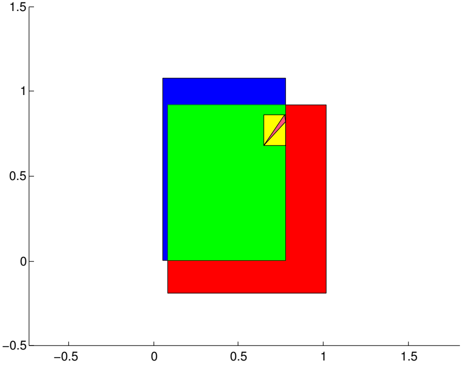
The green area shows the resulting intersection obtained by support functions and rectangular template polyhedra.

Example



The yellow area shows the resulting intersection obtained by sops and rectangular template polyhedra.

Example



Finally, the purple area is obtained by interpolation.

Overview

Hardness of performing the geometrical operation w. r. t. the given representation.

Representation	$M(\cdot)$	$\cdot + \cdot$	$\mathrm{conv}(\cdot \cup \cdot)$	$\cdot \cap \cdot$	$\cdot \subseteq \cdot$
$\mathcal{V}$-representation	$+$	$+$	$+$	$-$	$+$
$\mathcal{H}$-representation	$+^1$	$-$	$-$	$+$	$+$
support function	$+^2$	$+$	$+$	$-$	$-$
sop	$+$	$+$	$+$	$+$	$-$

^{1}for automorphism, ^{2}for endomorphism

Conclusions

- Sops are a new representation class of polyhedra, which is exact and efficient for most geometrical operations.
- Sops are evaluated with linear programming.

+ Sops enable us to compute the reachable sets up to a new degree of exactness.

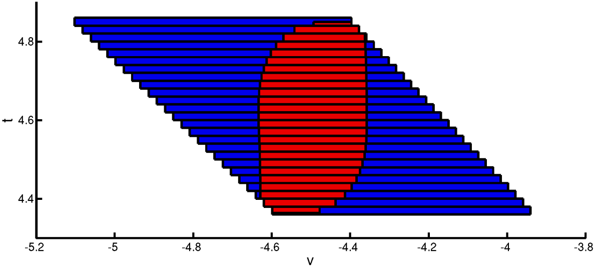
– Sops grow monotonic under these operations. There are different techniques for over-approximations / shrinking the size of sops: template polyhedra, ray shooting, and facet interpolation.

+ (Promising combination of Le Guernic & Girard's algorithm and a sop based algorithm)

What else can sops be used for?

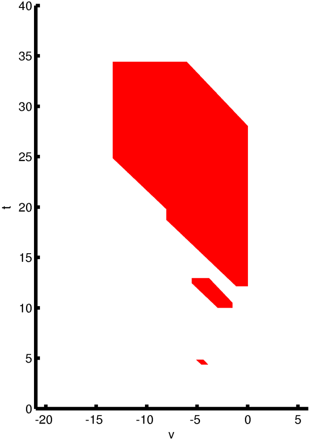
(My guess: program verification, motion planning,. . .)

Comparison of LGG-Algorithm and SOP-Algorithm



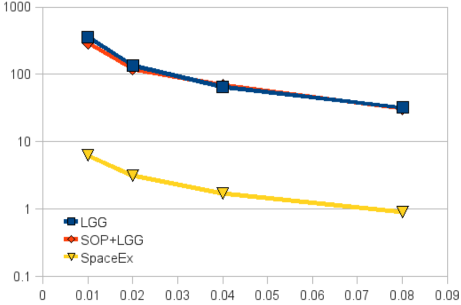
The figure shows the first intersection of a bouncing ball with a guard (the floor), where the dynamics of the model are given by $\dot{x} = v$, $\dot{v} = -1 \pm 0.05$, and $\dot{t} = 1$; the invariant is $x \geq 0$; and the guard is given by $x \leq 0$ and $v \leq 0$. The initial states are within the intervals $10 \leq x \leq 10.2$, $0 \leq v \leq 0.2$, and $t = 0$. For the computation we used the time step $\delta = 0.02$. The blue slices show the intersections computed by the LGG-algorithm using a rectangular template matrix. Each red slice shows a tight rectangular over-approximation of a sop representing a computed intersections of the SOP-algorithm. The representation matrices of these sops have a typical size of about 1500 rows and 750 columns with 6400 non-zero coefficients.

Comparison of LGG-Algorithm and SOP+LGG-Algorithm I



Same model as before, first four intersections are shown. Used time step is $\delta = 0.02$. The left figures shows the resulting intersections with the guard of LGG-algorithm. The blue areas of the right figures show the resulting intersections of the LGG-part of the SOP+LGG-algorithm and the red areas show over-approximations of the actual result.

Comparison of LGG-Algorithm and SOP+LGG-Algorithm II



time step δ	LGG	SOP+LGG	SpaceEx
0.08	32 sec	31 sec	0.91 sec
0.04	65 sec	69 sec	1.72 sec
0.02	135 sec	120 sec	3.13 sec
0.01	360 sec	293 sec	6.26 sec

We compared our experimental implementation of the SOP+LGG-algorithm against our implementation of the LGG-algorithm and the productive implementation of the LGG-algorithm in SpaceEx. For the computation we used different time steps δ and a rectangular template matrix.