# Symbolic Orthogonal Projections: A New Polyhedral Representation for Reachability Analysis of Hybrid Systems

Willem Hagemann

Dissertation zur Erlangung des Grades
des Doktors der Ingenieurwissenschaften
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

Saarbrücken
2014

| | |
|---|---|
| Tag des Kolloquiums | 2. Juli 2015 |
| Dekan | Univ.-Prof. Dr. Markus Bläser |
| Berichterstatter | Prof. Dr. Christoph Weidenbach |
| | Prof. Dr. Martin Fränzle |
| Vorsitzender des Prüfungsausschusses | Prof. Bernd Finkbeiner, Ph.D. |
| Akademischer Mitarbeiter | Dr. Uwe Waldmann |

# Abstract

This thesis deals with reachability analysis of linear hybrid systems. Special importance is given to the treatment of the necessary geometrical operations.

In the first part, we introduce a new representation class for convex polyhedra, the symbolic orthogonal projections (sops). A sop encodes a polyhedron as an orthogonal projection of a higher-dimensional polyhedron. This representation is treated purely symbolically, in the sense that the actual computation of the projection is avoided. We show that fundamental geometrical operations, like affine transformations, intersections, Minkowski sums, and convex hulls, can be performed by block matrix operations on the representation. Compared to traditional representations, like half-space representations, vertex representations, or representations by support functions, this is a unique feature of sops. Linear programming helps us to evaluate sops.

In the second part, we investigate the application of sops in reachability analysis of hybrid systems. It turns out that sops are well-suited for the discrete computations. Thereupon, we demonstrate also the applicability of sops for the continuous computation of reachability analysis. By a subtle parallel computation of a precise sop-based representation and a simplified representation, we tackle the problem of monotonic growing sizes of the sops. Additionally, we present experimental results which also show that sops allow an efficient and accurate computation of the reachable states.

# Zusammenfassung

Die vorliegende Arbeit beschäftigt sich mit der Erreichbarkeitsanalyse linearer hybrider Systeme, wobei wir besonderen Wert auf die Betrachtung der notwendigen geometrischen Operationen legen.

Im ersten Teil führen wir eine neue Darstellungsklasse für konvexe Polyeder ein: die symbolischen Orthogonalprojektionen (Sops). Basierend auf der Idee, einen Polyeder rein symbolisch als Orthogonalprojektion eines höherdimensionalen Polyeders zu beschreiben, zeigen wir, wie sich viele geometrische Operationen – u.a. affine Abbildungen, Schnitte, Minkowski-Summen und konvexe Hüllen – als einfache Blockmatrix-Operationen darstellen lassen. Diese Eigenschaft ist ein Alleinstellungsmerkmal der Sops gegenüber herkömmlichen Darstellungen wie der Halbraum-, der Vertexdarstellung oder der Repräsentation durch Supportfunktionen. Zudem lassen sich Sops durch lineare Optimierungen auswerten.

Im zweiten Teil setzen wir Sops zur Berechnung der erreichbaren Zustände hybrider Systeme ein. Die diskreten Zustandsübergänge lassen sich exakt durch Sops darstellen. Danach zeigen wir, dass sich Sops auch zur Berechnung der kontinuierlichen Entwicklung des hybriden Systems eignen. Durch paralleles Berechnen einer präzisen, sop-basierten Darstellung sowie einer vereinfachten Darstellung lösen wir das Problem der stetigen Zunahme der Darstellungsgröße der Sops. Zudem zeigen wir auch anhand von experimentellen Ergebnissen, dass sich Sops zur effizienten und präzisen Berechnung der erreichbaren Zustände eignen.

# Acknowledgments

First and foremost, I thank Uwe Waldmann, who supported me throughout the different stages of my research. He put a lot of time into the review of my results. Nevertheless, he managed to give me a perfectly balanced mixture of guidance and freedom. Without this, the thesis would probably not have been possible.

I thank Christoph Weidenbach and my colleagues in the Automation of Logics group at the Max Planck Institute for Informatics. Thanks to them I had a pleasant time, and I am looking forward to each stay in Saarbrücken.

I thank Werner Damm, who gave me the opportunity to resume my research in Oldenburg. I thank Boris Wirtz and Astrid Rakow for the warm welcome in the new office.

I thank Martin Fränzle and Bernd Finkbeiner for joining my thesis committee, as well as for interesting discussions and helpful comments.

I would like to thank the Deutsche Forschungsgemeinschaft for founding the AVACS project. It is inspiring to be a part of a transregional research project. In particular, I am thankful to be a member of the FOMC team. The joint meetings in Frankfurt are always a source of new insights and ideas.

Several persons directly influenced the content of my thesis, upon others there are Ernst Althaus and Eike Möhlmann, the latter caused me to develop a lot of features in SoapBox. I am grateful to all of them.

This work has been partly supported by the German Transregional Collaborative Research Center SFB/TR 14 AVACS.

# Contents

# List of Illustrations

## Figures

## Tables

# 1. Introduction

## 1.1. Structure of the Thesis

This thesis consists of two parts. The first part deals with convex polyhedra and could be read independently. The second part deals with the reachability analysis of hybrid systems and uses methods which heavily depend on the geometrical insights gained in the first part. Both subjects, convex polyhedra and hybrid systems, have been studied intensively in several works. Convex polyhedra have already been analyzed in the ancient world: Euclid investigated the Platonic solids circa 300 BC (Heath, 1956). The study of hybrid systems evolved in recent times. Therefore, the chosen arrangement reflects the history of both subjects. However, to provide a better motivation, we will tell the story the other way round in this introduction.

## 1.2. Reachability Analysis of Hybrid Systems

The goal of this section is to introduce the reachability analysis of hybrid systems. A hybrid system describes the behavior of a cyber-physical system evolving according to continuous and discrete laws. For example, a simple hybrid system can describe the movement of a mass point according to Newton's continuous laws of motion, until it hits a wall where it is instantaneously reflected by the discrete law of an elastic collision. We shall give a mathematical definition of a hybrid system that subsumes the classical definition of hybrid automata as it has been given by Alur et al. (1993). To emphasize the continuous behavior of a hybrid system, our definition of a hybrid system is split into two parts. Firstly, we define continuous dynamical systems whose behavior is purely continuous. Then we define hybrid systems as systems consisting of continuous dynamical systems that are connected by discrete transitions.

### 1.2.1. Continuous Dynamical System

**Definition 1.1 (Mathematical Model of a Continuous Dynamical System)**
A continuous dynamical system $C$ is a tuple

$$C = (\mathit{Var}, \mathit{Flow})$$

consisting of the following components:

- The finite set $\mathit{Var}$ of *variables* of the system with $|\mathit{Var}| = d$. We assume that for each variable there exists an injective mapping $\iota$ from the set of its values to $\mathbb{R}$. Hence, after fixing an ordering of the variables, every valuation over $\mathit{Var}$ can be uniquely represented as a point $\mathbf{x} \in \mathbb{R}^d$ where the $i$th component of $\mathbf{x}$ represents the value of the $i$th variable. Every valuation over *Var* is called a *state* of the continuous dynamical system. The set of all states is called the *state space* of $C$, denoted by $\mathcal{S}(C)$. For simplicity we shall refer to any $\mathbf{x} \in \mathbb{R}^d$ as a state of the continuous dynamical system. Accordingly, we will often identify $\mathcal{S}(C)$ with $\mathbb{R}^d$.

- The evolution of $C$ over time is described by the relation *Flow*. *Flow* assigns continuous changes to the states. We assume that continuous changes can be represented by differentials, i. e., by providing the linear part of the change at each state. Furthermore, we assume that $\iota$ uniquely maps the linear part to $\mathbb{R}^d$.

  Hence, *Flow* is formally described as a subset of $\mathbb{R}^d \times \mathbb{R}^d$ where each pair $(\mathbf{y}, \mathbf{x})$ of *Flow* encodes a state $\mathbf{x}$ and the differential $\mathbf{y}\,\mathrm{d}t$. A function $\mathbf{y}(t)$ describes a valid evolution of $C$ if for each $t$ in the domain of $\mathbf{y}$ it holds that $(\dot{\mathbf{y}}(t), \mathbf{y}(t)) \in \mathit{Flow}$.

  The system exhibits only those states for which a differential exists. Hence, $C$ admits only states that are located in the invariant $\mathbf{I}$, which is defined by

$$\mathbf{I} = \{\mathbf{x} \mid \exists \dot{\mathbf{x}}\colon\ (\dot{\mathbf{x}}, \mathbf{x}) \in \mathit{Flow}\}\,. \qquad \blacksquare$$

Useful instruments to describe the evolution of a continuous dynamical systems are trajectories.

**Definition 1.2 (Trajectory of a Continuous Dynamical System)**
Let $C$ be a continuous dynamical system and $\mathbf{x}_0$ be a state of $C$. A trajectory $\mathbf{y} = \mathbf{y}(t)$ is a function $\mathbf{y} : [0, T] \to \mathcal{S}(C)$ such that for each $t \in [0, T]$ the derivative $\dot{\mathbf{y}}(t)$ exists and $(\dot{\mathbf{y}}(t), \mathbf{y}(t)) \in \mathit{Flow}$.

The notion $\mathbf{y}(t, \mathbf{x}_0)$ shall denote a trajectory $\mathbf{y}(t)$ for which $\mathbf{y}(0) = \mathbf{x}_0$ holds. In this case we call $\mathbf{x}_0$ the *initial state* of $\mathbf{y}$. $\blacksquare$

Note that we consider *time-invariant* continuous dynamical systems only: *Flow* does not change over time and, hence, for any $T$ and any trajectory $\mathbf{y}$ of the continuous dynamical system $C$ it holds that $\mathbf{y}'(t) = \mathbf{y}(t + T)$ is also a valid trajectory of $C$. An *autonomous* continuous dynamical system $C$ is a continuous dynamical system where *Flow* satisfies the additional requirement that $(\mathbf{y}, \mathbf{x}) \in \mathit{Flow}$ and $(\mathbf{y}', \mathbf{x}) \in \mathit{Flow}$ implies $\mathbf{y} = \mathbf{y}'$ for all $\mathbf{x}$, $\mathbf{y}$, $\mathbf{y}'$. With other words, *Flow* is given as the set $\{(\mathbf{f}(\mathbf{x}), \mathbf{x})\}$ where $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$ is a vector field. A useful criterion for the existence of trajectories is the Peano existence theorem, which ensures the existence of solutions of the initial value problem

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \quad \mathbf{x}(0) = \mathbf{x}_0$$

if the vector field $\mathbf{f}$ is continuous. Moreover, if the vector field $\mathbf{f}$ is also Lipschitz continuous, i. e., there exists a real constant $L \geq 0$ such that for all $\mathbf{x}_1$ and $\mathbf{x}_2$ it holds $||\mathbf{f}(\mathbf{x}_1) - \mathbf{f}(\mathbf{x}_2)|| \leq L\,||\mathbf{x}_1 - \mathbf{x}_2||$ for a norm $||\cdot||$, then the Picard-Lindelöf theorem ensures the uniqueness of the solution of the initial value problem. For the Peano existence theorem and the Picard-Lindelöf theorem see e. g. Heuser (1995).

The reason for the rather general definition of the *Flow*-relation is to capture also the so-called *non-autonomous* systems. These are systems that admit non-deterministic external input functions $\mathbf{u}(t)$ such that *Flow* is characterized by the differential equation

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{u}.$$

Here, the non-determinism of the input functions $\mathbf{u}(t)$ carries over to the continuous dynamical system.

### 1.2.2. Linear Systems

There are various different mathematical description of the system's behavior that are embraced by the definition of continuous dynamical systems. In this thesis we restrict our attention to *linear systems*.

**Definition 1.3 (Linear System)**
A linear system is a continuous dynamical system where the continuous flow of each mode is specified by a linear differential equation of the form

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + \mathbf{u}(t), \quad \mathbf{x}(t) \in \mathbf{I},\ \mathbf{u}(t) \in \mathbf{U} \text{ for all } t. \qquad \blacksquare$$

**Example 1.4 (Approach Velocity Controller)**
As an example for a linear system we discuss the following model of an approach velocity

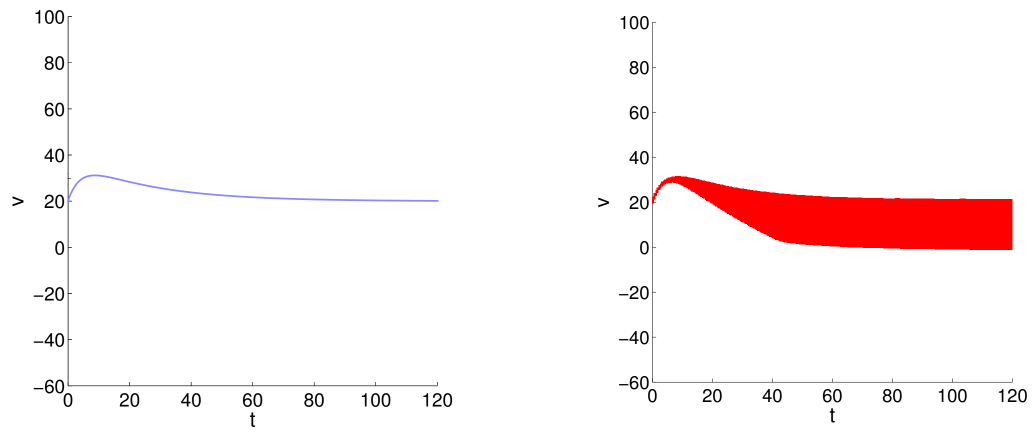


Figure 1.1.: Behavior of the AVC

controller (AVC). We consider two cars on a single lane. Both cars have distance $d$. The approach velocity controller controls the velocity $v$ of the following car in order to establish a desired distance $d_{\text{des}} = 50$ to the leading car, which has velocity $v_a$. The evolution of the distance depends on the velocities of the two cars only and is given by $\dot{d} = v_a - v$. The velocity of the leading car evolves according to a non-deterministic input function that can take values between $-0.5$ and $0.5$, and we postulate that $v_a$

can only take values in the interval $[0, 20]$. Finally, the control law for $v$ is specified as $\dot{v} = 0.29(v_a - v) + 0.01(d - d_{\text{des}})$. Additionally, we explicitly model the time $t$. Hence, *Flow* is the set of all tuples $(\dot{v_a}, \dot{v}, \dot{d}, \dot{t}, v_a, v, d, t)$ satisfying the relation

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

Figure 1.1 shows the evolution of $v$ over time $t$ for the initial values $v = v_a = 20$ and $d = 450$. The left figure shows a sample trajectory generated by SpaceEx (Frehse et al., 2011). The right figure shows the area in which every sample trajectory must lie, and has been generated by SoapBox. ■

### 1.2.3. Hybrid System

Continuous dynamical systems allow continuous changes of the variables only; a hybrid system allows both, continuous updates, also called *flows*, and discrete updates of the variables, which are called *jumps*.

**Definition 1.5 (Mathematical Model of a Hybrid System)**
A hybrid system $H$ is a tuple

$$H = (\mathit{Var}, \mathit{Mod}, \mathit{Trans})$$

consisting of the following components:

- *Var* is a finite set of variables of the hybrid system with $|\mathit{Var}| = d$. As for the continuous dynamical systems, we assume that each valuation of the variables corresponds to a point $\mathbf{x} \in \mathbb{R}^d$.
- *Mod* is a finite set of *modes* of the system. Each mode is a continuous dynamical system. The variables of each mode form a subset of *Var*, and its *Flow* relation describes the continuous evolution of these variables. We assume there is an injective mapping $\mathit{Mod} \to \mathbb{N}$ that specifies the modes by natural numbers.
- A *state* of the system consists of a valuation of the variables and an additional value for the mode. Hence, each state is uniquely determined by the pair $(\mathbf{x}, m) \in \mathbb{R}^d \times \mathbb{N}$, and we will often identify the state space $\mathcal{S}(H)$ with $\mathbb{R}^d \times \mathbb{N}$.
- The instantaneous jumps of the system are described by the relation *Trans*. *Trans* relates the states before an instantaneous jump with the states after the jump. Hence, *Trans* is formally described as a subset of $(\mathbb{R}^d \times \mathbb{N}) \times (\mathbb{R}^d \times \mathbb{N})$. Any tuple $\tau = (\mathbf{x}, m, \mathbf{x}', m') \in \mathit{Trans}$ signifies that the state $(\mathbf{x}, m)$ may instantaneously jump to the state $(\mathbf{x}', m')$. ■

Due to the instantaneous jumps in $H$, a formal definition of trajectories is slightly more elaborate.

**Definition 1.6 (Trajectory of a Hybrid System)**
Let $H$ be a hybrid system. A trajectory of $H$ is a sequence $(\mathbf{y}_n, \mathbf{x}_n, m_n, T_n)_{n=0,\dots N}$ that satisfies the following conditions

(i) for each $n = 0 \dots N$ the function $\mathbf{y}_n(t) : [0, T_n] \to \mathcal{S}(m_n)$ is a trajectory of the mode $m_n$ with the initial state $\mathbf{x}_n$,

(ii) for each $n = 1 \dots N$ the initial state $\mathbf{x}_n$ of the trajectory $\mathbf{y}_n$ is related to the end state of the preceding trajectory by *Trans*, formally:

$$(\mathbf{y}_{n-1}(T_{n-1}), m_{n-1}, \mathbf{x}_n, m_n) \in \mathit{Trans}.$$

We call $(\mathbf{x}_0, m_0)$ the initial state of $\mathbf{y}(t)$. ■

### 1.2.4. Linear Hybrid Systems

In practice discrete transitions are often specified by a finite number of *guarded assignments*: Given a source mode $m$ and a target mode $m'$, a guarded assignment $\gamma = (\mathbf{G}, m, \mathbf{f}, m')$ specifies a subset $\mathbf{G} \subseteq \mathcal{S}(m)$, called the *guard*, and an assignment $\mathbf{f} : \mathbf{G} \to \mathcal{S}(m')$, called the *update function*, that assigns states in $\mathcal{S}(m')$ to the states of $\mathbf{G}$. Hence, a discrete transition $\tau = (\mathbf{x}, m, \mathbf{x}', m')$ is defined by the guarded assignment $\gamma$ if and only if $\mathbf{x} \in \mathbf{G}$ and $\mathbf{x}' = \mathbf{f}(\mathbf{x})$. If we further restrict $\mathbf{f}$ to be an affine function, this finally yields a *linear hybrid system*, which is the main object of investigation in the second part of this thesis.

**Definition 1.7 (Linear Hybrid System)**
A linear hybrid system is a hybrid system where

(i) the continuous flow of each mode is specified by a linear differential equation of the form

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + \mathbf{u}(t), \quad \mathbf{x}(t) \in \mathbf{I},\ \mathbf{u}(t) \in \mathbf{U} \text{ for all } t,$$

(ii) the discrete transitions are specified as guarded assignments of the form $\gamma = (\mathbf{G}, m, \mathbf{f}, m')$, and the update function $\mathbf{f}$ is an affine mapping. ■

**Example 1.8 (Bouncing Ball)**
The bouncing ball model is a well-known example of a hybrid system. We consider a point mass, the ball, at height $x$ and velocity $v$. The ball is exposed to the gravitational force $\dot{v} = -g$ where $g$ is the acceleration due to gravity. When the ball encounters the ground, located at height $x = 0$, it is instantaneously repulsed due to a partial inelastic collision $v' = -c_r v$ where $v'$ is the velocity after the collision, $v$ is the velocity before the collision, and $c_r \in [0, 1]$ is the constant coefficient of restitution.

The continuous behavior is modeled as the dynamical system $C = (\{x, v, t\}, \mathit{Flow})$ where *Flow* is defined as follows: $(\dot{x}, \dot{v}, \dot{t}, x, v, t) \in \mathit{Flow}$ if and only if

$$\begin{aligned} &\dot{x} = v, && \dot{v} = -g, && \dot{t} = 1, \\ &x \geq 0. \end{aligned}$$

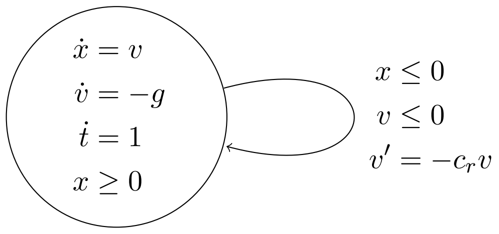


Figure 1.2.: Hybrid System Model of a Bouncing Ball

The invariant $x \geq 0$ reflects the fact that the ball cannot penetrate the ground. Additionally, we explicitly model the time $t$.

The hybrid system is given by $H = (\{x, v, t\}, \{C\}, \mathit{Trans})$ where *Trans* is defined by the guarded assignment $\gamma = (\{x \leq 0, v \leq 0\}, C, f : v \mapsto -c_r v, C)$, see also Figure 1.2. ■

### 1.2.5. Reachability Analysis

Given a hybrid system $H$ and a designated set *Init* of initial states, the main goal of reachability analysis is to compute the reachable states of $H$ from *Init*: We would like to compute all states that are located on any trajectory emanating from an initial state $\mathbf{x}_0$ in *Init*. Reachability analysis is an important technique for the safety analysis of hybrid systems, e. g. to verify that certain states can or cannot be reached. The states for which we want to decide the reachability are often called *bad states*, and the set of all bad states is called *Unsafe*. The question whether *Unsafe* is reachable from *Init* is known as the *reachability problem for hybrid systems*, and it is known to be undecidable in general (Alur et al., 1995; Henzinger et al., 1995).

The basic idea of reachability analysis is to compute the set of reachable states by a fix-point iteration. The computation starts with the initial states and subsequently adds those states that are reachable by a discrete transition or by a continuous evolution until all added states are already contained in the collection. A similar approach is to start with the bad states and to subsequently add the continuous and discrete *preimages* until either an initial state or a fix-point is reached. The continuous and discrete preimages of a state $(\mathbf{x}, m)$ consist of all states from which $(\mathbf{x}, m)$ is reachable via a continuous evolution or a discrete transition, respectively. Both approaches can be used for safety analysis. The latter approach is called *backward* reachability analysis, and to avoid any possibility of confusion, the former is sometimes attributed as *forward* reachability analysis. This thesis deals with forward reachability analysis, backward reachability is addressed only marginally.

The fix-point computation is an algorithmic realization of the following mathematical construction: Let us define two *post operators*, $\mathrm{post}_d$ and $\mathrm{post}_c$. Both operators map from $\mathcal{S}(H)$ to $\mathrm{Pow}(\mathcal{S}(H))$. The *discrete post operator* $\mathrm{post}_d(\mathbf{x}, m)$ is defined as

$$\mathrm{post}_d : (\mathbf{x}, m) \mapsto \{(\mathbf{x}', m') \,|\, (\mathbf{x}, m, \mathbf{x}', m') \in \mathit{Trans}\},$$

and the *continuous post operator* $\mathrm{post}_c(\mathbf{x}, m)$ is defined as

$$\mathrm{post}_c : (\mathbf{x}, m) \mapsto \\ \{(\mathbf{x}', m) \,|\, \mathbf{x}' = \mathbf{y}(t, \mathbf{x}) \text{ for a trajectory } \mathbf{y} : [0, T] \to \mathcal{S}(m) \text{ and } \ t \in [0, T]\}\,.$$

We extend $\mathrm{post}_c$ and $\mathrm{post}_d$ to operate on sets of states by

$$\mathrm{post}_c(\mathbf{X}) = \bigcup_{(\mathbf{x},m)\in\mathbf{X}} \mathrm{post}_c(\mathbf{x}, m), \qquad \mathrm{post}_d(\mathbf{X}) = \bigcup_{(\mathbf{x},m)\in\mathbf{X}} \mathrm{post}_d(\mathbf{x}, m).$$

Let $\mathbf{R}_0$ be the set of initial states, let

$$\mathbf{R}_{k+1} = \mathbf{R}_k \cup \mathrm{post}_d(\mathbf{R}_k) \cup \mathrm{post}_c(\mathbf{R}_k), \tag{1.1}$$

and let $\mathbf{R}_\infty = \bigcup_{k\in\mathbb{N}} \mathbf{R}_k$. Then $\mathbf{R}_\infty$ is the set of all reachable states of $H$. $\mathbf{R}_\infty$ is the least set that includes $\mathbf{R}_0$ and that is closed under the discrete and continuous post operator. Successful termination of the fix-point computation means that there is some $K$ for which the sequence already stabilizes, i. e., $\mathbf{R}_K = \mathbf{R}_{K+1} = \cdots = \mathbf{R}_\infty$. Termination of the computation cannot be guaranteed in general, and this is one factor which contributes to the undecidability of the reachability problem of hybrid systems. Notwithstanding this principal limitation, the iterative approach allows us to gain useful information on the behavior of a system. For example, when *Unsafe* is reachable, i. e., $\mathit{Unsafe} \cap \mathbf{R}_\infty \neq \emptyset$, then there exists a least index $K$ for which $\mathit{Unsafe} \cap \mathbf{R}_K \neq \emptyset$ holds. Hence, the fix-point computation provides a *semi-decision* procedure for the reachability problem. Furthermore, even a fix-point iteration of bounded depth may give satisfactory insights into the system's evolution for a bounded period.

Alas, there are several restrictions in practice which prevent an efficient computation of the sets $\mathbf{R}_k$. The reachability analysis has to deal with infinitely many different states. Clearly, such an amount of states cannot be handled explicitly. Hence, we have to find a suitable symbolic representation of the set of states. Ideally, a *symbolic state representation* is (i) capable to specify a wide range of sets of states, (ii) allows us to express various set operations, like $\mathrm{post}_d$, $\mathrm{post}_c$, intersections, or unions, and (iii) has efficient procedures to decide properties like emptiness or subset relations. Yet, a symbolic state representation that satisfies all these requirements has not been found.

## 1.3. Polyhedra

Current approaches in the area of reachability analysis investigate different symbolic state representations and restrict the class of admissible hybrid system (Girard, 2005; Chen et al., 2013; Frehse et al., 2011) such that at least over-approximations of $\mathrm{post}_c$, $\mathrm{post}_d$, and other set operations can be computed efficiently. We follow the lead and restrict our attention to polyhedral set representations.

Closed convex polyhedral sets, we shall use the term polyhedra, are a useful data structure to describe the linear inequality relations of the state variables of a system. Together with convexity preserving manipulations, like affine transformations, convex hull, Minkowski sum, and intersections, they often form the theoretical basis of system analysis, as is evident in program verification (Cousot and Halbwachs, 1978), hybrid model checking (Le Guernic and Girard, 2009; Hagemann, 2014a), or motion planing. In practice, system analysis hinges on an efficient computational realization of polyhedra-based algorithms.

### 1.3.1. $\mathcal{H}$-Polyhedra and $\mathcal{V}$-Polyhedra

In the literature typically two different representation of closed convex polyhedra are introduced, the *$\mathcal{H}$-representation*, where a polyhedron $\mathbf{P}$ is given as an intersection of finitely many closed half-spaces

$$\mathbf{P} = \mathbf{P}\left(A, \mathbf{a}\right) = \left\{\mathbf{x} \,|\, A\mathbf{x} \leq \mathbf{a}\right\},$$

and the *$\mathcal{V}$-representation*, where $\mathbf{P}$ is given as the Minkowski sum

$$\mathbf{P} = \operatorname{cone}(\mathbf{U}) + \operatorname{conv}(\mathbf{V})$$

of $\operatorname{cone}(\mathbf{U})$ and $\operatorname{conv}(\mathbf{V})$, which are sets generated by conical and convex combinations of vectors from the finite sets $\mathbf{U}$ and $\mathbf{V}$. Under the assumption that the respective representation does not contain any redundant information, the $\mathcal{H}$-representation specifies the facets of the polyhedron and the set $\mathbf{U}$ specifies all rays and $\mathbf{V}$ all vertices of the polyhedron. According to the given representation, we call a polyhedron either an $\mathcal{H}$- or a $\mathcal{V}$-polyhedron. The Minkowski-Weyl Theorem states that both representations are theoretically equivalent. But, unfortunately, the representations differ algorithmically (Ziegler, 1995): While the $\mathcal{V}$-representation of the convex hull or the Minkowski sum of $\mathcal{V}$-polyhedra is easy to compute, the enumeration of the facets is NP-hard for the convex hull and the Minkowski sum of $\mathcal{H}$-polyhedra; and the contrary relation holds for the intersection, see Table 1.1. The problem of converting between both representations is known as the *vertex enumeration* and *facet enumeration problem*, respectively, and its complexity is still open (Boros et al., 2011; Khachiyan et al., 2008). Note that, since the output size of geometrical operations, including conversions, can clearly be exponential in the input size[1], we typically measure the complexity in terms of *output-sensitive* algorithms in this area.

In their seminal article, Cousot and Halbwachs (1978) discussed the computation of linear relations among variables of a program and used both kinds of polyhedral representation. They faced the problem of diverging algorithmic performance of geometrical manipulations in the different representations. Finally, they propose the usage of both representations since “it appears that it is difficult to simplify one representation without knowing the other one and neither can be used efficiently without simpli[fi]cations”. Nowadays, the usage of specialized polyhedral libraries is quite common in the field of program analysis (Sankaranarayanan et al., 2006). Internally, these libraries can utilize the $\mathcal{V}$- or the $\mathcal{H}$-representation, and the appropriate representation is chosen accordingly to the requested manipulation. The conversion algorithm between both representations

[1] This effect is known as *vertex explosion* or *facet explosion*, respectively.

is based on the double description method invented by Motzkin et al. (1953), see e. g. Bagnara et al. (2002). Still, these polyhedral libraries suffer from the complexity in time and space of the conversion algorithm (Sankaranarayanan et al., 2006).

### 1.3.2. Simplified Operations and Alternative Representation Classes

Another approach is the usage of simplified manipulations instead of the exact geometrical operation, e. g. using the *weak join* or the *inversion join* instead of the convex hull, see e. g. Sankaranarayanan et al. (2006). Clearly, this approach is accompanied by a loss of exactness.

Especially in the field of reachability analysis of hybrid automata, the usage of alternative representations has been investigated, for which at least some geometrical operations behave nicely. For example, Girard (2005) proposes the usage of zonotopes, which allow a very efficient computation of Minkowski sums and are an adequate data-structure to cope with linear systems whose dynamics are subject to bounded disturbances or inputs. It turns out that alternative data-structures are highly specialized for certain manipulations while other operations become extremely hard or even impossible, e. g. the intersection of zonotopes is in general not a zonotope. Typically, the usage of specialized representations is accompanied with strong restrictions on the eligible shapes.

### 1.3.3. Support Functions

Le Guernic and Girard (2009) propose a reachability analysis based on *template polyhedra* and *support functions*. Support functions can be seen as generalizations of linear programs. 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

$$h_{\mathbf{S}}(\mathbf{n}) = \sup_{\mathbf{x} \in \mathbf{S}} \mathbf{n}^T \mathbf{x}.$$

If $\mathbf{S}$ is a closed convex polyhedron, i. e., $\mathbf{S} = \mathbf{P}(A, \mathbf{a})$, then the value $h_{\mathbf{S}}(\mathbf{n})$ coincides with the optimal value of the linear program

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

Support functions behave nicely under most geometrical operations: In detail, for any two compact convex sets $\mathbf{P}$ and $\mathbf{Q}$ in $\mathbb{R}^d$ the following easily computable equations hold (Rockafellar and Wets, 1998):

- for any $(d \times d)$-transformation matrix $M$ the support function of the transformed set $M\mathbf{P}$ is given by

$$h_{M\mathbf{P}}(\mathbf{n}) = h_{\mathbf{P}}(M^T \mathbf{n}),$$

- the support function of the Minkowski sum is given by

$$h_{\mathbf{P}+\mathbf{Q}}(\mathbf{n}) = h_{\mathbf{P}}(\mathbf{n}) + h_{\mathbf{Q}}(\mathbf{n}),$$

- and the support function of the convex hull is given by

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

Contrariwise, the support function of the intersection is not easily computable:

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

### 1.3.4. Template Polyhedra

Template polyhedra are $\mathcal{H}$-polyhedra of the form $\mathbf{P}\left(A_{\text{fix}}, \mathbf{a}\right)$ with a fixed representation matrix $A_{\text{fix}}$ and varying constant terms $\mathbf{a}$. They play an important role in system analysis (Sankaranarayanan et al., 2008) since for many manipulations they are capable to render the exact extension of the resulting state sets in the directions given by some fixed template matrix $A_{\text{fix}}$: Given the support function $h_{\mathbf{S}}$ of some set $\mathbf{S} \in \mathbb{R}^d$ and the $(m \times d)$ matrix $A_{\text{fix}}$, we easily obtain a closed convex over-approximation $\mathbf{P}\left(A_{\text{fix}}, \mathbf{a}_{\mathbf{S}}\right)$ of $\mathbf{S}$ by setting

$$\mathbf{a}_{\mathbf{S}} = \begin{pmatrix} h_{\mathbf{S}}(\mathbf{n}_1) \\ h_{\mathbf{S}}(\mathbf{n}_2) \\ \vdots \\ h_{\mathbf{S}}(\mathbf{n}_m) \end{pmatrix},$$

where the row vector $\mathbf{n}_i^T$ are the rows of the matrix $A_{\text{fix}}$. For example, it follows that

$$\mathbf{P}\left(A_{\text{fix}}, \mathbf{a}_1\right) \cap \mathbf{P}\left(A_{\text{fix}}, \mathbf{a}_2\right) = \mathbf{P}\left(A_{\text{fix}}, \min(\mathbf{a}_1, \mathbf{a}_2)\right)$$

and

$$\mathrm{conv}(\mathbf{P}\left(A_{\text{fix}}, \mathbf{a}_1\right) \cup \mathbf{P}\left(A_{\text{fix}}, \mathbf{a}_2\right)) \subseteq \mathbf{P}\left(A_{\text{fix}}, \max(\mathbf{a}_1, \mathbf{a}_2)\right).$$

Moreover, if all coefficients of $\mathbf{a}_1$ and $\mathbf{a}_2$ are tight upper bounds for the extension of the sets in the directions given by $A_{\text{fix}}$, then all coefficients of $\min(\mathbf{a}_1, \mathbf{a}_2)$ and $\max(\mathbf{a}_1, \mathbf{a}_2)$ are also tight upper bounds for the extension of the resulting set in the given directions.

### 1.3.5. Symbolic Orthogonal Projections

We will propose a new polyhedral representation, the symbolic orthogonal projections (sops). A sop encodes a polyhedron as an orthogonal projection of a higher dimensional $\mathcal{H}$-polyhedron. Fundamental geometrical operations, like affine transformations, intersections, Minkowski sums, and convex hulls, can be performed by block matrix operations on the representation. Due to the underlying $\mathcal{H}$-representation, linear programming helps us to evaluate sops.

### 1.3.6. Overview on Different Representations

The following table provides an overview on the hardness of performing linear transformations, Minkowski sums, closed convex hulls[2], intersections, and deciding subset relations on polyhedra in the respective representation. The tick indicates computability in (weakly) polynomial time. Otherwise, the enumeration problem is either NP-hard or its complexity is unknown, see Tiwary (2008).

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

[a]for automorphism, [b]for endomorphism

Table 1.1.: Hardness Results for Selected Operations on Different Representations

### 1.3.7. Linear Hybrid Systems with Polyhedral Set Representation

At the end of this introduction, we eventually return to hybrid systems and fix the symbolic state representation as it shall be presented in the second part of this thesis. We will discuss the reachability analysis of linear hybrid systems with polyhedral set representation:

**Definition 1.9 (Linear Hybrid System with Polyhedral Set Representation)** A linear hybrid system with polyhedral set representation is a linear hybrid system where

(i) the continuous flow of each mode is specified by a linear differential equation of the form

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + \mathbf{u}(t), \quad \mathbf{x}(t) \in \mathbf{I},\ \mathbf{u}(t) \in \mathbf{U} \text{ for all } t,$$

and $\mathbf{I}$ and $\mathbf{U}$ are given as polyhedra,

(ii) the discrete transitions are specified as guarded assignments of the form $\gamma = (\mathbf{G}, m, \mathbf{f}, m')$, the update function $\mathbf{f}$ is an affine mapping, and the guard $\mathbf{G}$ is a polyhedron. ■

To make reachability analysis accessible to practical computations, we shall only deal with finite sets of symbolic states, where a symbolic state $(\mathbf{X}, m)$ is a pair of a polyhedron $\mathbf{X}$ and a mode $m$. Accordingly, the sets *Init* and *Unsafe* shall be given as finite sets of symbolic states.

[2] Note that the convex hull of two closed sets is not necessarily closed, see also Example 3.23, and cannot be represented as a closed polyhedron. Hence, we have to restrict our investigation to the *closed* convex hull.

# Part I.

# Convex Polyhedra

# 2. Theory of Systems of Linear Inequalities

This chapter is a compilation of important theorems of the theory of systems of linear inequalities. Proofs and additional material can be found in almost any textbook on convex polyhedra, e. g. Ziegler (1995), or linear programming, e. g. Matoušek and Gärtner (2007); Schrijver (1986).

**Convention.** All results of this chapter are valid in any vector space $\mathbb{K}^d$ over an ordered field of finite dimension $d$. Matrices are denoted by upper case letters and vectors by bold lower case letters. A column vector whose coefficients are all zero is denoted by $\mathbf{0}$, transposed vectors or matrices are denoted by the superscript $^T$. The dimensions of objects are often not specified if the concrete value is not important. Nevertheless, a notation like $A\mathbf{x}$ implicitly implies that the number of columns of the matrix $A$ agrees with dimension of the column vector $\mathbf{x}$.

## 2.1. Solving Systems of Linear Inequalities

The Fourier-Motzkin elimination procedure is a method for solving system of linear inequalities similar to Gaussian elimination.

**Theorem 2.1 (Fourier-Motzkin Elimination)**
Let $A\mathbf{x} \leq \mathbf{a}$ be a system over $d \geq 1$ variables and $m$ non-strict inequalities. Then there exists a system $A'\mathbf{x}' \leq \mathbf{a}'$ over $d-1$ variables such that

(i) $A\mathbf{x} \leq \mathbf{a}$ has a solution if and only if $A'\mathbf{x}' \leq \mathbf{a}'$ has a solution,

(ii) the system $A'\mathbf{x}' \leq \mathbf{a}'$ may have as many as $\lfloor \frac{m^2}{4} \rfloor$ inequalities,

(iii) there exists an algorithm which generates the system $A'\mathbf{x} \leq \mathbf{a}'$ from the system $A\mathbf{x} \leq \mathbf{a}$. ■

From a geometrical perspective, the Fourier-Motzkin elimination procedure computes the orthogonal projection of the polyhedron $\mathbf{P}(A, \mathbf{a}) \subseteq \mathbb{K}^d$ onto $\mathbb{K}^{d-1}$.

Fourier-Motzkin elimination can be applied to verify the solvability of a system of linear inequalities: We use Fourier-Motzkin elimination to successively eliminate the variables $\mathbf{x}$ of the system $A\mathbf{x} \leq \mathbf{a}$ until we obtain an equisatisfiable system without variables, i. e., a system that consists of inequalities over constants only. Clearly, the solvability of such a system can be read off immediately. Moreover, in case of solvability, any solution $\mathbf{x}$ of the system $A\mathbf{x} \leq \mathbf{a}$ can be found by a backward substitution through the successively emerged systems of linear inequalities.

We should note that there also exists a variant of the Fourier-Motzkin elimination procedure for systems with mixed strict and non-strict inequalities.

## 2.2. Farkas' Lemma

**Theorem 2.2 (Motzkin's Transposition Theorem)**
For any system

$$A\mathbf{x} \leq \mathbf{a},\, B\mathbf{x} < \mathbf{b} \tag{2.1}$$

of strict and non-strict linear inequalities the following conditions are equivalent:

1. there exists a solution $\mathbf{x}$ of $A\mathbf{x} \leq \mathbf{a}$, $B\mathbf{x} < \mathbf{b}$,
2. for all $\mathbf{u} \geq \mathbf{0}$ and $\mathbf{v} \geq \mathbf{0}$ it holds that
   a) if $A^T\mathbf{u} + B^T\mathbf{v} = \mathbf{0}$, then $\mathbf{a}^T\mathbf{u} + \mathbf{b}^T\mathbf{v} \geq 0$ and
   b) if $A^T\mathbf{u} + B^T\mathbf{v} = \mathbf{0}$ and $\mathbf{v} \neq \mathbf{0}$, then $\mathbf{a}^T\mathbf{u} + \mathbf{b}^T\mathbf{v} > 0$. ■

As a direct consequence of the Transposition Theorem, we obtain Farkas' Lemma.

**Theorem 2.3 (Farkas' Lemma)**
For any system $A\mathbf{x} \leq \mathbf{a}$ of linear inequalities exactly one of the following alternatives holds:

1. there exists some $\mathbf{x}$ such that $A\mathbf{x} \leq \mathbf{a}$,
2. there exists some $\mathbf{u} \geq \mathbf{0}$ with $A^T\mathbf{u} = \mathbf{0}$ and $\mathbf{a}^T\mathbf{u} < 0$. ■

## 2.3. Linear Programs

**Definition 2.4 (Linear Program)**
Let $A\mathbf{x} \leq \mathbf{a}$ be a system of non-strict linear inequalities over $\mathbb{K}^d$. Further, let $\mathbf{n} \in \mathbb{K}^d$. A *linear program* (in canonical form) is the task:

$$\text{maximize } \mathbf{n}^T\mathbf{x} \text{ subject to } A\mathbf{x} \leq \mathbf{a}. \tag{2.2}$$

■

We shall note that any minimizing or maximizing task over a system of non-strict inequalities and arbitrary equalities can be transformed by simple equivalent transformations to the canonical form (2.2) of a linear program. Hence, we call all these variants a linear program.

The following wording is often used in the context of linear programs. Given a linear program (2.2), any vector $\mathbf{x}$ satisfying the system $A\mathbf{x} \leq \mathbf{a}$ is called a *feasible solution*. If at least one feasible solution exists, then the linear program is said to be *feasible*. Otherwise, if the linear program has no feasible solution, it is *infeasible*. An *optimal solution* $\mathbf{x}^*$ is a feasible solution that fulfills the maximality (minimality) condition. Then $\mathbf{n}^T\mathbf{x}^*$ is the *optimal value* of the linear program. A feasible linear program does not necessarily have an optimal value. In this case the linear program is *unbounded*.

Theoretically, a linear program can be solved by Fourier-Motzkin elimination. But, due to enormous growth of the number of linear inequalities in every elimination step, Fourier-Motzkin elimination is in practice not suitable for solving linear programs. There exist matured algorithms, like the simplex method, proposed by Dantzig in 1947 (Dantzig, 1991), or interior point methods that are capable to solve linear programs with large systems of linear inequalities in reasonable time.

**Definition 2.5 (Dual Linear Program)**
Let

$$\text{maximize } \mathbf{n}^T\mathbf{x} \text{ subject to } A\mathbf{x} \le \mathbf{a} \tag{P}$$

be a linear program. We call (P) the *primal linear program* and associate the primal linear program with the *dual linear program*

$$\text{minimize } \mathbf{u}^T\mathbf{a} \text{ subject to } A^T\mathbf{u} = \mathbf{n},\ \mathbf{u} \ge \mathbf{0}. \tag{D}$$

■

To familiarize us with an important proof technique of the upcoming chapter, we prove the next small lemma.

**Lemma 2.6 (The Dual of (D) is Equivalent to (P))**
Given a primal linear program (P) and its dual linear program (D) as above, then the dual of (D) is equivalent to the primal linear program. ■

*Proof.* Let I be the unit matrix. The following are equivalent transformations of (D).

$$\begin{aligned}
&\text{minimize } \mathbf{u}^T\mathbf{a} \text{ subject to } A^T\mathbf{u} = \mathbf{n},\ \mathbf{u} \ge \mathbf{0}\\
\iff\ &\text{minimize } \mathbf{u}^T\mathbf{a} \text{ subject to } \begin{pmatrix} A^T \\ -A^T \\ -\mathrm{I} \end{pmatrix} \mathbf{u} \le \begin{pmatrix} \mathbf{n} \\ -\mathbf{n} \\ \mathbf{0} \end{pmatrix}\\
\iff\ &\text{maximize } \mathbf{u}^T(-\mathbf{a}) \text{ subject to } \begin{pmatrix} A^T \\ -A^T \\ -\mathrm{I} \end{pmatrix} \mathbf{u} \le \begin{pmatrix} \mathbf{n} \\ -\mathbf{n} \\ \mathbf{0} \end{pmatrix}.
\end{aligned} \tag{2.3}$$

Dualizing the linear program (2.3) yields

$$\begin{aligned}
&\text{minimize } \mathbf{n}^T\mathbf{x} - \mathbf{n}^T\mathbf{y} \text{ subject to } A\mathbf{x} - A\mathbf{y} - \mathbf{z} = -\mathbf{a},\ \mathbf{x} \ge \mathbf{0},\ \mathbf{y} \ge \mathbf{0},\ \mathbf{z} \ge \mathbf{0}\\
\iff\ &\text{minimize } -\mathbf{n}^T(\mathbf{y} - \mathbf{x}) \text{ subject to } A(\mathbf{y} - \mathbf{x}) \le \mathbf{a},\ \mathbf{x} \ge \mathbf{0},\ \mathbf{y} \ge \mathbf{0}\\
\iff\ &\text{maximize } \mathbf{n}^T\mathbf{w} \text{ subject to } A\mathbf{w} \le \mathbf{a},
\end{aligned}$$

where the last equivalence is exactly the linear program (P). □

The following theorems establish some properties of the pair of a primal linear program (P) and its dual linear program (D).

**Theorem 2.7 (Weak Duality Theorem)**
For each feasible solution $\mathbf{x}$ of the primal linear program (P) and each feasible solution $\mathbf{u}$ of its dual linear program (D) we have

$$\mathbf{n}^T\mathbf{x} \leq \mathbf{a}^T\mathbf{u}. \qquad \blacksquare$$

**Theorem 2.8 (Strong Duality Theorem)**
For the primal linear program (P) and its dual linear program (D) exactly one of the following four possibilities hold

1. neither (P) nor (D) have a feasible solution,
2. (P) is unbounded and (D) is infeasible,
3. (P) is infeasible and (D) is unbounded
4. (P) and (D) are feasible, and for each optimal solution $\mathbf{x}$ of (P) and each optimal solution $\mathbf{u}$ of (D) we have

$$\mathbf{n}^T\mathbf{x} = \mathbf{a}^T\mathbf{u}. \qquad \blacksquare$$

# 3. Symbolic Orthogonal Projections

Im großen Garten der Geometrie kann sich jeder nach seinem Geschmack einen Strauß pflücken.

In the big garden that is geometry everyone may pick a bouquet to his liking.

David Hilbert, 1862-1943

We introduce a novel representation 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 $\mathbb{K}$. A sop encodes a polyhedron as an orthogonal projection of a higher-dimensional $\mathcal{H}$-polyhedron. The idea is to treat the projection and the higher-dimensional polyhedron symbolically rather than to compute an $\mathcal{H}$-representation of the polyhedron itself that is described by the sop. We show that this representation allows an efficient computation of various geometrical operations. In fact, many operations like the convex hull or the Minkowski sum can be performed by simple block matrix operations. Hence, the complexity of these operations is far better than the negative complexity results for $\mathcal{H}$- and $\mathcal{V}$-polyhedra. Due to the underlying $\mathcal{H}$-polyhedron, linear programs can be used to evaluate sops. This allows an efficient computation of support functions and over-approximations in terms of template polyhedra. Beyond this, we will also show how linear programming can be used to improve these over-approximations by adding additional supporting half-spaces, or even better, by adding facet-defining half-spaces. On the other hand, we shall also discuss a drawback of this representation, the lack of an efficient method to decide the subset relation of sops.

Parts of this chapter have been published as Hagemann (2014b). Most proofs have undergone substantial simplifications since they were first published.

**Convention.** For convenience we will use a block matrix notation. The unit matrix is denoted by I, the zero matrix by O, and an empty matrix having either zero rows or columns is denoted by $\emptyset$. A column vector whose coefficients are all zero is denoted by $\mathbf{0}$, and a column vector whose coefficient are all one by $\mathbf{1}$. The concrete number of columns and rows should become clear from the context.

The following convention for support functions is useful to handle infeasibility, unboundedness, and optimality uniformly: The notation $h_{\mathbf{P}}(\mathbf{n}) = \infty$ indicates that $\mathbf{P}$ is unbounded in direction $\mathbf{n}$, and $h_{\mathbf{P}}(\mathbf{n}) = -\infty$ indicates that $\mathbf{P}$ is empty.

**Definition 3.1 (Symbolic Orthogonal Projection)**
Let $A$ be an $(m \times d)$ matrix, $L$ an $(m \times k)$ matrix, and $\mathbf{a}$ be a column vector with $m$ coefficients. A *symbolic orthogonal projection*, or *sop*, for short, $\mathbf{P}\,(A, L, \mathbf{a})$ is the triple

$(A, L, \mathbf{a})$ together with its intended geometrical interpretation

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

$\mathbf{P}$ is a polyhedron in $\mathbb{K}^d$, and we shall write $\mathbf{P} = \mathbf{P}\left(A, L, \mathbf{a}\right)$ to denote that the polyhedron $\mathbf{P}$ is the interpretation of the symbolic orthogonal projection $\mathbf{P}\left(A, L, \mathbf{a}\right)$. In other words, $\mathbf{P} = \mathbf{P}\left(A, L, \mathbf{a}\right)$ is the image of the higher-dimensional $\mathcal{H}$-polyhedron

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

under the orthogonal projection

$$\operatorname{proj}_d\colon \quad \mathbb{K}^{d+k} \to \mathbb{K}^d, \qquad \begin{pmatrix}\mathbf{x}\\ \mathbf{z}\end{pmatrix} \mapsto \mathbf{x}. \qquad \blacksquare$$

Sops have interesting properties which make them useful in the context of system analysis. The entire chapter is devoted to derive various properties of sops. We start with the following obvious proposition.

**Proposition 3.2 (Emptiness)**
A sop $\mathbf{P}\left(A, L, \mathbf{a}\right)$ is empty if and only if the preimage $\mathbf{P}\left(\begin{pmatrix} A & L \end{pmatrix}, \mathbf{a}\right)$ is empty. $\blacksquare$

## 3.1. Conversions

The following propositions show that $\mathcal{H}$- and $\mathcal{V}$-representations may easily be converted to symbolic orthogonal projections.

**Proposition 3.3 (Converting $\mathcal{H}$-Representation to Sop)**
Let $\mathbf{P}\left(A, \mathbf{a}\right)$ be an $\mathcal{H}$-polyhedron. Then $\mathbf{P}\left(A, \emptyset, \mathbf{a}\right)$ is a symbolic orthogonal projection and the identity $\mathbf{P}\left(A, \mathbf{a}\right) = \mathbf{P}\left(A, \emptyset, \mathbf{a}\right)$ holds. $\blacksquare$

*Proof.* Obvious. □

**Proposition 3.4 (Converting $\mathcal{V}$-Representation to Sop)**
Let $\mathbf{P} = \operatorname{cone}(\mathbf{U}) + \operatorname{conv}(\mathbf{V})$ be a $\mathcal{V}$-polyhedron and let $U$ and $V$ be matrices whose columns are the elements of $\mathbf{U}$ and $\mathbf{V}$, respectively. The polyhedron $\mathbf{P}$ is defined as the set

$$\mathbf{P} = \left\{\mathbf{x} \in \mathbb{K}^d \,\middle|\, \exists \mathbf{p} \geq \mathbf{0},\, \mathbf{q} \geq \mathbf{0}\colon\ \mathbf{1}^T\mathbf{q} = 1,\, U\mathbf{p} + V\mathbf{q} = \mathbf{x}\right\}. \tag{3.1}$$

Then

$$\mathbf{P} = \mathbf{P}\left(\begin{pmatrix} -\mathrm{I} \\ \mathrm{I} \\ \mathrm{O} \\ \mathrm{O} \\ \mathbf{0}^T \\ \mathbf{0}^T \end{pmatrix}, \begin{pmatrix} U & V \\ -U & -V \\ -\mathrm{I} & \mathrm{O} \\ \mathrm{O} & -\mathrm{I} \\ \mathbf{0}^T & \mathbf{1}^T \\ \mathbf{0}^T & -\mathbf{1}^T \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ 1 \\ -1 \end{pmatrix}\right). \qquad \blacksquare$$

*Proof.* By carefully rewriting (3.1). □

On the other hand, the conversion of a symbolic orthogonal projection back to an $\mathcal{H}$-representation may be done with Fourier-Motzkin elimination (Theorem 2.1). This operation is expensive since for each eliminated variable the number of inequalities roughly gets squared. Thus, we should avoid this operation whenever possible. Moreover, since the sop-based representation encompasses the $\mathcal{H}$-representation, conversion of a sop to a $\mathcal{V}$-polyhedron is at least as hard as generating all vertices and extreme rays of an $\mathcal{H}$-polyhedron (Khachiyan et al., 2008).

## 3.2. Support Function and Template Polyhedra

The following proposition shows that the value of the support function of a sop can be computed by means of linear programming.

**Proposition 3.5 (Support Function)**
Let $\mathbf{P} = \mathbf{P}(A, L, \mathbf{a})$ be a sop. Then for any $\mathbf{n}$ the identity

$$h_{\mathbf{P}(A,L,\mathbf{a})}(\mathbf{n}) = h_{\mathbf{P}((A\ \ L),\mathbf{a})}\left(\begin{pmatrix}\mathbf{n}\\ \mathbf{0}\end{pmatrix}\right)$$

holds. Hence, the value of the support function $h_{\mathbf{P}}(\mathbf{n})$ is given by the linear program

$$\text{maximize } \mathbf{n}^T\mathbf{x} \text{ subject to } A\mathbf{x} + L\mathbf{z} \leq \mathbf{a}. \tag{3.2}$$

Moreover, let $\binom{\mathbf{x}_0}{\mathbf{z}_0}$ be an optimal solution of the linear program (3.2). Then the orthogonal projection $\mathbf{x}_0 = \mathrm{proj}_d\left(\binom{\mathbf{x}_0}{\mathbf{z}_0}\right)$ is an optimal vector of the sop $\mathbf{P}(A, L, \mathbf{a})$. ■

*Proof.* Obvious.

Support functions are useful for computing over-approximations in terms of template polyhedra.

**Definition 3.6 (Vector Valued Support Function)**
Let $\mathbf{P} = \mathbf{P}(A, L, \mathbf{a})$ be a sop in $\mathbb{K}^d$ and $A_{\text{fix}}$ be an $(m \times d)$ matrix. We denote the rows of $A_{\text{fix}}$ by the row vectors $\mathbf{n}_1^T, \ldots, \mathbf{n}_m^T$ and define

$$\mathbf{h}_{\mathbf{P}}(A_{\text{fix}}) = \begin{pmatrix} h_{\mathbf{P}}(\mathbf{n}_1)\\ h_{\mathbf{P}}(\mathbf{n}_2)\\ \vdots\\ h_{\mathbf{P}}(\mathbf{n}_m)\end{pmatrix}.$$ ■

As a first application we note the following criterion.

**Proposition 3.7 (Boundedness of Sops)**
Let $\mathbf{P} = \mathbf{P}(A, L, \mathbf{a})$ be a non-empty sop in $\mathbb{K}^d$ and $A_{\text{fix}}$ be a matrix of $d + 1$ affinely independent row vectors $\mathbf{n}_1^T, \ldots, \mathbf{n}_{d+1}^T$, (i. e., $\mathbf{n}_1 - \mathbf{n}_{d+1}, \ldots, \mathbf{n}_d - \mathbf{n}_{d+1}$ are linearly independent). Let $\mathbf{a} = \mathbf{h}_{\mathbf{P}}(A_{\text{fix}})$. If all coefficients of $\mathbf{a}$ are in $\mathbb{K}$, i. e., none is equal to $\infty$, then $\mathbf{P}$ is a polytope. Otherwise, $\mathbf{P}$ is unbounded. ■

*Proof.* Let $\mathbf{P}' = \mathbf{P}(A_{\text{fix}}, \mathbf{a})$. Clearly, $\mathbf{P}'$ is an over-approximation of $\mathbf{P}$. If every coefficient of $\mathbf{a}$ is bounded, then $\mathbf{P}'$ is a $d$-simplex and, hence, bounded in every direction. Then $\mathbf{P}$ must also be bounded in every direction. On the other hand, if $\mathbf{P}$ is bounded, then every coefficient of $\mathbf{a}$ is bounded. □

In general a sop and an over-approximating template polyhedron will have none or only a few facets in common since the normal vectors of the facets of a template polyhedron are fixed by the template matrix. In Section 3.12 we will see how linear programming allows us to extract additional facet-defining half-spaces of a sop.

## 3.3. Duality and the Polar

The dual linear program to (3.2) is

$$\text{minimize } \mathbf{a}^T\mathbf{u} \text{ subject to } \mathbf{u} \geq \mathbf{0},\ A^T\mathbf{u} = \mathbf{n},\ L^T\mathbf{u} = \mathbf{0}. \tag{3.3}$$

By Strong Duality (Theorem 2.8) we have $\mathbf{a}^T\mathbf{u} \geq h_{\mathbf{P}}(\mathbf{n})$ for any feasible solution of (3.3), and, if $\mathbf{P}$ is bounded in direction $\mathbf{n}$, then at least one feasible solution $\mathbf{u} \geq \mathbf{0}$ with $A^T\mathbf{u} = \mathbf{n}$, $L^T\mathbf{u} = \mathbf{0}$ and $\mathbf{a}^T\mathbf{u} = h_{\mathbf{p}}(\mathbf{n})$ exists.

As a first application of the duality of linear programs we show the following proposition that allows us to represent polars in terms of symbolic orthogonal projections. The polar of set $\mathbf{P}$ is defined as the set

$$\mathbf{P}^* = \left\{\mathbf{n} \,\middle|\, \mathbf{n}^T\mathbf{x} \leq 1 \text{ for all } \mathbf{x} \in \mathbf{P}\right\}.$$

Building the polar is a dual operation for polyhedra containing the origin. Indeed, if $\mathbf{P}$ is a closed convex set and $\mathbf{0} \in \mathbf{P}$, then $\mathbf{P}^{**} = \mathbf{P}$, see Rockafellar and Wets (1998).

**Proposition 3.8 (Polar)**
Let $\mathbf{P} = \mathbf{P}\left(A, L, \mathbf{a}\right)$ be a sop. Then the following equation holds:

$$\mathbf{P}^* = \mathbf{P}\left(\begin{pmatrix} \mathrm{I} \\ -\mathrm{I} \\ \mathrm{O} \\ \mathrm{O} \\ \mathrm{O} \\ \mathbf{0}^T \end{pmatrix}, \begin{pmatrix} -A^T \\ A^T \\ L^T \\ -L^T \\ -\mathrm{I} \\ \mathbf{a}^T \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ 1 \end{pmatrix}\right). \tag{3.4}$$

■

*Proof.* The proposition follows from the following identities:

$$\begin{aligned} \mathbf{P}^* &= \left\{\mathbf{n} \,\middle|\, \mathbf{n}^T\mathbf{x} \leq 1 \text{ for all } \mathbf{x} \in \mathbf{P}\right\} = \left\{\mathbf{n} \,\middle|\, h_{\mathbf{P}}(\mathbf{n}) \leq 1\right\} \\ &= \left\{\mathbf{n} \,\middle|\, \exists \mathbf{u} \geq \mathbf{0}\colon\ A^T\mathbf{u} = \mathbf{n},\ L^T\mathbf{u} = \mathbf{0},\ \mathbf{a}^T\mathbf{u} \leq 1\right\} \\ &= \left\{\mathbf{n} \,\middle|\, \exists \mathbf{u}\colon\ \mathrm{I}\mathbf{n} - A^T\mathbf{u} = \mathbf{0},\ L^T\mathbf{u} = \mathbf{0},\ \mathbf{u} \geq \mathbf{0},\ \mathbf{a}^T\mathbf{u} \leq 1\right\}. \end{aligned}$$

Finally, rewriting the equalities as pairs of inequalities yields (3.4). □

## 3.4. Intersection, Minkowski Sum, and Translation

**Proposition 3.9 (Intersection)**
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 sops. Then the following equation holds:

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

*Proof.* The proposition follows immediately from the equivalences:

$$\begin{aligned} \mathbf{x} \in \mathbf{P}_1 \cap \mathbf{P}_2 &\iff \mathbf{x} \in \mathbf{P}_1 \text{ and } \mathbf{x} \in \mathbf{P}_2 \\ &\iff \exists \mathbf{z}_1, \mathbf{z}_2\colon\ A_1\mathbf{x} + L_1\mathbf{z}_1 \le \mathbf{a}_1,\ A_2\mathbf{x} + L_2\mathbf{z}_2 \le \mathbf{a}_2. \qquad \square \end{aligned}$$

**Proposition 3.10 (Minkowski Sum)**
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 sops. Then the following equation holds:

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

where $\mathbf{P}_1 + \mathbf{P}_2$ denotes the Minkowski sum of $\mathbf{P}_1$ and $\mathbf{P}_2$. $\blacksquare$

*Proof.* The proposition follows from the equivalences:

$$\begin{aligned} \mathbf{x} \in \mathbf{P}_1 + \mathbf{P}_2 &\iff \exists \mathbf{x}_1 \in \mathbf{P}_1\ \exists \mathbf{x}_2 \in \mathbf{P}_2\colon\ \mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2 \iff \exists \mathbf{x}_2 \in \mathbf{P}_2\colon\ \mathbf{x} - \mathbf{x}_2 \in \mathbf{P}_1 \\ &\iff \exists \mathbf{x}_2, \mathbf{y}_1, \mathbf{y}_2\colon\ A_1\mathbf{x} - A_1\mathbf{x}_2 + L_1\mathbf{y}_1 \le \mathbf{a}_1,\ A_2\mathbf{x}_2 + L_2\mathbf{y}_2 \le \mathbf{a}_2. \qquad \square \end{aligned}$$

The translation of a polyhedron by some vector $\mathbf{v}$ is a special case of the Minkowski sum.

**Proposition 3.11 (Translation)**
Let $\mathbf{P} = \mathbf{P}(A, L, \mathbf{a})$ be a sop and let $\mathbf{v}$ a vector. Then for the translation $\mathbf{P} + \mathbf{v}$ the identity $\mathbf{P}(A, L, \mathbf{a} + A\mathbf{v}) = \mathbf{P} + \mathbf{v}$ holds. $\blacksquare$

*Proof.* The proposition follows from the equivalences:

$$\begin{aligned} \mathbf{x} \in \mathbf{P} + \mathbf{v} &\iff \exists \mathbf{x}_1 \in \mathbf{P}\colon\ \mathbf{x} = \mathbf{x}_1 + \mathbf{v} \iff \mathbf{x} - \mathbf{v} \in \mathbf{P} \\ &\iff \exists \mathbf{z}_1\colon\ A\mathbf{x} + L\mathbf{z}_1 \le \mathbf{a} + A\mathbf{v}. \qquad \square \end{aligned}$$

Another special case is given in the following proposition.

**Proposition 3.12 (Extrusion)**
Let $\mathbf{P} = \mathbf{P}(A, L, \mathbf{a})$ be a sop, $\mathbf{v}$ be a vector, and $[t_0, t_1]$ be an interval with $t_0 \in \mathbb{K} \cup \{-\infty\}$, $t_1 \in \mathbb{K} \cup \{\infty\}$ and $t_0 \le t_1$. Then the sop representing the extrusion of $\mathbf{P}$ along $\{t\mathbf{v} \mid t \in [t_0, t_1]\}$, i. e., $\mathbf{P} + \bigcup_{t\in[t_0,t_1]} t\mathbf{v}$, is given by the following equation

$$\mathbf{P} + \bigcup_{t\in[t_0,t_1]} t\mathbf{v} = \mathbf{P}\left(\begin{pmatrix} A \\ \mathbf{0}^T \\ \mathbf{0}^T \end{pmatrix}, \begin{pmatrix} L & -A\mathbf{v} \\ \mathbf{0}^T & -1 \\ \mathbf{0}^T & 1 \end{pmatrix}, \begin{pmatrix} \mathbf{a} \\ -t_0 \\ t_1 \end{pmatrix}\right),$$

where we may omit the last or the second last row of the sop representation if $t_1 = \infty$ or $t_0 = -\infty$, respectively. $\blacksquare$

*Proof.* The equation follows immediately from the equivalences:

$$\begin{aligned}\mathbf{x} \in \mathbf{P} + \bigcup_{t\in[t_0,t_1]} t\mathbf{v} &\iff \exists \mathbf{y},\mathbf{z}\ \exists t \in [t_0,t_1]\colon\ A\mathbf{y} + L\mathbf{z} \leq \mathbf{a},\ \mathbf{x} = \mathbf{y} + t\mathbf{v} \\ &\iff \exists \mathbf{z}, t\colon\ A\mathbf{x} - tA\mathbf{v} + L\mathbf{z} \leq \mathbf{a}, -t \leq -t_0, t \leq t_1. \qquad \square\end{aligned}$$

## 3.5. Linear and Affine Transformations

We begin with the most general proposition, which is inspired by Bastout (2004).

**Proposition 3.13 (Affine Mapping)**
Let $\mathbf{P}_1 = \mathbf{P}(A_1, L_1, \mathbf{a}_1)$ be a sop in $\mathbb{K}^d$. Further, let $M \in \mathbb{K}^{l\times d}$ be the transformation matrix and $\mathbf{v} \in \mathbb{K}^l$ be the translation vector of the affine mapping $\mathbf{f}(\mathbf{x}) = M\mathbf{x}+\mathbf{v}$. Then

$$\mathbf{f}(\mathbf{P}_1) = M\mathbf{P}_1 + \mathbf{v} = \mathbf{P}\left(\begin{pmatrix} \mathrm{I}_l \\ -\mathrm{I}_l \\ \mathrm{O} \end{pmatrix}, \begin{pmatrix} -M & \mathrm{O} \\ M & \mathrm{O} \\ A_1 & L_1 \end{pmatrix}, \begin{pmatrix} \mathbf{v} \\ -\mathbf{v} \\ \mathbf{a}_1 \end{pmatrix}\right).$$

The sop-based representation of $M\mathbf{P}_1 + \mathbf{v}$ has $l$ additional columns and $2l$ additional rows compared to the representation of $\mathbf{P}_1$. ■

*Proof.* The set identity immediately follows from the equivalences:

$$\mathbf{x} \in \mathbf{f}(\mathbf{P}_1) \iff \exists \mathbf{y}\colon\ \mathbf{y} \in \mathbf{P}_1,\ \mathbf{x} = M\mathbf{y} + \mathbf{v} \iff \exists \mathbf{y},\mathbf{z}\colon\ A_1\mathbf{y} + L_1\mathbf{z} \leq \mathbf{a}_1,\ \mathrm{I}_l\mathbf{x} = M\mathbf{y} + \mathbf{v}. \ \square$$

The preimage of an affine mapping is a simple operation.[1]

**Proposition 3.14 (Preimage under Affine Mappings)**
Let $\mathbf{P}_1 = \mathbf{P}(A_1, L_1, \mathbf{a}_1)$ be a sop in $\mathbb{K}^l$. Further, let $M \in \mathbb{K}^{l\times d}$ be the representation matrix, and $\mathbf{v} \in \mathbb{K}^l$ be the translation vector of the affine mapping $\mathbf{f}(\mathbf{x}) = M\mathbf{x} + \mathbf{v}$. Then the sop representing the preimage $\mathbf{f}^{-1}(\mathbf{P}_1) = \{\mathbf{x} \in \mathbb{K}^d \mid \exists \mathbf{y} \in \mathbf{P}_1\colon\ M\mathbf{x} + \mathbf{v} = \mathbf{y}\}$ is given by the following equation

$$\mathbf{f}^{-1}(\mathbf{P}_1) = \mathbf{P}(A_1M, L_1, \mathbf{a}_1 - A_1\mathbf{v}). \qquad ■$$

*Proof.* The equation follows immediately from the following equivalences.

$$\begin{aligned}\mathbf{x} \in \mathbf{f}^{-1}(\mathbf{P}_1) &\iff \exists \mathbf{y} \in \mathbf{P}_1\colon\ \mathbf{y} = M\mathbf{x} + \mathbf{v} \iff \exists \mathbf{y},\mathbf{z}\colon\ A_1\mathbf{y} + L_1\mathbf{z} \leq \mathbf{a}_1,\ \mathbf{y} = M\mathbf{x} + \mathbf{v} \\ &\iff \exists \mathbf{z}\colon\ A_1M\mathbf{x} + L_1\mathbf{z} \leq \mathbf{a}_1 - A_1\mathbf{v} \iff \mathbf{x} \in \mathbf{P}(A_1M, L_1, \mathbf{a}_1 - A_1\mathbf{v}). \qquad \square\end{aligned}$$

[1] It is a well-known fact that the preimage computation of affine mappings on $\mathcal{H}$-polyhedra does not involve quantifier elimination. For example, the model-checker FOMC exploits this fact via backward model-checking. I would like to thank Uwe Waldmann who pointed out that this property should carry over to sops.

In the following we describe how an arbitrary linear mapping can be decomposed into special mappings for which compact sop-representations exist. The decomposition method leads to smaller representation sizes in general. Any linear mapping $\mathbf{f}$ is uniquely determined by its transformation matrix $M \in \mathbb{K}^{l\times d}$, i. e., $\mathbf{f}(\mathbf{x}) = M\mathbf{x}$. The next proposition shows that any linear transformation can be written as a composition of the following three types of linear mappings, where the $(n\times n)$-identity matrix is denoted by $\mathrm{I}_n$: (i) *automorphisms*, having invertible transformation matrices; (ii) *orthogonal projections* $\mathrm{proj}_{r,d}$ having $(r\times d)$-matrices of the form $(\mathrm{I}_r \;\; \mathrm{O})$; and (iii) *elementary embeddings* $\mathrm{embed}_{l,r}$ having $(l\times r)$-matrices of the form $\binom{\mathrm{I}_r}{\mathrm{O}}$.

**Proposition 3.15 (Decomposition of Linear Mappings)**
Every transformation matrix $M \in \mathbb{K}^{l\times d}$ can be written as the product $M = S^{-1}EPT^{-1}$ where $S \in \mathbb{K}^{l\times l}$ and $T \in \mathbb{K}^{d\times d}$ are invertible, $E \in \mathbb{K}^{l\times r}$ is the matrix of an elementary embedding, and $P \in \mathbb{K}^{r\times d}$ is the matrix of an orthogonal projection for some $r$ with $0 \le r \le \min(d,l)$. ■

*Proof.* We transform $M$ into row echelon form and then into column echelon form by multiplying first with an invertible matrix $S$ from the left and then multiplying with an invertible matrix $T$ from the right, yielding

$$SMT = \begin{pmatrix} \mathrm{I}_r & \mathrm{O} \\ \mathrm{O} & \mathrm{O} \end{pmatrix} = \begin{pmatrix} \mathrm{I}_r \\ \mathrm{O} \end{pmatrix} (\mathrm{I}_r \quad \mathrm{O}) = EP. \qquad \square$$

**Proposition 3.16 (Bijective Linear Mapping)**
Let $\mathbf{P}_1 = \mathbf{P}(A_1, L_1, \mathbf{a}_1)$ be a sop in $\mathbb{K}^d$ and $T$ an invertible $(d\times d)$-transformation matrix of a linear mapping. Then

$$T\mathbf{P}_1 = \mathbf{P}\left(A_1T^{-1}, L_1, \mathbf{a}_1\right).$$

The representation of $T\mathbf{P}_1$ has as many rows and columns as the representation of $\mathbf{P}_1$.■

*Proof.* The set identity immediately follows from the equivalences:

$$\begin{aligned} \mathbf{x} \in T\mathbf{P}_1 &\iff T^{-1}\mathbf{x} \in \mathbf{P}_1 \iff \exists \mathbf{z}\colon\; A_1T^{-1}\mathbf{x} + L_1\mathbf{z} \le \mathbf{a}_1 \\ &\iff \mathbf{x} \in \mathbf{P}\left(A_1T^{-1}, L_1, \mathbf{a}_1\right). \end{aligned} \qquad \square$$

**Proposition 3.17 (Orthogonal Projection)**
Let $\mathbf{P}_1 = \mathbf{P}(A_1, L_1, \mathbf{a}_1)$ be a sop in $\mathbb{K}^d$ and $\mathrm{proj}_{r,d}$ an orthogonal projection with $0 \le r \le d$. Then

$$\mathrm{proj}_{r,d}(\mathbf{P}_1) = \mathbf{P}(A, L, \mathbf{a}_1)$$

where the matrix equality $(A \quad L) = (A_1 \quad L_1)$ holds and $A$ has $r$ columns. The representation of $\mathrm{proj}_{r,d}(\mathbf{P}_1)$ has as many rows and columns as the representation of $\mathbf{P}_1$. ■

*Proof.* Obviously, $\mathbf{P}(A_1, L_1, \mathbf{a}) \in \mathbb{K}^d$ and $\mathbf{P}(A, L, \mathbf{a}) \in \mathbb{K}^k$ are both orthogonal projections of the same polyhedron $\mathbf{Q} = \mathbf{P}((A_1 \quad L_1), \mathbf{a})$, only the dimension of the codomain differs. It does not matter whether we project $\mathbf{Q}$ in two steps first onto $\mathbb{K}^d$ and then project the result onto $\mathbb{K}^k$, or project $\mathbf{Q}$ in one step onto $\mathbb{K}^k$, as we easily see by looking at the matrix product of the corresponding projection matrices. □

**Proposition 3.18 (Elementary Embedding)**
Let $\mathbf{P}_1 = \mathbf{P}(A_1, L_1, \mathbf{a}_1)$ be a sop in $\mathbb{K}^r$ and $\text{embed}_{l,r}$ an elementary embedding with $l \geq r$. Then

$$\text{embed}_{l,r}(\mathbf{P}_1) = \mathbf{P}\left(\begin{pmatrix} A_1 & \mathrm{O} \\ \mathrm{O} & \mathrm{I}_{l-r} \\ \mathrm{O} & -\mathrm{I}_{l-r} \end{pmatrix}, \begin{pmatrix} L_1 \\ \mathrm{O} \\ \mathrm{O} \end{pmatrix}, \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}\right). \tag{3.5}$$

The representation of $\text{embed}_{l,r}(\mathbf{P}_1)$ has $l - r$ additional columns and $2(l - r)$ additional rows compared to the representation of $\mathbf{P}_1$ ■

*Proof.* The set identity (3.5) immediately follows from the equivalences:

$$\begin{aligned} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \in \text{embed}_{l,r}(\mathbf{P}_1) &\iff \exists \mathbf{z}\colon\ A_1\mathbf{x} + L_1\mathbf{z} \leq \mathbf{a}_1,\ \mathbf{y} = \mathbf{0} \\ &\iff \exists \mathbf{z}\colon\ \begin{pmatrix} A_1 & \mathrm{O} \\ \mathrm{O} & \mathrm{I}_{l-r} \\ \mathrm{O} & -\mathrm{I}_{l-r} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} + \begin{pmatrix} L_1 \\ \mathrm{O} \\ \mathrm{O} \end{pmatrix} \mathbf{z} \leq \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}. \end{aligned}$$ □

The elementary embedding is strongly related to cylindrification.

**Proposition 3.19 (Cylindrification)**
Let $\mathbf{P}_1 = \mathbf{P}(A_1, L_1, \mathbf{a}_1)$ be a sop in $\mathbb{K}^r$. Further, for $l \geq r$ let $\text{cyl}_{l,r}(\mathbf{P}_1)$ be the cylindrification of $\mathbf{P}_1$ in $\mathbb{K}^l$, i. e., the set

$$\text{cyl}_{l,r}(\mathbf{P}_1) = \mathbf{P}_1 \times \mathbb{K}^{l-r} = \left\{ \begin{pmatrix} \mathbf{x} \\ \mathbf{z} \end{pmatrix} \middle| \mathbf{x} \in \mathbf{P}_1,\ \mathbf{z} \in \mathbb{K}^{l-r} \right\}.$$

Then

$$\text{cyl}_{l,r}(\mathbf{P}_1) = \mathbf{P}((A_1 \quad \mathrm{O}), L_1, \mathbf{a}_1). \tag{3.6}$$

■

*Proof.* The set identity (3.6) immediately follows from the equivalences:

$$\mathbf{x} \in \mathbf{P}_1,\ \mathbf{z} \in \mathbb{K}^{l-d} \iff \exists \mathbf{y}\colon\ (A_1 \quad \mathrm{O}) \begin{pmatrix} \mathbf{x} \\ \mathbf{z} \end{pmatrix} + L_1\mathbf{y} \leq \mathbf{a} \iff \begin{pmatrix} \mathbf{x} \\ \mathbf{z} \end{pmatrix} \in \text{cyl}_{l,r}(\mathbf{P}_1).$$ □

Finally, let us shortly discuss the two different methods to compute the image of a sop under the affine mapping $M\mathbf{x} + \mathbf{v}$ with $M \in \mathbb{K}^{l \times d}$ and $\mathbf{v} \in \mathbb{K}^l$. The direct method of Proposition 3.13 allows us to represent the image by simple block operations without additional computations and leads to a growth of the representation by $l$ columns and $2l$ rows. The decomposition method involves additional computations, which are Gaussian elimination and matrix products, and the representation grows by $l - r$ columns and $2(l-r)$ rows where $r$ with $0 \leq r \leq \min(d, l)$ is uniquely determined by the decomposition of $M$. Which method we should prefer is not clear in general and might depend on the context. The direct method has been found recently, and our experiences are mostly based on the decomposition method.

## 3.6. Scaling

We discuss a special case of a linear mapping, the mapping $\lambda : \mathbf{P} \to \lambda\mathbf{P}$ for $\lambda \in \mathbb{K}$, i. e., scaling of polyhedra. For the special case of polytopes, this operation can be expressed as a manipulation of the vector $\mathbf{a}$ in the sop-representation $\mathbf{P}\,(A, L, \mathbf{a})$. It turns out that polyhedra and polytopes behave slightly differently under this manipulation.

The recession cone of a polyhedron $\mathbf{P}$ is the set

$$\operatorname{rec}(\mathbf{P}) = \{\mathbf{x} \,|\, \forall \mathbf{y} \in \mathbf{P}, \lambda \geq 0\colon\ \mathbf{y} + \lambda\mathbf{x} \in \mathbf{P}\}\,,$$

which is the set of all directions in that $\mathbf{P}$ is not bounded. If $\mathbf{P}$ is bounded in each direction, i. e., $\mathbf{P}$ is a polyhedron, then $\operatorname{rec}(\mathbf{P}) = \{\mathbf{0}\}$. Moreover, let $\mathbf{P} = \mathbf{P}(A, \mathbf{a})$ be an $\mathcal{H}$-representation and $\mathbf{P} = \operatorname{cone}(\mathbf{U}) + \operatorname{conv}(\mathbf{V})$ be a $\mathcal{V}$-representation. Then the identities $\operatorname{rec}(\mathbf{P}) = \mathbf{P}(A, \mathbf{0})$ and $\operatorname{rec}(\mathbf{P}) = \operatorname{cone}(\mathbf{U})$ hold, see Ziegler (1995).

**Lemma 3.20 (Separation)**
Let $\mathbf{P} = \mathbf{P}\,(A, L, \mathbf{a})$ be a sop and $\mathbf{z}$ some point with $\mathbf{z} \notin \mathbf{P}$. Then there exists some $\mathbf{u}_0 \geq \mathbf{0}$ such that the half-space $\mathbf{H}(\mathbf{n}, c)$ with $\mathbf{n} = A^T\mathbf{u}_0$, $c = \mathbf{a}^T\mathbf{u}_0$ separates $\mathbf{z}$ and $\mathbf{P}$:

$$\mathbf{n}^T\mathbf{z} > c \text{ and } \forall \mathbf{x} \in \mathbf{P}\colon\ \mathbf{n}^T\mathbf{x} \leq c. \qquad \blacksquare$$

*Proof.* If $\mathbf{z}$ is not in $\mathbf{P}$, then the following system of linear inequalities has no solution.

$$A\mathbf{x} + L\mathbf{z} \leq \mathbf{a},\ \mathrm{I}\mathbf{x} \leq \mathbf{z},\ -\mathrm{I}\mathbf{x} \leq -\mathbf{z}$$

Hence, by Farkas' Lemma there exist $\mathbf{u}_0 \geq \mathbf{0}$, $\mathbf{u}_1 \geq \mathbf{0}$, and $\mathbf{u}_2 \geq \mathbf{0}$ with

$$\begin{aligned} A^T\mathbf{u}_0 + \mathrm{I}^T\mathbf{u}_1 - \mathrm{I}^T\mathbf{u}_2 &= \mathbf{0} \\ L^T\mathbf{u}_0 &= \mathbf{0} \\ \mathbf{a}^T\mathbf{u}_0 + \mathbf{z}^T\mathbf{u}_1 - \mathbf{z}^T\mathbf{u}_2 &< 0. \end{aligned}$$

We set $\mathbf{u}_2 - \mathbf{u}_1 = \mathbf{n}$, $\mathbf{a}^T\mathbf{u}_0 = c$ and obtain

$$A^T\mathbf{u}_0 = \mathbf{n},\ L^T\mathbf{u}_0 = \mathbf{0},\ c < \mathbf{n}^T\mathbf{z}.$$

On the other hand, for any $\mathbf{u} \geq 0$ with $L^T\mathbf{u} = \mathbf{0}$ and any $\mathbf{x}$, $\mathbf{y}$ with $A\mathbf{x} + L\mathbf{y} \leq \mathbf{a}$ we have $\mathbf{u}^T A\mathbf{x} \leq \mathbf{u}^T\mathbf{a}$. Hence, also for $\mathbf{u}_0$, with $\mathbf{n} = A^T\mathbf{u}_0$ and $c = \mathbf{a}^T\mathbf{u}_0$ as above, we obtain $\mathbf{n}^T\mathbf{x} \leq c$ for all $\mathbf{x} \in \mathbf{P}$. $\square$

**Proposition 3.21 (Non-negative Scaling and Recession Cone)**
Let $\mathbf{P}_1 = \mathbf{P}\left(A_1, L_1, \mathbf{a}_1\right)$ be a sop and $\lambda \geq 0$. Then for $\lambda > 0$ the identity

$$\lambda \mathbf{P}_1 = \mathbf{P}\left(A_1, L_1, \lambda \mathbf{a}\right)$$

and for $\lambda = 0$ the identity

$$\operatorname{rec}(\mathbf{P}_1) = \mathbf{P}\left(A_1, L_1, \lambda \mathbf{a}\right)$$

holds. If $\mathbf{P}_1$ is a polytope, i. e., bounded in every direction, then we may forgo the case distinction and obtain the identity $\lambda \mathbf{P}_1 = \mathbf{P}(A_1, L_1, \lambda \mathbf{a}_1)$ for any $\lambda \geq 0$. ■

*Proof.* Let $\lambda > 0$. Then the following equivalences hold.

$$\begin{aligned}\mathbf{x} \in \lambda \mathbf{P}_1 &\iff \exists \mathbf{y}, \mathbf{z}\colon\ A_1\mathbf{y} + L_1\mathbf{z} \leq \mathbf{a}_1,\ \mathbf{x} = \lambda\mathbf{y} \iff \exists \mathbf{y}, \mathbf{z}\colon\ A_1\lambda\mathbf{y} + L_1\lambda\mathbf{z} \leq \lambda\mathbf{a}_1, \mathbf{x} = \lambda\mathbf{y} \\ &\iff \exists \mathbf{z}'\colon\ A_1\mathbf{x} + L_1\mathbf{z}' \leq \lambda\mathbf{a}_1 \iff \mathbf{x} \in \mathbf{P}(A_1, L_1, \lambda\mathbf{a}_1).\end{aligned}$$

Let $\lambda = 0$. Assume, $\mathbf{x} \in \mathbf{P}\left(A_1, L_1, \mathbf{0}\right)$. Then there exists some $\mathbf{z}$ with $A_1\mathbf{x} + L_1\mathbf{z} \leq \mathbf{0}$ and, moreover, for all $\lambda' \geq 0$ the inequality

$$A_1\lambda'\mathbf{x} + L_1\lambda'\mathbf{z} \leq \mathbf{0} \tag{3.7}$$

holds. We want to show $\mathbf{x} \in \operatorname{rec}(\mathbf{P}_1)$. Therefore, let $\mathbf{y} \in \mathbf{P}_1$. Then there exists some $\mathbf{z}'$ with $A_1\mathbf{y} + L_1\mathbf{z}' \leq \mathbf{a}_1$. Using (3.7) we obtain $A_1(\lambda'\mathbf{x} + \mathbf{y}) + L_1(\lambda'\mathbf{z} + \mathbf{z}') \leq \mathbf{a}_1$. Hence, $\mathbf{y} + \lambda'\mathbf{x} \in \mathbf{P}_1$ for all $\lambda' \geq 0$, and we have shown $\mathbf{x} \in \operatorname{rec}(\mathbf{P}_1)$. On the other hand, if $\mathbf{x} \in \operatorname{rec}(\mathbf{P}_1)$, then we have to show that $\mathbf{x} \in \mathbf{P}(A_1, L_1, \mathbf{0})$. Suppose, $\mathbf{x} \notin \mathbf{P}(A_1, L_1, \mathbf{0})$. According to the previous lemma, there exists some $\mathbf{u}_0 \geq 0$ with $\mathbf{n} = A^T\mathbf{u}_0$, and $c = \mathbf{0}^T\mathbf{u}_0 = 0$ such that $\mathbf{P}\left(A_1, L_1, \mathbf{0}\right) \subseteq \mathbf{H}(\mathbf{n}, 0)$ and $\mathbf{n}^T\mathbf{x} > 0$. Since $\mathbf{x} \in \operatorname{rec}(\mathbf{P}_1)$, for all $\mathbf{y} \in \mathbf{P}_1$ and $\lambda' \geq 0$ we have $\mathbf{y} + \lambda'\mathbf{x} \in \mathbf{P}(A_1, L_1, \mathbf{a}_1)$, i. e., there exists some $\mathbf{z}$ with $A_1\lambda'\mathbf{x} + A_1\mathbf{y} + L_1\mathbf{z} \leq \mathbf{a}_1$. We multiply by $\mathbf{u}_0$ and obtain $\lambda'\mathbf{n}^T\mathbf{x} + \mathbf{n}^T\mathbf{y} \leq \mathbf{u}^T\mathbf{a}_1$. Since $\mathbf{n}^T\mathbf{x} > 0$ and $\lambda'$ may be chosen arbitrary large, this yields a contradiction. Hence, $\mathbf{x} \in \mathbf{P}(A_1, L_1, \mathbf{0})$. □

**Proposition 3.22 (Negative Scaling)**
Let $\mathbf{P}_1 = \mathbf{P}\left(A_1, L_1, \mathbf{a}_1\right)$ be a sop and $\lambda < 0$. Then the identity

$$\lambda \mathbf{P}_1 = \mathbf{P}\left(-A_1, L_1, -\lambda \mathbf{a}\right)$$

holds. ■

*Proof.* The proposition follows from the following equivalences:

$$\mathbf{x} \in \lambda\mathbf{P}_1 \iff -\mathbf{x} \in -\lambda\mathbf{P}_1 \iff \exists \mathbf{z}\colon\ -A_1\mathbf{x} + L_1\mathbf{z} \leq -\lambda\mathbf{a}_1.$$ □

## 3.7. Towards the Convex Hull

As motivation for the next section, we discuss the following construction describing the convex hull of a polyhedron $\mathbf{P}$ and the origin $\mathbf{0}$. Let $\mathbf{P} = \mathbf{P}(A, L, \mathbf{a})$ be a sop, and let $\mathbf{P}_1 = \bigcup_{t\in[0,1]} t\mathbf{P}$. Then $\mathbf{P}_1$ is the convex hull of the point $\mathbf{0}$ and $\mathbf{P}$. From the previous section we learnt that $t\mathbf{P}$ can be written as the sop $\mathbf{P}(A, L, t\mathbf{a})$ if $t > 0$ or if $t \geq 0$ and $\mathbf{P}$ is a polytope. Now, let $\mathbf{P}_2 = \bigcup_{t\in[0,1]} \mathbf{P}(A, L, t\mathbf{a})$. It is not hard to see that $\mathbf{P}_2$ can be represented by the sop

$$\mathbf{P}_2 = \mathbf{P}\left(\begin{pmatrix} A \\ \mathbf{0}^T \\ \mathbf{0}^T \end{pmatrix}, \begin{pmatrix} L & -\mathbf{a} \\ \mathbf{0}^T & 1 \\ \mathbf{0}^T & -1 \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ 1 \\ 0 \end{pmatrix}\right).$$

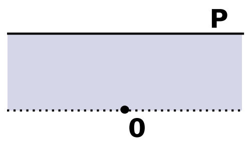
Hence, for polytopes it is possible to express the set $\operatorname{conv}(\{\mathbf{0}\} \cup \mathbf{P})$ as a sop.

We shall discuss the case where $\mathbf{P}$ is unbounded. Let us look at the following example.

**Example 3.23 (The Convex Hull of Polyhedra is Not Necessarily Closed)**
Let $\mathbf{P}$ be the unbounded line $\mathbf{P} = \{\binom{x}{1} \mid x \in \mathbb{K}\} \subset \mathbb{K}^2$. Then $\operatorname{conv}(\{\mathbf{0}\} \cup \mathbf{P})$ is the set $\mathbf{P}_1 = \{\binom{x}{y} \mid x \in \mathbb{K}, 0 < y \leq 1\} \cup \{\binom{0}{0}\}$. This set is convex but not closed. A set that is not closed can neither be described as an $\mathcal{H}$-polyhedron, nor as a projection of an $\mathcal{H}$-polyhedron.

Now, let us look at representations in terms of sops. $\mathbf{P}$ can be represented as the sop $\mathbf{P}(A, L, \mathbf{a})$ with $A = \begin{pmatrix} 0 & 1 \\ 0 & -1 \end{pmatrix}$, $L = \emptyset$, and $\mathbf{a} = \binom{1}{-1}$. The set $\mathbf{P}_2 = \bigcup_{t\in[0,1]} \mathbf{P}(A, L, t\mathbf{a})$ is given by

$$\begin{aligned}\mathbf{P}_2 = \mathbf{P}&\left(\begin{pmatrix} A \\ \mathbf{0}^T \\ \mathbf{0}^T \end{pmatrix}, \begin{pmatrix} -\mathbf{a} \\ 1 \\ -1 \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ 1 \\ 0 \end{pmatrix}\right) \\ &= \left\{\begin{pmatrix} x \\ y \end{pmatrix} \,\middle|\, y = \lambda,\ 0 \leq \lambda \leq 1\right\} \\ &= \left\{\begin{pmatrix} x \\ y \end{pmatrix} \,\middle|\, 0 \leq y \leq 1\right\}.\end{aligned}$$



Interestingly, this is the smallest closed set containing the convex hull of $\{\mathbf{0}\} \cup \mathbf{P}$. ∎

The example shows that, while the convex hull of polyhedra cannot be represented as a sop in general, it might be possible to represent at least their *closed* convex hull as a sop. To see that this is actually always possible, we need some more theoretical insight, which shall be provided in the next sections.

## 3.8. Half-Space Cones

The normal vector $\mathbf{n}$ and the coefficient $c$ of the half-space representation $\mathbf{H}(\mathbf{n}, c) = \{\mathbf{x} \mid \mathbf{n}^T\mathbf{x} \leq c\}$ form the vector $\binom{\mathbf{n}}{c} \in \mathbb{K}^{d+1}$. For any closed convex set $\mathbf{S}$ this allow us to define the set

$$\mathbf{C} = \left\{ \begin{pmatrix} \mathbf{n} \\ c \end{pmatrix} \middle| \mathbf{S} \subseteq \mathbf{H}(\mathbf{n}, c) \right\}$$

that contains all vectors $\binom{\mathbf{n}}{c}$ which represent an including half-space of $\mathbf{S}$. It is an easy observation that $\mathbf{C}$ is closed under conical combinations. The set $\mathbf{C}$ is called the *subsumption cone* of $\mathbf{S}$ (Lassez, 1990). On the other hand, it is well-known that any closed convex set $\mathbf{S}$ is given as the intersection of all including closed half-spaces. We are observing a mutual dependency between the closed convex set $\mathbf{S}$ and the subsumption cone $\mathbf{C}$:

$$\mathbf{S} = \bigcap_{\mathbf{H}(\mathbf{n},c) \in \mathbf{C}} \mathbf{H}(\mathbf{n}, c) \quad \text{and} \quad \mathbf{C} = \left\{ \begin{pmatrix} \mathbf{n} \\ c \end{pmatrix} \middle| \mathbf{S} \subseteq \mathbf{H}(\mathbf{n}, c) \right\}. \tag{3.8}$$

We can exploit the interdependence in different directions:

(i) Assume, the closed convex set $\mathbf{S}$ is given explicitly, for example as an $\mathcal{H}$-polyhedron. Then (3.8) yields a definition of the subsumption cone $\mathbf{C}$.

(ii) Or, we assume that $\mathbf{C}$ is given explicitly. Then (3.8) yields a definition of $\mathbf{S}$.

Let us analyze the second case: Since $\mathbf{C}$ defines the set $\mathbf{S}$ via (3.8), we may say that $\mathbf{C}$ represents $\mathbf{S}$. Furthermore, even if $\mathbf{C}$ is not the subsumption cone of $\mathbf{S}$, it may still represent $\mathbf{S}$: Assume, $\mathbf{C}$ is a finite sets of vectors of the form $\binom{\mathbf{n}_i}{c_i}$, $i = 1, \ldots, m$. Then $\mathbf{S}$ is a closed convex polyhedron. Actually, this is nothing else than an alternative way of writing

$$\mathbf{S} = \mathbf{P}\left(A, \mathbf{a}\right) \text{ with } A = \begin{pmatrix} \mathbf{n}_1^T \\ \mathbf{n}_2^T \\ \vdots \\ \mathbf{n}_m^T \end{pmatrix} \text{ and } \mathbf{a} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{pmatrix}.$$

Instead of finite sets, this section deals with *conical* sets $\mathbf{C}$ which represent polyhedral sets $\mathbf{S}$. For simplicity, we use the notation $\mathbf{H}(\mathbf{n}, c) \in \mathbf{C}$ to indicate that we identify $\mathbf{H}(\mathbf{n}, c)$ with the vector $\binom{\mathbf{n}}{c} \in \mathbf{C}$ in the following.

**Definition 3.24 (Half-Space Cone)**
Let $\mathbf{P}$ be a polyhedron in $\mathbb{K}^d$. Any conical set $\mathbf{C} \in \mathbb{K}^{d+1}$ with the following properties

(i) for all $\mathbf{H}(\mathbf{n}, c) \in \mathbf{C}$ it holds that $\mathbf{P}$ is a subset of $\mathbf{H}(\mathbf{n}, c)$,

(ii) for all supporting half-spaces $\mathbf{H}(\mathbf{n}, c)$ of $\mathbf{P}$ it holds that $\mathbf{H}(\mathbf{n}, c) \in \mathbf{C}$,

(iii) if $\mathbf{P}$ is empty, then $\mathbf{H}(\mathbf{0}, -1) \in \mathbf{C}$,

is called a *half-space cone* of $\mathbf{P}$. ■

Clearly, the subsumption cone of $\mathbf{P}$ is a half-space cone of $\mathbf{P}$. Since the empty set is subset of any set, every half-space cone of $\mathbf{P}$ is a subset of the subsumption cone of $\mathbf{P}$, even for an empty polyhedron $\mathbf{P}$. The following proposition and its corollary illustrate how half-space cones help us to characterize polyhedra.

**Proposition 3.25 (Representation)**
Let $\mathbf{P}$ be a polyhedron and $\mathbf{C}$ be a half-space cone of $\mathbf{P}$. Then the following equivalence holds:

$$\mathbf{x} \in \mathbf{P} \text{ if and only if } \mathbf{x} \in \mathbf{H}(\mathbf{n}, c) \text{ for all } \mathbf{H}(\mathbf{n}, c) \in \mathbf{C}. \qquad \blacksquare$$

*Proof.* Assume $\mathbf{x} \in \mathbf{P}$. Every half-space $\mathbf{H}(\mathbf{n}, c) \in \mathbf{C}$ includes $\mathbf{P}$ by definition. Hence, $\mathbf{x} \in \mathbf{H}(\mathbf{n}, c)$ for all $\mathbf{H}(\mathbf{n}, c) \in \mathbf{C}$.

Conversely, suppose $\mathbf{x} \in \mathbf{H}$ for all half-spaces $\mathbf{H} \in \mathbf{C}$, but $\mathbf{x} \notin \mathbf{P}$. Suppose $\mathbf{P}$ is empty, then $\mathbf{H}(\mathbf{0}, -1) \in \mathbf{C}$ by definition, and we have the contradiction $\mathbf{x} \in \mathbf{H}(\mathbf{0}, -1)$. Hence, $\mathbf{P}$ is not empty. Since every non-empty polyhedron can be represented as a non-empty intersection of finitely many half-spaces, there must be a half-space $\mathbf{H}_0 = \mathbf{H}(\mathbf{n}_0, c_0)$ such that $\mathbf{P} \subseteq \mathbf{H}_0$ and $\mathbf{x} \notin \mathbf{H}_0$. Sharpening $c_0$ to the optimal value $h_{\mathbf{P}}(\mathbf{n}_0)$ only decreases the value of $c_0$. Therefore, the relations $\mathbf{P} \subseteq \mathbf{H}(\mathbf{n}_0, h_{\mathbf{P}}(\mathbf{n}_0))$ and $\mathbf{x} \notin \mathbf{H}(\mathbf{n}_0, h_{\mathbf{P}}(\mathbf{n}_0))$ still hold for the supporting half-space $\mathbf{H}(\mathbf{n}_0, h_{\mathbf{P}}(\mathbf{n}_0))$. But, by definition, $\mathbf{H}(\mathbf{n}_0, h_{\mathbf{P}}(\mathbf{n}_0)) \in \mathbf{C}$, which is a contradiction to the supposition $\mathbf{x} \in \mathbf{H}$ for all $\mathbf{H} \in \mathbf{C}$. $\square$

**Corollary 3.26 (Equality)**
Let $\mathbf{P}$ and $\mathbf{Q}$ be two polyhedra, and let $\mathbf{C}$ be a half-space cone of $\mathbf{P}$. If $\mathbf{C}$ is also a half-space cone of $\mathbf{Q}$, then $\mathbf{P} = \mathbf{Q}$. $\blacksquare$

*Proof.* Let $\mathbf{C}$ be a half-space cone of $\mathbf{P}$ and $\mathbf{Q}$. Then we have the following equivalences:

$$\mathbf{x} \in \mathbf{P} \iff \mathbf{x} \in \mathbf{H}(\mathbf{n}, c) \text{ for all } \mathbf{H}(\mathbf{n}, c) \in \mathbf{C} \iff \mathbf{x} \in \mathbf{Q}. \qquad \square$$

There is a purely syntactical construction on the $\mathcal{H}$-representation that yields half-space cones.

**Definition 3.27 (Cone of Implied Half-Spaces)**
Let $\mathbf{P} = \mathbf{P}(A, \mathbf{a})$ be an $\mathcal{H}$-polyhedron. Then

$$\mathbf{C}(A, \mathbf{a}) = \left\{ \begin{pmatrix} \mathbf{n} \\ c \end{pmatrix} \,\middle|\, \exists \mathbf{u} \geq \mathbf{0}\colon\ \mathbf{n} = A^T \mathbf{u}, c = \mathbf{a}^T \mathbf{u} \right\},$$

is called the *cone of implied half-spaces* of $\mathbf{P}$. $\blacksquare$

**Proposition 3.28 (Cone of Implied Half-Spaces is a Half-Space Cone)**
Let $\mathbf{P} = \mathbf{P}(A, \mathbf{a})$ be an $\mathcal{H}$-polyhedron in $\mathbb{K}^d$ and $\mathbf{C}(A, \mathbf{a}) \in \mathbb{K}^{d+1}$ its cone of implied half-spaces. Then $\mathbf{C}(A, \mathbf{a})$ is a half-space cone of $\mathbf{P}$. $\blacksquare$

*Proof.* At first we note that the cone $\mathbf{C}(A, \mathbf{a})$ is generated by the rows of the matrix $(A \quad \mathbf{a})$. We have to show that the conditions (i), (ii), and (iii) of Definition 3.24 hold.

(i) Let $\mathbf{H}(\mathbf{n}, c) \in \mathbf{C}(A, \mathbf{a})$. Then there exists some $\mathbf{u} \geq \mathbf{0}$ with $\mathbf{n} = A^T\mathbf{u}$ and $c = \mathbf{a}^T\mathbf{u}$. For every $\mathbf{x} \in \mathbf{P}$, i. e., $A\mathbf{x} \leq \mathbf{a}$, it follows $\mathbf{u}^T A\mathbf{x} \leq \mathbf{u}^T\mathbf{a}$, i. e., $\mathbf{n}^T\mathbf{x} \leq c$. Hence, $\mathbf{P} \subseteq \mathbf{H}(\mathbf{n}, c)$.

(ii) If $\mathbf{H}(\mathbf{n}, c)$ is a supporting half-space of $\mathbf{P}$, then $c = h_{\mathbf{P}}(\mathbf{n})$. The value $h_{\mathbf{P}}(\mathbf{n})$ is given by the optimal value of the linear program

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

By Strong Duality (Theorem 2.8), the dual linear program

$$\text{minimize } \mathbf{a}^T\mathbf{u} \text{ subject to } A^T\mathbf{u} = \mathbf{n},\ \mathbf{u} \geq \mathbf{0}$$

has also an optimal solution $\tilde{\mathbf{u}}$ with $\tilde{\mathbf{u}} \geq \mathbf{0}$, $\mathbf{n} = A^T\tilde{\mathbf{u}}$ and $c = \mathbf{a}^T\tilde{\mathbf{u}}$. Hence, $\mathbf{H}(\mathbf{n}, c) \in \mathbf{C}(A, \mathbf{a})$.

(iii) Suppose, $\mathbf{P}$ is empty. Then $A\mathbf{x} \leq \mathbf{a}$ has no solution and, by Farkas' Lemma (Theorem 2.3), there exists some $\mathbf{u} \geq \mathbf{0}$ with $A^T\mathbf{u} = \mathbf{0}$ and $\mathbf{a}^T\mathbf{u} < 0$. Hence, there is some positive factor $\lambda$ such that for $\mathbf{u}' = \lambda\mathbf{u}$ the relations $\mathbf{u}' \geq \mathbf{0}$, $A^T\mathbf{u}' = \mathbf{0}$, and $\mathbf{a}^T\mathbf{u}' = -1$ hold. That is, $\mathbf{H}(\mathbf{0}, -1) \in \mathbf{C}(A, \mathbf{a})$. □

While two different $\mathcal{H}$-representations $\mathbf{P}(A, \mathbf{a})$ and $\mathbf{P}(A', \mathbf{a}')$ of a polyhedron $\mathbf{P}$ describe exactly the same set of points in $\mathbb{K}^d$, the cones $\mathbf{C}(A, \mathbf{a})$ and $\mathbf{C}(A', \mathbf{a}')$ may be different sets in $\mathbb{K}^{d+1}$. At least, $\mathbf{C}(A, \mathbf{a})$ and $\mathbf{C}(A', \mathbf{a}')$ are both subsets of the unique subsumption cone of $\mathbf{P}$. We discuss the *completion* of a cone of implied half-spaces, a simple modification of the $\mathcal{H}$-representation, which yields the subsumption cone. First, we define the notion of completeness.

**Definition 3.29 (Completeness)**
Let $\mathbf{C}$ be a half-space cone of a polyhedron $\mathbf{P}$. We call $\mathbf{C}$ *complete* if $\mathbf{H}(\mathbf{0}, 1) \in \mathbf{C}$. Additionally, we call any $\mathcal{H}$-polyhedron $\mathbf{P}(A, \mathbf{a})$ *complete* if its cone of implied half-spaces $\mathbf{C}(A, \mathbf{a})$ is complete. ■

Hence, $\mathbf{P}(A, \mathbf{a})$ is complete if and only if there exists some $\mathbf{u} \geq \mathbf{0}$ with $A^T\mathbf{u} = \mathbf{0}$ and $\mathbf{a}^T\mathbf{u} = 1$. With other words, the completeness of $\mathbf{P}(A, \mathbf{a})$ only depends on the augmented matrix $(A \quad \mathbf{a})$.

Complete half-space cones have an important property that will help us in the upcoming proofs. In order to show that a certain half-space $\mathbf{H}(\mathbf{n}, c)$ is in a complete half-space cone $\mathbf{C}$, it suffices to find some $\mathbf{H}(\mathbf{n}, c') \in \mathbf{C}$ with $c' \leq c$: Let $c - c' = \mu$. Since $\mathbf{H}(\mathbf{0}, 1)$ is in the cone $\mathbf{C}$, it holds that $\mathbf{H}(\mathbf{n}, c) = \mathbf{H}(\mathbf{n}, c' + \mu) = \mathbf{H}(\mathbf{n}, c') + \mu\mathbf{H}(\mathbf{0}, 1)$ is also in $\mathbf{C}$.

**Proposition 3.30 (Complete Half-Space Cones are Subsumption Cones)**
Let $\mathbf{P}$ be a non-empty polyhedron. A complete half-space cone of $\mathbf{P}$ is a subsumption cone of $\mathbf{P}$. ■

*Proof.* Let $\mathbf{C}$ be a complete half-space cone of a non-empty polyhedron $\mathbf{P}$. Further, let $\mathbf{H}(\mathbf{n}, c)$ be a half-space with $\mathbf{P} \subseteq \mathbf{H}(\mathbf{n}, c)$. We have to show that $\mathbf{H}(\mathbf{n}, c) \in \mathbf{C}$. Since

$\mathbf{C}$ is a half-space cone, it contains at least the supporting half-space $\mathbf{H}(\mathbf{n}, h_{\mathbf{P}}(\mathbf{n}))$ where $h_{\mathbf{P}}(\mathbf{n}) \leq c$. Either $h_{\mathbf{P}}(\mathbf{n}) = c$ and we are done, or $h_{\mathbf{P}}(\mathbf{n}) < c$. $\mathbf{C}$ is complete and contains $\mathbf{H}(\mathbf{0}, 1)$. Then there exists a conical combination $\mathbf{H}(\mathbf{n}, h_{\mathbf{P}}(\mathbf{n})) + \lambda \mathbf{H}(\mathbf{0}, 1) = \mathbf{H}(\mathbf{n}, c)$ with $\lambda > 0$. $\mathbf{C}$ is closed under conical combinations and, hence, $\mathbf{H}(\mathbf{n}, h_{\mathbf{P}}(\mathbf{n})) \in \mathbf{C}$. □

Note that the preceding proposition only holds for non-empty polyhedra. In case of an empty polyhedron $\mathbf{P} \in \mathbb{K}^d$, the subsumption cone is equal to the set $\mathbb{K}^d \times \mathbb{K}$. Further below we will introduce the normal cones which help to deal with empty polyhedra.

The cone of implied half-spaces is complete for a large subclass of polyhedra. This subclass includes all bounded polyhedra in $\mathbb{K}^d$, $d \geq 1$.

**Proposition 3.31 (Polyhedra with Complete Cones)**
Let $\mathbf{P}(A, \mathbf{a})$ be a $\mathcal{H}$-polyhedron and $\mathbf{C}(A, \mathbf{a})$ its cone of implied half-spaces. If for at least one $\mathbf{n} \in \mathbb{K}^d$ the inequality

$$\infty > h_{\mathbf{P}}(\mathbf{n}) > -h_{\mathbf{P}}(-\mathbf{n}) > -\infty$$

holds, then $\mathbf{C}(A, \mathbf{a})$ is complete. ■

*Proof.* Let $\mathbf{n}$ be given as in the proposition. The values of $h_{\mathbf{P}}(\mathbf{n})$ and $h_{\mathbf{P}}(-\mathbf{n})$ are given by the optimal solutions $\mathbf{u}_1$ and $\mathbf{u}_2$ of the (dual) linear programs

$$\text{minimize } \mathbf{a}^T\mathbf{u} \text{ subject to } A^T\mathbf{u} = \mathbf{n},\ \mathbf{u} \geq 0,$$

and

$$\text{minimize } \mathbf{a}^T\mathbf{u} \text{ subject to } A^T\mathbf{u} = -\mathbf{n},\ \mathbf{u} \geq 0,$$

respectively. Now, for the conical combination $\mathbf{u}_0 = \mathbf{u}_1 + \mathbf{u}_2$ it holds $A^T\mathbf{u}_0 = A^T\mathbf{u}_1 + A^T\mathbf{u}_2 = \mathbf{n} - \mathbf{n} = \mathbf{0}$ and $\mathbf{a}^T\mathbf{u}_0 = \mathbf{a}^T\mathbf{u}_1 + \mathbf{a}^T\mathbf{u}_2 = h_{\mathbf{P}}(\mathbf{n}) + h_{\mathbf{P}}(-\mathbf{n}) > 0$. Finally, scaling with a positive factor yields $\binom{\mathbf{0}}{1} \in \mathbf{C}(A, \mathbf{a})$. □

The following example shows that a polyhedron does not need to be complete in general.

**Example 3.32**
Let $\mathbf{P} = \mathbf{P}(1, 1)$ be the $\mathcal{H}$-polyhedron which corresponds to the set $\{x \mid x \leq 1\} \subset \mathbb{K}$. The cone of implied half-spaces is the set

$$\mathbf{C}(A, \mathbf{a}) = \operatorname{cone} \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{3.9}$$

The half-space $\mathbf{H}(1, 2) = \{x \mid x \leq 2\}$ is an including half-space, but it is obviously not a half-space in $\mathbf{C}(A, \mathbf{a})$. ■

The completeness of a polyhedron $\mathbf{P}(A, \mathbf{a})$ is easily checked by linear programming: For any polyhedron $\mathbf{P}(A, \mathbf{a})$ the linear program

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

is feasible if and only if $\mathbf{P}(A, \mathbf{a})$ is complete. Now, if $\mathbf{P}(A, \mathbf{a})$ is not complete, we may complete it by adding the redundant linear inequality $\mathbf{0}^T\mathbf{x} \leq 1$:[2]

**Proposition 3.33 (Completion)**
Let $\mathbf{P}(A, \mathbf{a})$ an $\mathcal{H}$-representation of a polyhedron $\mathbf{P}$ and let $B = \binom{A}{\mathbf{0}^T}$, $\mathbf{b} = \binom{\mathbf{a}}{1}$. Then $\mathbf{P} = \mathbf{P}(A, \mathbf{a}) = \mathbf{P}(B, \mathbf{b})$ and $\mathbf{P}(B, \mathbf{b})$ is complete. ■

*Proof.* Obvious. □

## 3.9. Half-Space Cones of Symbolic Orthogonal Projections

In the preceding section, we have defined half-space cones of $\mathcal{H}$-polyhedra. A certain subclass, the cones of implied half-spaces, was easily constructed from the $\mathcal{H}$-representation of polyhedra. Now, we exploit the underlying $\mathcal{H}$-representation of sops and construct cones of implied half-spaces of sops in a similar way.

**Proposition 3.34 (Implied Half-Space Cone)**
Let $\mathbf{P} = \mathbf{P}(A, L, \mathbf{a}) \in \mathbb{K}^d$ be a symbolic orthogonal projection of the $\mathcal{H}$-polyhedron $\mathbf{Q} = \mathbf{P}((A \quad L), \mathbf{a})$ in $\mathbb{K}^{d+k}$. Then

$$\mathbf{C}(A, L, \mathbf{a}) = \left\{ \begin{pmatrix} \mathbf{n} \\ c \end{pmatrix} \middle| \exists \mathbf{u} \geq \mathbf{0} \colon \mathbf{n} = A^T\mathbf{u}, \mathbf{0} = L^T\mathbf{u}, c = \mathbf{a}^T\mathbf{u} \right\}.$$

is a half-space cone of $\mathbf{P}$, and we call $\mathbf{C}(A, L, \mathbf{a})$ the cone of implied half-spaces of $\mathbf{P}$. ■

*Proof.* From Definition 3.1 and Proposition 3.28 immediately follows that

$$\mathbf{C}\left((A \quad L), \mathbf{a}\right) = \left\{ \begin{pmatrix} \mathbf{n} \\ \mathbf{m} \\ c \end{pmatrix} \middle| \exists \mathbf{u} \geq \mathbf{0} \colon \mathbf{n} = A^T\mathbf{u},\ \mathbf{m} = L^T\mathbf{u},\ c = \mathbf{a}^T\mathbf{u} \right\}$$

is the cone of implied half-spaces of $\mathbf{Q}$. The set $\mathbf{C}(A, L, \mathbf{a})$ can be interpreted as a restriction of the cone $\mathbf{C}\left((A \quad L), \mathbf{a}\right)$ where we limit the choice of $\mathbf{u}$ to $L^T\mathbf{u} = \mathbf{0}$. This set is again a cone: If $\mathbf{u}_1 \geq \mathbf{0}$, $L^T\mathbf{u}_1 = \mathbf{0}$ and $\mathbf{u}_2 \geq \mathbf{0}$, $L^T\mathbf{u}_2 = \mathbf{0}$, then for $\lambda_1 \geq 0$ and $\lambda_2 \geq 0$ we have $\lambda_1\mathbf{u}_1 + \lambda_2\mathbf{u}_2 \geq \mathbf{0}$ and $L^T(\lambda_1\mathbf{u}_1 + \lambda_2\mathbf{u}_2) = \mathbf{0}$. It remains to show that conditions (i), (ii), and (iii) of Definition 3.24 hold for $\mathbf{C}(A, L, \mathbf{a})$.

(i) Let $\mathbf{H}(\mathbf{n}, c) \in \mathbf{C}(A, L, \mathbf{a})$. Then there exists some $\mathbf{u} \geq \mathbf{0}$ with $A^T\mathbf{u} = \mathbf{n}$, $L^T\mathbf{u} = \mathbf{0}$, and $\mathbf{a}^T\mathbf{u} = c$. We have to show that $\mathbf{H}(\mathbf{n}, c)$ is an including half-space of $\mathbf{P}(A, L, \mathbf{a})$. For this purpose, let $\mathbf{x} \in \mathbf{P}$. Then there exists some $\mathbf{z}$ such that $\binom{\mathbf{x}}{\mathbf{z}} \in \mathbf{Q}$ or, equivalently, $A\mathbf{x} + L\mathbf{z} \leq \mathbf{a}$. It follows $\mathbf{u}^TA\mathbf{x} + \mathbf{u}^TL\mathbf{z} \leq \mathbf{u}^T\mathbf{a}$ and, since $\mathbf{u}^TL = \mathbf{0}^T$, also $\mathbf{u}^TA\mathbf{x} \leq \mathbf{u}^T\mathbf{a}$, which is equivalent to $\mathbf{n}^T\mathbf{x} \leq c$, and, hence, $\mathbf{x} \in \mathbf{H}(\mathbf{n}, c)$.

[2] I would like to thank Ernst Althaus who pointed me to the idea of adding the trivial constraint $\mathbf{0}^T\mathbf{x} \leq 1$ at an early stage of the development of the sops.

(ii) We have to show that any supporting half-space $\mathbf{H}(\mathbf{n}, c)$, $c = h_{\mathbf{P}}(\mathbf{n})$, of $\mathbf{P}$ is in $\mathbf{C}(A, L, \mathbf{a})$. We use the equality $c = h_{\mathbf{P}}(\mathbf{n}) = h_{\mathbf{Q}}\binom{\mathbf{n}}{\mathbf{0}}$. The supporting half-space $\mathbf{H}(\binom{\mathbf{n}}{\mathbf{0}}, c)$ is in $\mathbf{C}((A \quad L), \mathbf{a})$. That is, there exists some $\mathbf{u} \geq \mathbf{0}$ with $A^T\mathbf{u} = \mathbf{n}$, $L^T\mathbf{u} = \mathbf{0}$, and $\mathbf{a}^T\mathbf{u} = c$. Hence, $\mathbf{H}(\mathbf{n}, c) \in \mathbf{C}(A, L, \mathbf{a})$.

(iii) We have to show that $\mathbf{P}$ is empty if and only if $\mathbf{H}(\mathbf{0}, -1) \in \mathbf{C}(A, L, \mathbf{a})$. Suppose, $\mathbf{P}$ is empty. Then $\mathbf{Q}$ is also empty, and there is some $\mathbf{u}$ such that $A^T\mathbf{u} = \mathbf{0}$, $L^T\mathbf{u} = \mathbf{0}$, and $\mathbf{a}^T\mathbf{u} = -1$. Hence, $\mathbf{H}(\mathbf{0}, -1) \in \mathbf{C}(A, L, \mathbf{a})$. Conversely, if $\mathbf{H}(\mathbf{0}, -1) \in \mathbf{C}(A, L, \mathbf{a})$ then $(\mathbf{0}, \mathbf{0}, -1)^T \in \mathbf{C}((A \quad L), \mathbf{a})$. Hence, $\mathbf{Q}$ is empty. The orthogonal projection $\mathbf{P}$ of an empty set is empty again. □

The previous proposition extends the one-to-one relation between $\mathbf{P}((A \quad L), \mathbf{a})$ and $\mathbf{C}((A \quad L), \mathbf{a})$ to an one-to-one relation between $\mathbf{P}(A, L, \mathbf{a})$ and $\mathbf{C}(A, L, \mathbf{a})$. The notion of completeness of a half-space cone also carries over. $\mathbf{P}(A, L, \mathbf{a})$ and $\mathbf{C}(A, L, \mathbf{a})$, respectively, are complete if and only if $\mathbf{H}(\mathbf{0}, 1) \in \mathbf{C}(A, L, \mathbf{a})$. This justifies the following definition:

**Definition 3.35 (Completeness)**
The sop $\mathbf{P}(A, L, \mathbf{a})$ is *complete* if and only if there exists some $\mathbf{u} \geq \mathbf{0}$ with $\mathbf{0} = A^T\mathbf{u}$, $\mathbf{0} = L^T\mathbf{u}$, and $1 = \mathbf{a}^T\mathbf{u}$. ■

As before, any sop can be completed by adding the redundant row $(\mathbf{0}^T, \mathbf{0}^T, 1)$ to its representation $(A, L, \mathbf{a})$.[3]

Most proofs of the next section are based on the following proposition.

**Proposition 3.36 (Equality)**
Let $\mathbf{P}(A, L, \mathbf{a})$ be a symbolic orthogonal projection in $\mathbb{K}^d$ and $\mathbf{Q}$ a polyhedron in $\mathbb{K}^d$. Then $\mathbf{P}(A, L, \mathbf{a}) = \mathbf{Q}$ if and only if the following conditions hold:

(i) for all $\mathbf{H} \in \mathbf{C}(A, L, \mathbf{a})$ it holds that $\mathbf{Q} \subseteq \mathbf{H}$,

(ii) for every supporting half-space $\mathbf{H}$ of $\mathbf{Q}$ it holds that $\mathbf{H} \in \mathbf{C}(A, L, \mathbf{a})$,

(iii) if $\mathbf{Q} = \emptyset$ then $\mathbf{H}(\mathbf{0}, -1) \in \mathbf{C}(A, L, \mathbf{a})$. ■

*Proof.* Let $\mathbf{C}(A, L, \mathbf{a})$ be a half-space cone of $\mathbf{P}(A, L, \mathbf{a})$. Conditions (i)-(iii) are exactly the conditions of Definition 3.24 that show that $\mathbf{C}(A, L, \mathbf{a})$ is a half-spaces cone of $\mathbf{Q}$. Clearly, if $\mathbf{P}(A, L, \mathbf{a}) = \mathbf{Q}$, then conditions (i)-(iii) follow. On the other hand, if conditions (i)-(iii) hold, then Corollary 3.26 establishes the equality. □

Finally, we define the normal cone[4] of a sop. The normal cone is an auxiliary construction that will help us to deal with empty polyhedra.

[3] One can show that any sop $\mathbf{P}$ which represents a full-dimensional polytope in $\mathbb{K}^d$ with $d \geq 1$ is complete, see Proposition 3.31.

[4] Usually, the term normal cone is used with a slightly different meaning, see e. g. the usage in Ziegler (1995).

**Definition 3.37 (Normal Cone)**
Let $\mathbf{P} = \mathbf{P}(A, L, \mathbf{a})$ be a sop. Then the set

$$\mathbf{N}(A, L) = \left\{ \mathbf{n} \,\middle|\, \exists \mathbf{u} \geq \mathbf{0}\colon\ \mathbf{n} = A^T\mathbf{u}, \mathbf{0} = L^T\mathbf{u} \right\}$$

is called the *normal cone* of the polyhedron $\mathbf{P}$. ■

Clearly, for any sop $\mathbf{P}(A, L, \mathbf{a}) \subseteq \mathbb{K}^d$ the set $\mathbf{N}(A, L)$ is a cone. It can be regarded as the orthogonal projection of $\mathbf{C}(A, L, \mathbf{a}) \subseteq \mathbb{K}^{d+1}$ onto the first $d$ components. Note that the normal cone $\mathbf{N}(A, L)$ is empty if and only if $A$ is an empty matrix. Otherwise, it always contains $\mathbf{0}$. Hence, even sops representing the empty set have a normal cone. If $\mathbf{P}(A, L, \mathbf{a}) \subseteq \mathbb{K}^d$ is a polytope, i. e., bounded in every direction, then $\mathbf{N}(A, L) = \mathbb{K}^d$.

**Lemma 3.38**
Let $\mathbf{P} = \mathbf{P}(A, L, \mathbf{a})$ be an empty but complete sop. Further, let $\mathbf{C}(A, L, \mathbf{a})$ its cone of implied half-spaces and $\mathbf{N}(A, L)$ be the normal cone of $\mathbf{P}$. Then $\mathbf{C}(A, L, \mathbf{a})$ is not necessarily the subsumption cone of $\mathbf{P}$, i. e., not all implied half-spaces of $\mathbf{P}$ are contained in $\mathbf{C}(A, L, \mathbf{a})$. But at least for any $\mathbf{n} \in \mathbf{N}(A, L)$ and any $c$ we have $\mathbf{H}(\mathbf{n}, c) \in \mathbf{C}(A, L, \mathbf{a})$.■

*Proof.* An empty set has no supporting half-spaces. Hence, it suffices that $\mathbf{C}(A, L, \mathbf{a})$ contains the half-space $\mathbf{H}(\mathbf{0}, -1)$ to represent an empty set and $\mathbf{H}(\mathbf{0}, 1)$ to be complete.
If the normal cone contains any further normal vectors $\mathbf{n} \neq \mathbf{0}$, i. e., there exists some $\mathbf{u} \geq \mathbf{0}$ with $A^T\mathbf{u} = \mathbf{n}$ and $L^T\mathbf{u} = \mathbf{0}$, then we can combine any half-space $\mathbf{H}(\mathbf{n}, c)$ from $\mathbf{H}(\mathbf{n}, \mathbf{a}^T\mathbf{u})$ by adding a multiple either of $\mathbf{H}(\mathbf{0}, 1)$ or $\mathbf{H}(\mathbf{0}, -1)$. □

## 3.10. Convex Hull

We show that symbolic orthogonal projections allow an efficient representation of the closed convex hull.

**Definition 3.39 (Closed Convex Hull)**
Let $\mathbf{P}_1$ and $\mathbf{P}_2$ two sets. Then $\operatorname{clconv}(\mathbf{P}_1 \cup \mathbf{P}_2)$ is the smallest closed convex set that contains $\mathbf{P}_1$ and $\mathbf{P}_2$. ■

Note that this definition implies $\mathbf{P}_1 \subseteq \operatorname{clconv}(\mathbf{P}_1 \cup \emptyset) = \operatorname{clconv}(\emptyset \cup \mathbf{P}_1)$ for any set $\mathbf{P}_1$.

**Proposition 3.40 (Closed Convex Hull)**
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 complete sops. Further, let

$$A = \begin{pmatrix} A_1 \\ \mathrm{O} \end{pmatrix}, \quad L = \begin{pmatrix} A_1 & L_1 & \mathrm{O} & \mathbf{a}_1 \\ -A_2 & \mathrm{O} & L_2 & -\mathbf{a}_2 \end{pmatrix}, \quad \text{and } \mathbf{a} = \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{0} \end{pmatrix}.$$

Then the following properties hold:

- $\operatorname{clconv}(\mathbf{P}_1 \cup \mathbf{P}_2) \subseteq \mathbf{P}(A, L, \mathbf{a})$, and $\mathbf{P}(A, L, \mathbf{a})$ is complete.
- If neither $\mathbf{P}_1$ nor $\mathbf{P}_2$ is empty, then $\mathbf{P}(A, L, \mathbf{a}) = \operatorname{clconv}(\mathbf{P}_1 \cup \mathbf{P}_2)$.

- Otherwise, the following equation holds for the support function of $\mathbf{P}$:

$$h_{\mathbf{P}}(\mathbf{n}) = \begin{cases} h_{\mathrm{clconv}(\mathbf{P}_1 \cup \mathbf{P}_2)}(\mathbf{n}) & \text{if } \mathbf{n} \in \mathbf{N}(A_1, L_1) \cap \mathbf{N}(A_2, L_2) \\ \infty & \text{otherwise.} \end{cases} \qquad \blacksquare$$

*Proof.* The proof is based on the equality criteria in Proposition 3.36.

(i) In order to establish the relation $\mathrm{clconv}(\mathbf{P}_1 \cup \mathbf{P}_2) \subseteq \mathbf{P}(A, L, \mathbf{a})$, we have to show that for all $\mathbf{H} \in \mathbf{C}(A, L, \mathbf{a})$ it holds that $\mathrm{clconv}(\mathbf{P}_1 \cup \mathbf{P}_2) \subseteq \mathbf{H}$. Any half-space $\mathbf{H} = \mathbf{H}(\mathbf{n}, c) \in \mathbf{C}(A, L, \mathbf{a})$ is given by some $\mathbf{u} \geq \mathbf{0}$ with

$$\mathbf{0} = L^T\mathbf{u},\ \mathbf{n} = A^T\mathbf{u}, \ \text{ and } c = \mathbf{a}^T\mathbf{u}.$$

After appropriately partitioning $\mathbf{u} = \binom{\mathbf{u}_1}{\mathbf{u}_2}$, $\mathbf{u}_1 \geq \mathbf{0}$, $\mathbf{u}_2 \geq \mathbf{0}$, the former system is equivalent to

$$\begin{aligned} \mathbf{0} &= L_1^T\mathbf{u}_1, & \mathbf{n} &= A_1^T\mathbf{u}_1, & c &= \mathbf{a}_1^T\mathbf{u}_1, \\ \mathbf{0} &= L_2^T\mathbf{u}_2, & \mathbf{n} &= A_2^T\mathbf{u}_1, & c &= \mathbf{a}_2^T\mathbf{u}_2. \end{aligned} \tag{3.10}$$

System (3.10) shows that $\mathbf{H}(\mathbf{n}, c) \in \mathbf{C}(A_1, L_1, \mathbf{a}_1)$ and $\mathbf{H}(\mathbf{n}, c) \in \mathbf{C}(A_2, L_2, \mathbf{a}_2)$. That is, $\mathbf{P}_1 \subseteq \mathbf{H}$, $\mathbf{P}_2 \subseteq \mathbf{H}$, and, hence, $\mathrm{clconv}(\mathbf{P}_1 \cup \mathbf{P}_2) \subseteq \mathbf{H}$. In addition, since $\mathbf{P}_1$ and $\mathbf{P}_2$ are complete, there is some $\mathbf{u}_1 \geq \mathbf{0}$ and $\mathbf{u}_2 \geq \mathbf{0}$ with $A_1^T\mathbf{u}_1 = \mathbf{0}$, $L_1^T\mathbf{u}_1 = \mathbf{0}$, $\mathbf{a}_1^T\mathbf{u}_1 = 1$, and $A_2^T\mathbf{u}_2 = \mathbf{0}$, $L_2^T\mathbf{u}_2 = \mathbf{0}$, $\mathbf{a}_2^T\mathbf{u}_2 = 1$. Let $\mathbf{u} = \binom{\mathbf{u}_1}{\mathbf{u}_2}$, then $A^T\mathbf{u} = \mathbf{0}$, $L^T\mathbf{u} = \mathbf{0}$, and $\mathbf{a}^T\mathbf{u} = 1$. Hence, $\mathbf{P}(A, L, \mathbf{a})$ is complete.

(ii) Instead of showing that every supporting half-space is in $\mathbf{C}(A, L, \mathbf{a})$, we show the more comprehensive property that every including half-space is in $\mathbf{C}(A, L, \mathbf{a})$. Let $\mathbf{H}(\mathbf{n}, c)$ be a half-space that includes $\mathrm{clconv}(\mathbf{P}_1 \cup \mathbf{P}_2)$. Then $\mathbf{P}_1 \subseteq \mathbf{H}(\mathbf{n}, c)$ and $\mathbf{P}_2 \subseteq \mathbf{H}(\mathbf{n}, c)$. Firstly, we discuss the case that neither $\mathbf{P}_1$ nor $\mathbf{P}_2$ are empty. Since $\mathbf{P}_1$ and $\mathbf{P}_2$ are complete, there is some $\mathbf{u}_1 \geq \mathbf{0}$, $L_1^T\mathbf{u}_1 = \mathbf{0}$ and $\mathbf{u}_2 \geq \mathbf{0}$, $L_2^T\mathbf{u}_2 = \mathbf{0}$ such that $\mathbf{n} = A_1^T\mathbf{u}_1 = A_2^T\mathbf{u}_2$ and $c = \mathbf{a}_1^T\mathbf{u}_1 = \mathbf{a}_2^T\mathbf{u}_2$. Thus, for $\mathbf{u} = \binom{\mathbf{u}_1}{\mathbf{u}_2}$ we have $\mathbf{u} \geq \mathbf{0}$, $L^T\mathbf{u} = \mathbf{0}$, $A^T\mathbf{u} = \mathbf{n}$ and $\mathbf{a}^T\mathbf{u} = c$, which is equivalent to $\mathbf{H}(\mathbf{n}, c) \in \mathbf{C}(A, L, \mathbf{a})$. Secondly, we have to discuss the case that at least one of the polyhedra $\mathbf{P}_1$ or $\mathbf{P}_2$ is empty. Nevertheless, both sops are complete. For those $\mathbf{n}$ that are in $\mathbf{N}(A_1, L_1) \cap \mathbf{N}(A_2, L_2)$ there is some $\mathbf{u}_1 \geq \mathbf{0}$, $L_1^T\mathbf{u}_1 = \mathbf{0}$ and some $\mathbf{u}_2 \geq \mathbf{0}$, $L_2^T\mathbf{u}_2 = \mathbf{0}$ with $\mathbf{n} = A_1^T\mathbf{u}_1 = A_2^T\mathbf{u}_2$ and $c = \mathbf{a}_1^T\mathbf{u}_1 = \mathbf{a}_2^T\mathbf{u}_2$. Thus, for $\mathbf{u} = \binom{\mathbf{u}_1}{\mathbf{u}_2}$ we have $\mathbf{u} \geq \mathbf{0}$, $L^T\mathbf{u} = \mathbf{0}$, $A^T\mathbf{u} = \mathbf{n}$ and $\mathbf{a}^T\mathbf{u} = c$, which is equivalent to $\mathbf{H}(\mathbf{n}, c) \in \mathbf{C}(A, L, \mathbf{a})$. In all other cases, i. e., $\mathbf{n} \notin \mathbf{N}(A_1, L_1)$ or $\mathbf{n} \notin \mathbf{N}(A_2, L_2)$, the system

$$\begin{aligned} A_1^T\mathbf{u}_1 &= \mathbf{n}, & A_2^T\mathbf{u}_2 &= \mathbf{n} \\ L_1^T\mathbf{u}_1 &= \mathbf{0}, & L_2^T\mathbf{u}_2 &= \mathbf{0} \\ \mathbf{u}_1 &\geq \mathbf{0}, & \mathbf{u}_2 &\geq \mathbf{0} \end{aligned}$$

is infeasible. Hence, the dual linear program of the support function $h_{\mathbf{P}(A,L,\mathbf{a})}(\mathbf{n})$,

$$\text{minimize } \mathbf{a}_1^T\mathbf{u}_1 \text{ subject to } \begin{aligned} A_1^T\mathbf{u}_1 &= \mathbf{n} \\ A_1^T\mathbf{u}_1 - A_2^T\mathbf{u}_2 &= \mathbf{0} \\ L_1^T\mathbf{u}_1 &= \mathbf{0} \\ L_2^T\mathbf{u}_2 &= \mathbf{0} \\ \mathbf{a}_1^T\mathbf{u}_1 - \mathbf{a}_2^T\mathbf{u}_2 &= 0 \\ \mathbf{u}_1 \geq \mathbf{0}, \mathbf{u}_2 &\geq \mathbf{0}, \end{aligned}$$

is infeasible. In turn, the primal linear program is unbounded, $h_{\mathbf{P}(A,L,\mathbf{a})}(\mathbf{n}) = \infty$.

(iii) The convex hull is empty if and only if $\mathbf{P}_1$ and $\mathbf{P}_2$ are empty, i. e., there is some $\mathbf{u}_1 \geq \mathbf{0}$ and $\mathbf{u}_2 \geq \mathbf{0}$ with $A_1^T\mathbf{u}_1 = \mathbf{0}$, $L_1^T\mathbf{u}_1 = \mathbf{0}$, $\mathbf{a}_1^T\mathbf{u}_1 = -1$, and $A_2^T\mathbf{u}_2 = \mathbf{0}$, $L_2^T\mathbf{u}_2 = \mathbf{0}$, $\mathbf{a}_2^T\mathbf{u}_2 = -1$. Let $\mathbf{u} = \binom{\mathbf{u}_1}{\mathbf{u}_2}$, then $A^T\mathbf{u} = \mathbf{0}$, $L^T\mathbf{u} = \mathbf{0}$, and $\mathbf{a}^T\mathbf{u} = -1$. □

Since the previous proposition is not very reader friendly, we explicitly state the main result for the closed convex hull of non-empty polyhedra as the following corollary.

**Corollary 3.41 (Closed Convex Hull)**
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 complete non-empty sops. Further, let

$$A = \begin{pmatrix} A_1 \\ \mathrm{O} \end{pmatrix}, \quad L = \begin{pmatrix} A_1 & L_1 & \mathrm{O} & \mathbf{a}_1 \\ -A_2 & \mathrm{O} & L_2 & -\mathbf{a}_2 \end{pmatrix}, \quad \text{and } \mathbf{a} = \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{0} \end{pmatrix}.$$

Then $\mathbf{P}(A, L, \mathbf{a}) = \operatorname{clconv}(\mathbf{P}_1 \cup \mathbf{P}_2)$. ■

**Proposition 3.42 (Convex Hull with Origin)**
Let $\mathbf{P}_1 = \mathbf{P}(A_1, L_1, \mathbf{a}_1)$ be a complete sop and $\lhd(\mathbf{P}_1) = \operatorname{clconv}(\{\mathbf{0}\} \cup \mathbf{P}_1)$. Then the following equation holds:

$$\lhd(\mathbf{P}_1) = \mathbf{P}\left(\begin{pmatrix} A_1 \\ \mathbf{0}^T \end{pmatrix}, \begin{pmatrix} L_1 & -\mathbf{a}_1 \\ \mathbf{0}^T & 1 \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}\right). \qquad ■$$

*Proof.* The proof is based on the equality criteria in Proposition 3.36. Let

$$A = \begin{pmatrix} A_1 \\ \mathbf{0}^T \end{pmatrix}, \quad L = \begin{pmatrix} L_1 & -\mathbf{a}_1 \\ \mathbf{0}^T & 1 \end{pmatrix}, \quad \text{and } \mathbf{a} = \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}.$$

(i) We have to show that for all $\mathbf{H}(\mathbf{n}, c) \in \mathbf{C}(A, L, \mathbf{a})$ the relation $\lhd(\mathbf{P}_1) \subseteq \mathbf{H}(\mathbf{n}, c)$ holds. Let $\mathbf{H}(\mathbf{n}, c) \in \mathbf{C}(A, L, \mathbf{a})$. Then there is some $\mathbf{u} \geq \mathbf{0}$ with $\mathbf{0} = L^T\mathbf{u}$, $\mathbf{n} = A^T\mathbf{u}$, and $c = \mathbf{a}^T\mathbf{u}$. After partitioning $\mathbf{u} = \binom{\mathbf{u}_1}{\lambda}$ with $\mathbf{u}_1 \geq \mathbf{0}$ and $\lambda \geq 0$, we obtain the following linear inequalities: $\mathbf{0} = L_1^T\mathbf{u}_1$, $0 = -\mathbf{a}_1^T\mathbf{u}_1 + \lambda$, $\mathbf{n} = A_1^T\mathbf{u}_1$, and $c = \lambda$. It follows $\mathbf{a}_1^T\mathbf{u}_1 = c$ and $\mathbf{H}(\mathbf{n}, c) \in \mathbf{C}(A_1, L_1, \mathbf{a}_1)$. Hence, $\mathbf{P}_1 \subseteq \mathbf{H}(\mathbf{n}, c)$. Additionally, since $c = \lambda$ and $\lambda \geq 0$, we have $\mathbf{n}^T\mathbf{0} \leq c$, i. e., the half-space $\mathbf{H}(\mathbf{n}, c)$ also includes $\{\mathbf{0}\}$, which finally shows $\lhd(\mathbf{P}_1) \subseteq \mathbf{H}(\mathbf{n}, c)$.

(ii) Instead of showing that every supporting half-space of $\lhd(\mathbf{P}_1)$ is in $\mathbf{C}(A, L, \mathbf{a})$, we show the more comprehensive property that every including half-space of $\lhd(\mathbf{P}_1)$ is in $\mathbf{C}(A, L, \mathbf{a})$. Let $\mathbf{H}(\mathbf{n}, c)$ be an including half-space of $\lhd(\mathbf{P}_1)$. Then $\mathbf{H}(\mathbf{n}, c)$ is an including half-space of $\mathbf{P}_1$ which also contains $\{\mathbf{0}\}$. Thus, there exists some $\mathbf{u}_1 \geq \mathbf{0}$ with $\mathbf{n} = A_1^T\mathbf{u}_1$, $\mathbf{0} = L_1^T\mathbf{u}_1$, $c = \mathbf{a}_1^T\mathbf{u}_1$, and, since $\mathbf{H}(\mathbf{n}, c)$ also contains $\mathbf{0}$, $c \geq 0$. Setting $\mathbf{u} = \left(\begin{smallmatrix}\mathbf{u}_1\\ c\end{smallmatrix}\right)$ yields $\mathbf{n} = A^T\mathbf{u}$, $\mathbf{0} = L^T\mathbf{u}$, and $c = \mathbf{a}^T\mathbf{u}$, i. e., $\mathbf{H}(\mathbf{n}, c) \in \mathbf{C}(A, L, \mathbf{a})$.

(iii) The set $\lhd(\mathbf{P}_1)$ is not empty since it contains at least $\mathbf{0}$. □

**Proposition 3.43 (Conical Hull)**
Let $\mathbf{P}_1 = \mathbf{P}(A_1, L_1, \mathbf{a}_1)$ be a complete sop. Then the conical hull of $\mathbf{P}_1$, cone($\mathbf{P}_1$), can be represented as a sop according to the following identity:

$$\text{cone}(\mathbf{P}_1) = \mathbf{P}\left(A_1, \begin{pmatrix}L_1 & -\mathbf{a}_1\end{pmatrix}, \mathbf{0}\right). \qquad \blacksquare$$

*Proof.* The proof is based on the equality criteria in Proposition 3.36. Let

$$A = A_1, \quad L = \begin{pmatrix}L_1 & -\mathbf{a}_1\end{pmatrix}, \quad \text{and } \mathbf{a} = \mathbf{0}.$$

All half-spaces in the half-space cone $\mathbf{C}(A, L, \mathbf{a})$ have the form $\mathbf{H}(\mathbf{n}, 0)$: Let $\mathbf{H}(\mathbf{n}, c) \in \mathbf{C}(A, L, \mathbf{a})$. Then there is some $\mathbf{u} \geq \mathbf{0}$ with $\mathbf{0} = L^T\mathbf{u}$, $\mathbf{n} = A^T\mathbf{u}$, and $c = \mathbf{a}^T\mathbf{u}$. Since $\mathbf{a} = \mathbf{0}$, we have $\mathbf{a}^T\mathbf{u} = c = 0$ for any $\mathbf{u}$.

(i) We have to show that for all $\mathbf{H}(\mathbf{n}, 0) \in \mathbf{C}(A, L, \mathbf{0}) = \mathbf{C}(A, L, \mathbf{a})$ the relation cone($\mathbf{P}_1$) $\subseteq \mathbf{H}(\mathbf{n}, 0)$ holds. Let $\mathbf{H}(\mathbf{n}, 0) \in \mathbf{C}(A, L, \mathbf{0})$. Then there is some $\mathbf{u} \geq \mathbf{0}$ with $\mathbf{0} = L^T\mathbf{u}$ and $\mathbf{n} = A^T\mathbf{u}$. Substituting $A_1$ for $A$ and $\begin{pmatrix}L_1 & -\mathbf{a}_1\end{pmatrix}$ for $L$ yields the following linear inequalities: $\mathbf{0} = L_1^T\mathbf{u}$, $-\mathbf{a}_1^T\mathbf{u} = 0$, and $\mathbf{n} = A_1^T\mathbf{u}$. It follows $\mathbf{H}(\mathbf{n}, 0) \in \mathbf{C}(A_1, L_1, \mathbf{a}_1)$. Hence, for all $\mathbf{x} \in \mathbf{P}_1$ we have $\mathbf{n}^T\mathbf{x} \leq 0$. Furthermore, for any $\mathbf{x} \in \mathbf{P}_1$ and any $\mu \geq 0$ we have $\mathbf{n}^T\mu\mathbf{x} \leq 0$. Hence, cone($\mathbf{P}_1$) $\subseteq \mathbf{H}(\mathbf{n}, 0)$.

(ii) Let $\mathbf{H}(\mathbf{n}, c)$ be a supporting half-space of cone($\mathbf{P}_1$). That is, there is some $\mathbf{x} \in$ cone($\mathbf{P}_1$) such that $\mathbf{n}^T\mathbf{x} = c$ holds. Additionally, for any $\mu \geq 0$ the inequality $\mathbf{n}^T\mu\mathbf{x} \leq c$ has to hold. An easy consideration shows that these constraints force $c = 0$. Hence, there exists some $\mathbf{u} \geq 0$ with $A_1^T\mathbf{u} = \mathbf{n}$, $L_1^T\mathbf{u} = \mathbf{0}$, and $\mathbf{a}_1^T\mathbf{u} = 0$. It follows $L^T\mathbf{u} = \mathbf{0}$ and, hence, $\mathbf{H}(\mathbf{n}, 0) \in \mathbf{C}(A, L, \mathbf{0})$.

(iii) The set cone($\mathbf{P}_1$) is not empty since it contains at least $\mathbf{0}$. □

## 3.11. Simple Flow Operations

The following two propositions are useful for the pre- and postimage computation of linear systems with constant derivatives. Although the naming of the proposition anticipates their later use, they can be understood as purely geometrical operations without deeper knowledge of the pre- and postimage computation of linear systems.

**Proposition 3.44 (Postimage under Constant Derivatives)**
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. Additionally, let $\mathbf{P}_2$ be a polytope, i. e., bounded in every direction. Further, let $[t_0, t_1]$ be an interval with $t_0 \geq 0$, $t_1 \in \mathbb{K} \cup \{\infty\}$ and $t_0 \leq t_1$. Then the sop representing $\mathbf{P}_1 + \bigcup_{t\in[t_0,t_1]} t\mathbf{P}_2$ is given by the following equation

$$\mathbf{P}_1 + \bigcup_{t\in[t_0,t_1]} t\mathbf{P}_2 = \mathbf{P}\left(\begin{pmatrix} A_1 \\ \mathrm{O} \\ \mathbf{0}^T \\ \mathbf{0}^T \end{pmatrix}, \begin{pmatrix} -A_1 & L_1 & \mathrm{O} & \mathbf{0} \\ A_2 & \mathrm{O} & L_2 & -\mathbf{a}_2 \\ \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & -1 \\ \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & 1 \end{pmatrix}, \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{0} \\ -t_0 \\ t_1 \end{pmatrix}\right), \tag{3.11}$$

where we may omit the second last row if $t_0 = 0$ and $\mathbf{P}_2$ is complete, and omit the last row if $t_1 = \infty$. We call $\mathbf{P}_1 + \bigcup_{t\in[t_0,t_1]} t\mathbf{P}_2$ the *postimage of* $\mathbf{P}_1$ *under the constant derivatives* $\mathbf{P}_2$. ■

*Proof.* For any $\mathbf{x}$ we have $\mathbf{x} \in \mathbf{P}_1 + \bigcup_{t\in[t_0,t_1]} t\mathbf{P}_2$ if and only if there exists $\mathbf{x}_1 \in \mathbf{P}_1$, $t \in [t_0, t_1]$, and $\mathbf{x}_2 \in t\mathbf{P}_2$ with $\mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2$, or, equivalently, if and only if there exists some $t \in [t_0, t_1]$ and $\mathbf{x}_2 \in t\mathbf{P}_2$ such that $\mathbf{x} - \mathbf{x}_2 \in \mathbf{P}_1$. Since $\mathbf{P}_2$ is a polytope, we use Proposition 3.21 and obtain the characterization $\mathbf{x} \in \mathbf{P}_1 + \bigcup_{t\in[t_0,t_1]} t\mathbf{P}_2$ if and only if the following system of linear inequalities has a solution for $(\mathbf{x}_2, \mathbf{z}_1, \mathbf{z}_2, t)$:

$$\begin{array}{cccccl} A_1\mathbf{x} & -A_1\mathbf{x}_2 & +L_1\mathbf{z}_1 & & & \leq \mathbf{a}_1 \\ & A_2\mathbf{x}_2 & & +L_2\mathbf{z}_2 & -t\mathbf{a}_2 & \leq \mathbf{0} \\ & & & & -t & \leq -t_0 \\ & & & & t & \leq t_1. \end{array}$$

The sop in (3.11) exactly encodes this system. Clearly, for $t_1 = \infty$ the last row may be omitted. It remains to show that the second last row may be omitted if $t_0 = 0$ and $\mathbf{P}_2$ is complete. Let $t_0 = 0$ and $\mathbf{P}_2$ be complete. Then there is some $\mathbf{u} \geq \mathbf{0}$ with $A_2^T\mathbf{u} = \mathbf{0}$, $L_2^T\mathbf{u} = \mathbf{0}$, $\mathbf{a}_2^T\mathbf{u} = 1$. Hence, $-t\mathbf{a}_2^T\mathbf{u}$ is positive for any $t < 0$. That is, $A_2\mathbf{x}_2 + L_2\mathbf{z}_2 - t\mathbf{a}_2 \leq \mathbf{0}$ has no solution for $t < 0$. Hence, the second last row may be omitted. □

**Proposition 3.45 (Preimage under Constant Derivatives)**
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. Additionally, let $\mathbf{P}_2$ be a polytope, i. e., bounded in every direction. Further, let $[t_0, t_1]$ be an interval with $t_0 \geq 0$, $t_1 \in \mathbb{K} \cup \{\infty\}$ and $t_0 \leq t_1$. Then the preimage of $\mathbf{P}_1$ under the constant derivatives $\mathbf{P}_2$ is the set

$$\mathbf{P}_1 - \bigcup_{t\in[t_0,t_1]} t\mathbf{P}_2 = \mathbf{P}_1 + \bigcup_{t\in[t_0,t_1]} -t\mathbf{P}_2. \quad ■$$

*Proof.* The postimage of a set $\mathbf{P}_1$ under the constant derivatives $\mathbf{P}_2$ can be written as the set

$$\{\mathbf{x} \mid \exists \mathbf{y} \in \mathbf{P}_1 \exists \mathbf{v} \in \mathbf{P}_2 \exists t \in [t_0, t_1]\colon \mathbf{x} = \mathbf{y} + t\mathbf{v}\}.$$

The preimage of a set $\mathbf{P}_1$ under the constant derivatives $\mathbf{P}_2$ is then the set

$$\begin{aligned}
&\{\mathbf{x} \mid \exists \mathbf{y} \in \mathbf{P}_1 \exists \mathbf{v} \in \mathbf{P}_2 \exists t \in [t_0, t_1]\colon \mathbf{y} = \mathbf{x} + t\mathbf{v}\} \\
\iff &\{\mathbf{x} \mid \exists \mathbf{y} \in \mathbf{P}_1 \exists \mathbf{v} \in \mathbf{P}_2 \exists t \in [t_0, t_1]\colon \mathbf{x} = \mathbf{y} - t\mathbf{v}\} = \mathbf{P}_1 - \bigcup_{t \in [t_0, t_1]} t\mathbf{P}_2 \\
\iff &\{\mathbf{x} \mid \exists \mathbf{y} \in \mathbf{P}_1 \exists \mathbf{v} \in -\mathbf{P}_2 \exists t \in [t_0, t_1]\colon \mathbf{x} = \mathbf{y} + t\mathbf{v}\} = \mathbf{P}_1 + \bigcup_{t \in [t_0, t_1]} -t\mathbf{P}_2. \qquad \square
\end{aligned}$$

## 3.12. Linear Programming and Sops

Sops profit from the underlying $\mathcal{H}$-representation, e. g., we may solve linear programs to test for emptiness, to find separating half-spaces, or to find relative interior points. Additionally, we may switch from the primal system of linear inequalities to its dual. The combination of projections and linear programming is not new, e. g. in Benoy et al. (2005) the authors propose a technique which computes the convex hull from a projection using linear programming. In this section we shall make use of these techniques. Eventually, we present two methods which take a sop $\mathbf{P}$ and a point $\mathbf{r}$ which is not in the interior of $\mathbf{P}$ and return a supporting half-space $\mathbf{H}$ of $\mathbf{P}$. The half-space $\mathbf{H}$ either contains $\mathbf{r}$, then $\mathbf{r}$ is a boundary point of $\mathbf{P}$, or separates $\mathbf{r}$ from $\mathbf{P}$. These methods are then extended to an interpolation method that allows us to improve existing over-approximations. The needed geometrical concepts are shortly introduced in the following. A comprehensive introduction can be found in Ziegler (1995).

**Proposition 3.46 (Strictly Separating Hyperplane/Half-Space)**
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 sops in $\mathbb{K}^d$. The following statements are mutually exclusive.

1. There exists some common point $\mathbf{x}$ with $\mathbf{x} \in \mathbf{P}_1$ and $\mathbf{x} \in \mathbf{P}_2$.
2. There exists $\mathbf{u} \geq 0, \mathbf{v} \geq 0$ with $A_1^T\mathbf{u} + A_2^T\mathbf{v} = \mathbf{0}$, $L_1^T\mathbf{u} = L_2^T\mathbf{v} = \mathbf{0}$, and $\mathbf{a}_1^T\mathbf{u} + \mathbf{a}_2^T\mathbf{v} = -1$. Let $\mathbf{H}$ be the half-space $\mathbf{H}(A_1^T\mathbf{u}, \mathbf{a}_1^T\mathbf{u}) \subseteq \mathbb{K}^d$. Then $\mathbf{P}_1 \subseteq \mathbf{H}_1$ and $\mathbf{H}_1 \cap \mathbf{P}_2 = \emptyset$, i. e., $\mathbf{H}$ strictly separates $\mathbf{P}_1$ and $\mathbf{P}_2$. $\blacksquare$

*Proof.* The intersection $\mathbf{P}_1 \cap \mathbf{P}_2$ is given by the sop $\mathbf{P}\left(\begin{pmatrix} A_1 \\ A_2 \end{pmatrix}, \begin{pmatrix} L_1 & O \\ O & L_2 \end{pmatrix}, \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{pmatrix}\right)$, and it is not empty if and only if

$$\begin{aligned} A_1\mathbf{x} + L_1\mathbf{z}_1 &\leq \mathbf{a}_1, \\ A_2\mathbf{x} + L_2\mathbf{z}_2 &\leq \mathbf{a}_2 \end{aligned} \tag{3.12}$$

has a solution $(\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2)$. According to Farkas' Lemma (Theorem 2.3) either the system (3.12) has a solution, or there exist vectors $\mathbf{u} \geq 0$, $\mathbf{v} \geq \mathbf{0}$ with

$$\begin{gathered} A_1^T\mathbf{u} + A_2^T\mathbf{v} = \mathbf{0}, \\ L_1^T\mathbf{u} = \mathbf{0},\ L_2^T\mathbf{v} = \mathbf{0}, \\ \mathbf{a}_1^T\mathbf{u} + \mathbf{a}_2^T\mathbf{v} < 0. \end{gathered} \tag{3.13}$$

Assume, such vectors $\mathbf{u}$ and $\mathbf{v}$ exist with $\mathbf{a}_1^T\mathbf{u} + \mathbf{a}_2^T\mathbf{v} = \epsilon < 0$. Then $\mathbf{u}' = -\frac{1}{\epsilon}\mathbf{u}$ and $\mathbf{v}' = -\frac{1}{\epsilon}\mathbf{v}$ is a solution of

$$\begin{aligned} A_1^T\mathbf{u} + A_2^T\mathbf{v} &= \mathbf{0}, \\ L_1^T\mathbf{u} = \mathbf{0},\, L_2^T\mathbf{v} &= \mathbf{0}, \\ \mathbf{a}_1^T\mathbf{u} + \mathbf{a}_2^T\mathbf{v} &= -1. \end{aligned} \tag{3.14}$$

Clearly, any solution of (3.14) is also a solution of (3.13). Hence, the systems (3.12) and (3.14) are mutually exclusive. For the case $\mathbf{P}_1 \cap \mathbf{P}_2 = \emptyset$ it remains to show that $\mathbf{H} = \mathbf{H}(A_1^T\mathbf{u}, \mathbf{a}_1^T\mathbf{u})$ is a separating half-space. Firstly, we note that $\mathbf{H}(A_1^T\mathbf{u}, \mathbf{a}_1^T\mathbf{u}) \in \mathbf{C}(A_1, L_1, \mathbf{a}_1)$ and $\mathbf{H}(A_2^T\mathbf{v}, \mathbf{a}_2^T\mathbf{v}) \in \mathbf{C}(A_2, L_2, \mathbf{a}_2)$. Hence $\mathbf{P}_1 \subseteq \mathbf{H}(A_1^T\mathbf{u}, \mathbf{a}_1^T\mathbf{u})$ and $\mathbf{P}_2 \subseteq \mathbf{H}(A_2^T\mathbf{v}, \mathbf{a}_2^T\mathbf{v})$. Suppose, there is some $\mathbf{x}$ such that $\mathbf{x} \in \mathbf{P}_2$ and $\mathbf{x} \in \mathbf{H}$. Then $\mathbf{v}^T A_2\mathbf{x} \leq \mathbf{v}^T\mathbf{a}_2$ has to hold. Transforming the inequality yields $-\mathbf{v}^T A_2\mathbf{x} \geq -\mathbf{v}^T\mathbf{a}_2 > -1 - \mathbf{v}^T\mathbf{a}_2$. Using (3.14) yields $\mathbf{u}^T A_1 > \mathbf{u}^T\mathbf{a}_1$, which is a contradiction to $\mathbf{x} \in \mathbf{H}$. □

Let $\mathbf{P} = \mathbf{P}(A, \mathbf{a})$ be an $\mathcal{H}$-polyhedron. The points of $\mathbf{P}$ are those vectors $\mathbf{x}$ that satisfy the system $A\mathbf{x} \leq \mathbf{a}$. 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 $\operatorname{aff}(\mathbf{P})$. Hence, we call a point $\mathbf{x}$ of $\mathbf{P}$ a *relative interior point* relatively to $\operatorname{aff}(\mathbf{P})$ if there exists a ball $\mathbf{B}_\epsilon$ with $\epsilon > 0$ such that $(\mathbf{x} + \mathbf{B}_\epsilon) \cap \operatorname{aff}(\mathbf{P}) \subseteq \mathbf{P}$. A *facet-defining* half-space $\mathbf{H}$ of $\mathbf{P}$ is a half-space $\mathbf{H} = \{\mathbf{x} \,|\, \mathbf{n}^T\mathbf{x} \leq b\}$, $\mathbf{P} \subseteq \mathbf{H}$, such that $\mathbf{P} \cap \{\mathbf{x} \,|\, \mathbf{n}^T\mathbf{x} = b\}$ has a relative interior point relatively to $\operatorname{aff}(\mathbf{P}) \cap \{\mathbf{x} \,|\, \mathbf{n}^T\mathbf{x} = b\}$.

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 given as 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 following proposition is adapted for the usage with sops and can be found in Fukuda (2011).

**Proposition 3.47 (Interior Point)**
Let $\mathbf{P} = \mathbf{P}(A, L, \mathbf{a})$ be a sop and $I$ a (possibly empty) subset of the row indices of the implicit equalities of the system $A\mathbf{x} + L\mathbf{z} \leq \mathbf{a}$. Further, let $S$ be the set of all row indices that are not in $I$. Then the linear program

$$\text{maximize } \lambda \text{ subject to } \quad \begin{aligned} A_I\mathbf{x} + L_I\mathbf{z} &= \mathbf{a}_I, \\ A_S\mathbf{x} + L_S\mathbf{z} + \mathbf{1}\lambda &\leq \mathbf{a}_S, \\ -\lambda \leq 0,\, \lambda &\leq 1 \end{aligned} \tag{3.15}$$

is either infeasible and $\mathbf{P}$ is empty, or an optimal solution exists. Let $(\mathbf{x}_0, \mathbf{z}_0, \lambda_0)$ be an optimal solution of the linear program and $(\mathbf{u}_0, \mathbf{v}_0, \mu_0, \nu_0)$ be the optimal solution of the dual linear program

$$\text{minimize } \mathbf{a}_I^T\mathbf{u} + \mathbf{a}_S^T\mathbf{v} + \nu \text{ subject to} \qquad \begin{aligned} A_I^T\mathbf{u} + A_S^T\mathbf{v} &= \mathbf{0}, \\ L_I^T\mathbf{u} + L_S^T\mathbf{v} &= \mathbf{0}, \\ \mathbf{1}^T\mathbf{v} - \mu + \nu &= 1, \\ \mathbf{v} \geq 0,\ \mu \geq 0, \nu &\geq 0. \end{aligned} \tag{3.16}$$

If $\lambda > 0$, then $\mathbf{x}_0$ is a relative interior point of $\mathbf{P}$. In the case $\lambda = 0$, some of the inequalities in $A_S\mathbf{x} + L_S\mathbf{z} \leq \mathbf{a}_s$ are implicit inequalities. Removing their row indices from $S$ and adding them to $I$ provides an equivalent system where additional implicit inequalities are indexed by $I$. Repeated solving of the linear programs eventually provides (i) the complete set $I$ of the row indices of all implicit inequalities and (ii) a relative interior point. ■

*Proof.* Assume $\mathbf{P}$ is not empty, i. e., there is some $\mathbf{x}_0$ and $\mathbf{z}_0$ such that $A_I\mathbf{x}_0 + L_I\mathbf{z}_0 = \mathbf{a}_I$ and $A_S\mathbf{x}_0 + L_S\mathbf{z}_0 \leq \mathbf{a}_S$ holds. Then $(\mathbf{x}_0, \mathbf{z}_0, 0)$ is clearly a feasible solution of the primal linear program. On the other hand, let $(\mathbf{x}_0, \mathbf{z}_0, \lambda_0)$ be a feasible solution of the primal linear program. Then $A_I\mathbf{x}_0 + L_I\mathbf{z}_0 = \mathbf{a}_I$ and $A_S\mathbf{x}_0 + L_S\mathbf{z}_0 \leq \mathbf{a}_S - \mathbf{1}\lambda_0 \leq \mathbf{a}_S$ hold. Hence, $\mathbf{x}_0$ in $\mathbf{P}$ and $\mathbf{P}$ is not empty.

For the remainder of the proof we assume that the linear program is feasible. The primal linear program is bounded by the constraint $0 \leq \lambda \leq 1$. Hence, by Strong Duality (Theorem 2.8), both, the primal and the dual linear program, have an optimal solution $(\mathbf{x}_0, \mathbf{z}_0, \lambda_0)$ and $(\mathbf{u}_0, \mathbf{v}_0, \mu_0, \nu_0)$ with $\mathbf{a}_I^T\mathbf{u}_0 + \mathbf{a}_S^T\mathbf{v}_0 + \nu_0 = \lambda_0$.

In the case $\lambda_0 > 0$, the point $\mathbf{x}_0$ is obviously a relative interior point.

In the case $\lambda_0 = 0$, it is important to note that any $\binom{\mathbf{x}}{\mathbf{z}}$ with $A_I\mathbf{x} + L_I\mathbf{z} = \mathbf{a}_I$ and $A_S\mathbf{x} + L_S\mathbf{z} \leq \mathbf{a}_S$ is an optimal solution of the primal linear program. Hence, if we show that for an optimal solution of the primal linear program some additional inequalities are tight, they must be tight for any $\mathbf{x} \in \mathbf{P}$. Firstly, we show $\mathbf{v}_0 \neq \mathbf{0}$. To this end, suppose $\mathbf{v}_0 = \mathbf{0}$. Since $(\mathbf{x}_0, \mathbf{z}_0, 0)$ is a solution of the primal linear program, the equality $\mathbf{a}_I = A_I\mathbf{x}_0 + L_I\mathbf{z}_0$, and, hence,

$$\mathbf{u}_0^T\mathbf{a}_I = \mathbf{u}_0^T A_I\mathbf{x}_0 + \mathbf{u}_0^T L_I\mathbf{z}_0 \tag{3.17}$$

holds. Substituting $\mathbf{v}_0 = \mathbf{0}$ in the dual linear program yields

$$A_I^T\mathbf{u}_0 = \mathbf{0}, \tag{3.18}$$

$$L_I^T\mathbf{u}_0 = \mathbf{0}, \tag{3.19}$$

$$\nu_0 = 1 + \mu_0,$$

and for the objective function

$$\mathbf{a}_I^T\mathbf{u}_0 + 1 + \mu_0 = 0. \tag{3.20}$$

Substituting (3.18) and (3.19) in (3.17) yields $\mathbf{u}_0^T\mathbf{a}_I = 0$. Together with (3.20) this yields a contradiction. Hence, at least one of the coefficients of $\mathbf{v}_0$ is positive. Now, Strong Duality yields

$$\begin{pmatrix}\mathbf{u}_0\\ \mathbf{v}_0\\ \mu_0\\ \nu_0\end{pmatrix}^T \begin{pmatrix}\mathbf{a}_I\\ \mathbf{a}_S\\ 0\\ 1\end{pmatrix} = \begin{pmatrix}\mathbf{0}\\ \mathbf{0}\\ 1\\ 0\end{pmatrix}^T \begin{pmatrix}\mathbf{x}_0\\ \mathbf{z}_0\\ 0\end{pmatrix} = \begin{pmatrix}\mathbf{u}_0\\ \mathbf{v}_0\\ \mu_0\\ \nu_0\end{pmatrix}^T \begin{pmatrix}A_I & L_I & \mathbf{0}\\ A_S & L_S & \mathbf{1}\\ \mathbf{0}^T & \mathbf{0}^T & -1\\ \mathbf{0}^T & \mathbf{0}^T & 1\end{pmatrix} \begin{pmatrix}\mathbf{x}_0\\ \mathbf{z}_0\\ 0\end{pmatrix},$$

$$\mathbf{u}_0^T(\mathbf{a}_I - A_I\mathbf{x}_0 - L_I\mathbf{z}_0) + \mathbf{v}_0^T(\mathbf{a}_S - A_S\mathbf{x}_0 - L_S\mathbf{z}_0) + \nu_0 = 0. \tag{3.21}$$

Since $(\mathbf{x}_0, \mathbf{z}_0, 0)$ and $(\mathbf{u}_0, \mathbf{v}_0, \mu_0, \nu_0)$ are feasible solutions, we have $\mathbf{a}_I - A_I\mathbf{x}_0 - L_I\mathbf{z}_0 = \mathbf{0}$ and $\mathbf{a}_S - A_S\mathbf{x}_0 - L_S\mathbf{z}_0 \geq \mathbf{0}$. Hence, $\mathbf{u}_0^T(\mathbf{a}_I - A_I\mathbf{x}_0 - L_I\mathbf{z}_0) = 0$ and $\mathbf{v}_0^T(\mathbf{a}_S - A_S\mathbf{x}_0 - L_S\mathbf{z}_0) \geq 0$. Using (3.21) we obtain $\mathbf{v}_0^T(\mathbf{a}_S - A_S\mathbf{x}_0 - L_S\mathbf{z}_0) = 0$ and $\nu_0 = 0$. Let $J$ be the set of indices where $\mathbf{v}_0$ is positive. We already know that $J \neq \emptyset$. Since $\mathbf{v}_0^T(\mathbf{a}_S - A_S\mathbf{x}_0 - L_S\mathbf{z}_0) = 0$ has to hold, every inequality $A_j\mathbf{x}_0 + L_j\mathbf{z}_0 \leq \mathbf{a}_j$, $j \in J$, must be an implicit equality. □

We present the first method that allows to find a supporting half-space of a sop $\mathbf{P}$ containing the origin in the direction of the point $\mathbf{r}$.

**Proposition 3.48 (Ray Shooting)**
Let $\mathbf{P} = \mathbf{P}(A, L, \mathbf{a})$ be a non-empty and complete sop that contains the origin $\mathbf{0}$ as relative interior point. Then the following linear program is feasible for any vector $\mathbf{r}$:

$$\text{maximize } \mathbf{r}^T A^T \mathbf{u} \text{ subject to } L^T\mathbf{u} = \mathbf{0},\ \mathbf{a}^T\mathbf{u} = 1,\ \mathbf{u} \geq \mathbf{0}. \tag{3.22}$$

■

Let $\lambda_0 = \mathbf{r}^T A^T \mathbf{u}_0 \in \mathbb{K} \cup \{\pm\infty\}$ be the optimal value of the linear program. Then exactly one of the following statements holds:

(st1) The linear program is unbounded and $\mathbf{r} \notin \operatorname{aff}(\mathbf{P})$.

(st2) The optimal value $\lambda_0$ equals zero, $\mathbf{r} \in \operatorname{aff}(\mathbf{P})$, and $\mathbf{P}$ is unbounded in direction $\mathbf{r}$.

(st3) The optimal value $\lambda_0$ is positive, $\mathbf{r} \in \operatorname{aff}(\mathbf{P})$, and $\mathbf{P}$ is bounded in direction $\mathbf{r}$.

Additionally, in case (st3) also the following statement holds:

(st3') Let $\mathbf{n} = A^T\mathbf{u}_0$ and $\mathbf{H} = \mathbf{H}(\mathbf{n}, 1)$ be a half-space. Then $\frac{1}{\lambda_0}\mathbf{r}$ is a boundary point of $\mathbf{P}$ and $\mathbf{H}$ is a supporting half-space of $\mathbf{P}$ in $\frac{1}{\lambda_0}\mathbf{r}$.

We need the following small lemma to prove the proposition.

**Lemma 3.49**
Let $S_1 = \{s_1, \ldots, s_n\}$ and $S_2 = \{t_1, \ldots, t_n\}$ be two sets of statements. Further, let the statements in each group be mutually exclusive and exhaustive, i. e., in every possible interpretation exactly one of the statements in $S_1$ and $S_2$ is true. Assume, we have shown the implications $s_i \Rightarrow t_i$ for $i = 1, \ldots, n$. Then $S_3 = \{s_1 \wedge t_1, \ldots, s_n \wedge t_n\}$ is again a set of mutually exclusive and exhaustive statements. ■

*Proof. (Proof of the lemma)* Assume, we have already shown the implications $s_i \Rightarrow t_i$ for $i = 1, \dots, n$. Since $S_1$ and $S_2$ are exhaustive, exactly one statement of each set has to hold, i. e., there exists some $j = 1, \dots, n$ and some $k = 1, \dots, n$ such that $s_j$ and $t_k$ hold. Assume, $k \neq j$. The statement $s_j$ implies $t_j$. Hence, $t_j$ and $t_k$ hold, which is a contradiction to the the mutual exclusiveness of $S_2$. Hence, $k = j$, i. e., the set $S_3$ consists of mutually exclusive and exhaustive statements. □

*Proof. (Proof of the proposition)* We split the statements (st1)-(st3) into the following sub-statements:

|  |  |
|---|---|
| (s1) $\lambda_0$ unbounded, | (t1) $\mathbf{r} \notin \operatorname{aff}(\mathbf{P})$, |
| (s2) $\lambda_0 = 0$, | (t2) $\mathbf{r} \in \operatorname{aff}(\mathbf{P})$, $\mathbf{P}$ is unbounded in direction $\mathbf{r}$, |
| (s3) $\lambda_0 > 0$, | (t3) $\mathbf{r} \in \operatorname{aff}(\mathbf{P})$, $\mathbf{P}$ is bounded in direction $\mathbf{r}$. |

We define the following sets of statements:

$$S_1 = \{(s1), (s2), (s3)\} \quad \text{and} \quad S_2 = \{(t1), (t2), (t3)\}\,.$$

Clearly, for any polyhedron $\mathbf{P}$ and any vector $\mathbf{r}$ the statements in $S_2$ are mutually exclusive and exhaustive. We show that $S_1$ is also a set of mutually exclusive and exhaustive statements: The linear program is always feasible since $\mathbf{P}$ is complete, i. e., there exists some $\mathbf{u} \geq 0$ with $A^T\mathbf{u} = \mathbf{0}$, $L^T\mathbf{u} = \mathbf{0}$, and $\mathbf{a}^T\mathbf{u} = 1$. It follows $\mathbf{r}^T A^T \mathbf{u} = \mathbf{r}^T\mathbf{0} \geq 0$. Hence, the optimal value of the linear program cannot be negative, i. e., $S_1$ consists of mutually exclusive and exhaustive statements.

In the following, we will make use of the dual linear program of (3.22).

$$\text{minimize } \mu \text{ subject to } L\mathbf{z} + \mu\mathbf{a} \geq A\mathbf{r}. \tag{3.23}$$

We prove the implication (s1) $\Rightarrow$ (t1) by contraposition. Assume, $\mathbf{r} \in \operatorname{aff}(\mathbf{P})$. Since $\mathbf{0} \in \mathbf{P}$, there exists some $\lambda > 0$ such that $\lambda\mathbf{r} \in \mathbf{P}$, i. e., there exists some $z$ such that $A\lambda\mathbf{r} + L\mathbf{z} \leq \mathbf{a}$. Hence, there also exists some $\mathbf{z}'$ with $A\mathbf{r} - L\mathbf{z}' \leq \frac{1}{\lambda}\mathbf{a}$. The tuple $(\mathbf{z}', \frac{1}{\lambda})$ forms a solution of the dual linear program (3.23), which is hence feasible. The feasibility of the dual linear program implies that the primal linear system (3.22) is not unbounded.

We prove the implication (s2) $\Rightarrow$ (t2) and (s3) $\Rightarrow$ (t3). Firstly, we note that $\mathbf{a} \geq \mathbf{0}$ has to hold since $\mathbf{0} \in \mathbf{P}$. Let either (s2) or (s3) hold, i. e., $\mathbf{r}^T A^T \mathbf{u}_0 \geq 0$. That is, the linear program (3.22) has an optimal solution $\mathbf{u}_0$ with the optimal value $\lambda_0 = \mathbf{r}^T A^T \mathbf{u}_0$. For any pair $(\mathbf{u}_0, \binom{\mathbf{z}_0}{\mu_0})$ of optimal solutions of the linear program (3.22) and of the dual linear program (3.23) Strong Duality (Theorem 2.8) yields $\mu_0 = \mathbf{r}^T A^T \mathbf{u}_0 = \lambda_0$. Transforming the linear inequalities of (3.23) yields

$$\begin{pmatrix} A & L \end{pmatrix} \begin{pmatrix} \mathbf{r} \\ -\mathbf{z}_0 \end{pmatrix} \leq \mu_0 \mathbf{a}. \tag{3.24}$$

Recall that $\mathbf{a} \geq \mathbf{0}$. If the optimal value $\mu_0$ is equal to zero, then (3.24) holds for all multiple $\nu\binom{\mathbf{r}}{-\mathbf{z}_0}$ with $0 \leq \nu$, and, hence, $\{\nu\mathbf{r} \,|\, 0 \leq \nu\} \subseteq \mathbf{P}$, i. e., $\mathbf{P}$ is unbounded in direction $\mathbf{r}$. Hence, (s2) $\Rightarrow$ (t2). Otherwise, the optimal value $\mu_0$ is positive and we may multiply both sides of (3.24) with $\frac{1}{\mu_0}$ and obtain $\frac{1}{\mu_0}\mathbf{r} \in \mathbf{P}$. For any feasible solution of the primal linear program $\mathbf{a}^T\mathbf{u}_0 = 1$ holds. Now, set $\mathbf{n} = A^T\mathbf{u}_0$. Then $\mathbf{H} = \mathbf{H}(\mathbf{n}, 1)$ is in the cone $\mathbf{C}(A, L, \mathbf{a})$ and, hence, an including half-space of $\mathbf{P}$. Furthermore, $\mathbf{n}^T\frac{1}{\lambda_0}\mathbf{r} = \frac{1}{\mathbf{r}^T A^T\mathbf{u}_0}\mathbf{r}^T A^T\mathbf{u}_0 = 1$, hence $\frac{1}{\lambda_0}\mathbf{r} \in \mathbf{H}$. That is, $\mathbf{P}$ is bounded in direction $\mathbf{r}$, $\mathbf{H}$ is a supporting hyperplane of $\mathbf{P}$, and $\frac{1}{\lambda_0}\mathbf{r}$ is a boundary point of $\mathbf{P}$, which shows (s3) $\Rightarrow$ (t3) and (s3) $\Rightarrow$ (st3').

Finally, application of the previous lemma finishes the proof. □

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.

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 $\operatorname{aff}(\mathbf{P} \cap \mathbf{H}_=)$ has the dimension $d-1$. In practice, we have to solve several linear programs, which makes this test costly.

We present a second method that allows to find a supporting half-space of a sop $\mathbf{P}(A, L, \mathbf{a})$. The idea is to change the vector $\mathbf{a}$ by adding a multiple of the vector $\mathbf{1}$ until some given point $\mathbf{r}$ is on the boundary of $\mathbf{P}(A, L, \mathbf{a} - \mathbf{1}\lambda)$.

**Proposition 3.50 (Face Bloating)**
Let $\mathbf{P} = \mathbf{P}(A, L, \mathbf{a})$ be a sop in $\mathbb{K}^d$ and $\mathbf{r}$ be a point in $\mathbb{K}^d$ which is not in the interior of $\mathbf{P}$. Then the linear program

$$\text{maximize } \lambda \text{ subject to } L\mathbf{z} + \mathbf{1}\lambda \le \mathbf{a} - A\mathbf{r} \tag{3.25}$$

is always feasible and has an optimal solution $\lambda_0$ if and only if $\mathbf{P} \ne \mathbb{K}^d$. Let $\lambda_0$ be the optimal value of the primal linear program and $\mathbf{u}_0$ be an optimal solution of the dual linear program

$$\text{minimize } (\mathbf{a} - A\mathbf{r})^T\mathbf{u} \text{ subject to } L^T\mathbf{u} = \mathbf{0},\ \mathbf{1}^T\mathbf{u} = 1,\ \mathbf{u} \ge \mathbf{0}. \tag{3.26}$$

Further, let $\mathbf{n} = A^T\mathbf{u}_0$ and $c = \mathbf{a}^T\mathbf{u}_0$. Then exactly one of the following statements holds:

(1) $\lambda_0 = 0$, $\mathbf{r}$ is a boundary point of $\mathbf{P}$, and $\mathbf{H}(\mathbf{n}, c)$ is a supporting half-space of $\mathbf{P}$ in $\mathbf{r}$.

(2) $\lambda_0 < 0$, $\mathbf{r}$ is not in $\mathbf{P}$, and $\mathbf{H}(\mathbf{n}, h_{\mathbf{P}}(\mathbf{n}))$ is a supporting half-space which separates $\mathbf{r}$ and $\mathbf{P}$. ■

*Proof.* We show that (3.25) is always feasible: Let $\mathbf{z} = \mathbf{0}$ and $\lambda$ be the minimal coefficient of the vector $\mathbf{a} - A\mathbf{r}$. Then $(\mathbf{z}, \lambda)$ is a feasible solution of (3.25). Hence, (3.25) is either unbounded or has an optimal solution. The former case is equivalent to the infeasibility of (3.26), i. e., $L^T\mathbf{u} = \mathbf{0}$, $\mathbf{1}^T\mathbf{u} = 1$, $\mathbf{u} \ge \mathbf{0}$ is infeasible. Equivalently, $L^T\mathbf{u} = \mathbf{0}$, $\mathbf{u} \ge \mathbf{0}$ if and only if $\mathbf{u} = \mathbf{0}$. This is again equivalent to the infeasibility of the dual linear program of the support function $h_{\mathbf{P}}(\mathbf{n})$ for $\mathbf{n} \neq \mathbf{0}$, see (3.3), i. e., $h_{\mathbf{P}}(\mathbf{n}) = \infty$ for any $\mathbf{n} \neq \mathbf{0}$, which finally establishes $\mathbf{P} = \mathbb{K}^d$ if and only if (3.25) is unbounded.

Let $\lambda_0$ be the optimal value of (3.25) and $\mathbf{u}_0$ be the optimal solution of (3.26). It is easy to see that $\mathbf{r}$ is a boundary point of $\mathbf{P}\,(A, L, \mathbf{a} - \mathbf{1}\lambda_0)$. The half-space $\mathbf{H}(A^T\mathbf{u}_0, (\mathbf{a} - \mathbf{1}\lambda_0)^T\mathbf{u}_0)$ is an including half-space of $\mathbf{P}\,(A, L, \mathbf{a} - \mathbf{1}\lambda_0)$. Furthermore, by Strong Duality we have $\mathbf{u}_0^T\mathbf{1}\lambda_0 = \lambda_0 = \mathbf{u}_0^T(\mathbf{a} - A\mathbf{r}) = \mathbf{u}_0^T\mathbf{a} - \mathbf{u}_0^T A\mathbf{r}$ and, hence, $\mathbf{r}$ is a point of $\{\mathbf{x} \,|\, \mathbf{u}_0^T A\mathbf{x} = (\mathbf{a} - \mathbf{1}\lambda_0)^T\mathbf{u}_0\}$, i. e., $\mathbf{r}$ is a common boundary point of $\mathbf{P}\,(A, L, \mathbf{a} - \mathbf{1}\lambda_0)$ and the supporting half-space $\mathbf{H}(A^T\mathbf{u}, (\mathbf{a} - \mathbf{1}\lambda_0)^T\mathbf{u})$. Let $\mathbf{n} = A^T\mathbf{u}_0$ and $c = \mathbf{a}^T\mathbf{u}_0$.

For $\lambda_0 = 0$ the point $\mathbf{r}$ is a common boundary point of $\mathbf{P}$ and its supporting half-space $\mathbf{H}(\mathbf{n}, c)$.

For $\lambda_0 < 0$ the point $\mathbf{r}$ is not in $\mathbf{P}$. The half-space $\mathbf{H}(\mathbf{n}, c)$ is an implied half-space. Hence, $h_{\mathbf{P}}(\mathbf{n}) \leq c$ and $\mathbf{H}(\mathbf{n}, h_{\mathbf{P}}(\mathbf{n}))$ is a supporting half-space. Additionally, $\mathbf{n}^T\mathbf{r} = c - \lambda_0 > c \geq h_{\mathbf{P}}(\mathbf{n})$ holds, i. e., $\mathbf{H}(\mathbf{n}, h_{\mathbf{P}}(\mathbf{n}))$ separates $\mathbf{P}$ and $\mathbf{r}$. □

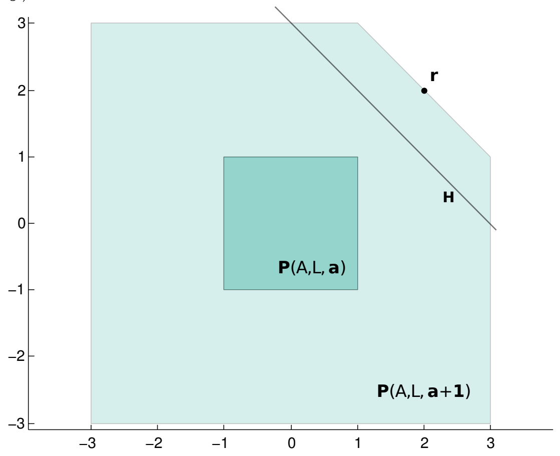
In case (2) we have to solve an additional linear program to obtain the supporting half-space. The following example shows that this additional linear program is indeed needed.

**Example 3.51**
Let $\mathbf{P} = \mathbf{P}\,(A, L, \mathbf{a})$ be a sop in $\mathbb{K}^2$ and $\mathbf{r}$ be a point in $\mathbb{K}^2$ with

$$A = \begin{pmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \\ 1 & 1 \end{pmatrix},\; L = \emptyset,\; \mathbf{a} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 3 \end{pmatrix},\; \text{and } \mathbf{r} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}.$$

The optimal value of the primal linear program (3.25) is $\lambda_0$, and the optimal solution of the dual linear program (3.26) is $\mathbf{u}_0$, with

$$\lambda_0 = -1,\; \mathbf{u}_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$



■

Hence, $\mathbf{n} = A^T\mathbf{u}_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and $c = \mathbf{a}^T\mathbf{u}_0 = 3$. But, $h_{\mathbf{P}}(\mathbf{n}) = 2 < c$. The last row of

$\mathbf{P}(A, L, \mathbf{a})$, which defines the half-space $\mathbf{H}\left(\begin{pmatrix}1\\1\end{pmatrix}, 3\right)$, is redundant for $\mathbf{P}$, but not for the polyhedron $\mathbf{P}(A, L, \mathbf{a}+\mathbf{1})$, for which $\mathbf{r}$ is a boundary point.

We conjecture that for a non-redundant $\mathcal{H}$-polyhedron one could use the half-space $\mathbf{H}(\mathbf{n}, c)$ instead of $\mathbf{H}(\mathbf{n}, h_{\mathbf{P}}(\mathbf{n}))$ in Item (2). Moreover, we conjecture that in this case $\mathbf{H}(\mathbf{n}, c)$ is a facet-defining half-space of $\mathbf{P}$ if and only if $\mathbf{r}$ is a relative interior point of a facet of $\mathbf{P}(A, L, \mathbf{a} - \mathbf{1}\lambda_0)$, which is likely the case for randomly chosen points $\mathbf{r} \notin \mathbf{P}$. Unfortunately, these conjectures do not seem to be generalizable to sops.

## 3.13. Interpolation

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. We may use Proposition 3.48 or Proposition 3.50 to find additional supporting 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 $\mathbf{P}'$.

Given a sop $\mathbf{P}$ and a boundary point $\mathbf{r}$ of an over-approximating $\mathcal{H}$-polyhedron $\mathbf{P}'$, ray shooting and face bloating provide a supporting half-space that either contains $\mathbf{r}$, i. e., $\mathbf{r}$ is also a boundary point of $\mathbf{P}$, or separates $\mathbf{r}$ from $\mathbf{P}$. Note that for ray shooting we have to ensure that (i) the origin $\mathbf{0}$ is a relative interior point of $\mathbf{P}$ and (ii) $\mathbf{r} \in \operatorname{aff}(\mathbf{P})$. This can be done with help of Proposition 3.47: Let $\mathbf{x}_0$ a relative interior point of $\mathbf{P}$. Translating $\mathbf{P}$ and $\mathbf{P}'$ by $-\mathbf{x}_0$ ensures conditions (i). Additionally, Proposition 3.47 also provides all implicit equalities of $\mathbf{P}$. This allows us to compute the affine hull of $\mathbf{P}$ and helps us to ensure condition (ii).

We use the following simple interpolation strategy, where the polyhedron $\mathbf{P}$ is given as a sop, and $\mathbf{P}'$ and $\mathbf{P}''$ are $\mathcal{H}$-polyhedra: If we decide to use the ray shooting method, we initialize $\mathbf{P}'' = \mathbf{P}(B, \mathbf{b})$ as the affine hull of $\mathbf{P}$. Otherwise, let $B$ and $\mathbf{b}$ be empty. Now, for an arbitrary inequality of $\mathbf{P}'$ we check 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 either ray shooting or face bloating on $\mathbf{P}$ and $\mathbf{r}$. The resulting half-space is then added to the representation of $\mathbf{P}''$. We proceed with any inequality of $\mathbf{P}'$ as before until all inequalities of $\mathbf{P}'$ are removed.

We observed that usage of the ray shooting method seems to provide better results, but may sometimes fail due to numerical issues. In this respect the face bloating method is more robust.

More sophisticated interpolation heuristics are possible but beyond the scope of this thesis.

## 3.14. Problem of Deciding the Subset Relation

We should address an open issue: Up to now, there is – to my best knowledge – no efficient method to decide subset relations for sops, and it is questionable whether efficient methods exists. Nevertheless, for any closed convex set for that a support function is available we can decide whether this set is the subset of an $\mathcal{H}$-polyhedron: Let

$\mathbf{P}_1 = \mathbf{P}\left(A_1, \mathbf{a}_1\right)$ be an $\mathcal{H}$-polyhedron and $\mathbf{P}_2$ be a closed convex set with support function $h_{\mathbf{P}_2}$, e. g. a sop of the form $\mathbf{P}_2 = \mathbf{P}\left(A_2, L_2, \mathbf{a}_2\right)$. Then $\mathbf{P}_2 \subseteq \mathbf{P}_1$ if and only if $\mathbf{h}_{\mathbf{P}_2}(A_1) \leq \mathbf{a}_1$.

# Part II.

# Reachability Analysis of Linear Hybrid Systems

# 4. Reachability Analysis of Linear Hybrid Systems

Zwei Seelen wohnen, ach! in meiner Brust, ...

Two souls, alas! reside within my breast, ...

JOHANN WOLFGANG VON GOETHE: Faust, 1808

The goal of this part is to describe and discuss the computational aspects of the reachability analysis of hybrid systems. We shall argue why sops are a suitable data structure for reachability analysis and present the reachable set computation as it is implemented in our prototype, a tool called SOAPBOX. While the chapter at hand mainly discusses the discrete transitions of a hybrid system and provides first impressions how computations can profit from the usage of sops, the usefulness of sops will entirely become clear in the upcoming chapters.

A hybrid system $H$ describes the discrete and continuous evolution of a system. Typically, we observe the evolution of $H$ with respect to additionally designated sets, which may include

- *Init*, the set of initial states,
- *Unsafe*, a set of states which we want to prove to be unreachable or reachable.

## 4.1. Interplay of the Discrete and Continuous Post Operator

Recall from the introduction that this thesis deals with the reachability analysis of linear hybrid system with polyhedral set representation. Let us, for a moment, restrict our attention to the discrete transitions. The discrete transitions of a linear hybrid system are given as guarded assignments of the form

$$\gamma = (\mathbf{G}, m, \mathbf{f}, m') \tag{4.1}$$

consisting of a polyhedral guard $\mathbf{G}$ together with an affine transformation $\mathbf{f}(\mathbf{x}) = M\mathbf{x}+\mathbf{v}$. If we now stipulate that *Init* is also given as a union of polyhedra, then the computation of the discrete postimage, $\mathrm{post}_d$, can be done efficiently by the usage of sops: We define a *symbolic state* to be a pair $(\mathbf{P}, m)$ of a polyhedron $\mathbf{P} \subseteq \mathbb{R}^d$ and a mode $m \in Mod$. Then the postimage of the symbolic state $(\mathbf{P}, m)$ under the guarded assignment (4.1) is the symbolic state

$$(M(\mathbf{P} \cap \mathbf{G}) + \mathbf{v}, m'). \tag{4.2}$$

In the first part of this thesis, we have learnt that sops can be used for an exact and efficient representation of $\mathbf{P}$ and $M(\mathbf{P} \cap \mathbf{G}) + \mathbf{v}$, see Proposition 3.9 and Proposition 3.13 on page 24. Note that, compared to $\mathcal{H}$-polyhedra, $\mathcal{V}$-polyhedra, and support functions, this is a unique feature of sops, see Table 1.1 on page 11.[1]

The discrete post operation, $\mathrm{post}_d$, comprises the individual application of guarded assignments to symbolic states: Let $\mathit{Trans} = \{\gamma_1, \ldots, \gamma_g\}$ be the set of guarded assignments, where each transition is of the form $\gamma_i = (\mathbf{G}_i, m_i, \mathbf{f}_i, m_i')$ with $\mathbf{f}_i(\mathbf{x}) = M_i\mathbf{x} + \mathbf{v}_i$. Then the discrete postimage of a symbolic state is given as

$$\mathrm{post}_d(\mathbf{P}, m) = \bigcup_{i=1,\ldots,g;\ m_i = m} (M_i(\mathbf{P} \cap \mathbf{G}_i) + \mathbf{v}_i, m_i'). \tag{4.3}$$

In analogy to the discrete postimage computation, we shall derive a variant of the continuous postimage computation which is compatible with the symbolic state representation. That is, the result of $\mathrm{post}_c(\mathbf{P}, m)$ shall be a disjunction of symbolic states,

$$\mathrm{post}_c(\mathbf{P}, m) = \bigcup_{i=I} (\mathbf{P}_i, m). \tag{4.4}$$

For the continuous postimage computation it is important to note that only those states $(\mathbf{x}, m)$ in $\mathrm{post}_c(\mathbf{P}, m)$ may jump – and, hence, have a discrete postimage – for which a guarded assignment of the form $\gamma = (\mathbf{G}, m, \mathbf{f}, m')$ with $\mathbf{x} \in \mathbf{G}$ exists. Hence, if we additionally stipulate that *Unsafe* is specified by guarded transitions whose target mode is a designated mode $u$, then it is sufficient that $\mathrm{post}_c(\mathbf{P}, m)$ returns those reachable states that are located in a mode-specific guard, leading to Algorithm 1.

Algorithm 1 has the following properties:

- Any trajectory of the hybrid system starts with a continuous flow. This commitment has only mild consequences: Only states $(\mathbf{x}, m)$ which initially satisfy a mode-specific guard have a non-trivial discrete postimage. On the other hand, the continuous postimage contains every state $(\mathbf{x}, m)$ that initially satisfies a mode-specific guard and also satisfies the invariant of the mode. Hence, only those states of *Init* are lost that do not fulfill the invariant of the associated mode. The question whether this restriction meets the expectation on the semantic of an invariant more adequately could be a matter of dispute.

- Looking at the discrete and continuous postimage computation (Lines 5-12) and their definitions (4.3), (4.4), a potential drawback of the chosen polyhedral state set representation becomes apparent. The number of generated symbolic states by the post-operators may explode after several applications.

[1] More precisely, it is the postimage computation which cannot be computed efficiently on $\mathcal{H}$-polyhedra. Analogous to Proposition 3.14 the preimage computation of $\mathcal{H}$-polyhedra can be done efficiently. Thus, a discrete backward reachability analysis – computing the preimages of *Unsafe* until *Init* is hit or a fix-point is reached – can be realized on $\mathcal{H}$-polyhedra.

**Algorithm 1** Reachability Algorithm for a Hybrid System

```
Input: A disjunction of symbolic states $\mathbf{R}_0$ representing the initial states, a designated
    mode $u$ representing Unsafe
Output: true if Unsafe is not reachable, false otherwise
1:  $k \leftarrow 0$
2:  repeat
3:    $\mathbf{C}_{k+1} \leftarrow \emptyset$;
4:    $\mathbf{R}_{k+1} \leftarrow \emptyset$;
5:    /* continuous postimage computation */
6:    for each $(\mathbf{P}, m) \in \mathbf{R}_k$ do
7:      $\mathbf{C}_{k+1} \leftarrow \mathbf{C}_{k+1} \cup \mathrm{post}_c(\mathbf{P}, m)$;
8:    end for;
9:    /* discrete postimage computation */
10:   for each $(\mathbf{P}, m) \in \mathbf{C}_{k+1}$ do
11:     $\mathbf{R}_{k+1} \leftarrow \mathbf{R}_{k+1} \cup \mathrm{post}_d(\mathbf{P}, m)$;
12:   end for;
13:   /* check for intersections with Unsafe */
14:   for each $(\mathbf{P}, m) \in \mathbf{R}_{k+1}$ do
15:     if $m = u$ then
16:       output "Unsafe is reachable!";
17:       return false;
18:     end if
19:   end for;
20:   $k \leftarrow k + 1$;
21: until $\mathbf{R}_{k+1} \subseteq \bigcup_{i=1,\ldots,k} \mathbf{R}_i$;
22: return true;
```

- There is no known efficient algorithm for the fix-point check in Line 21 for sops. The fix-point computation needs an efficient method to decide subset relations.[2] We already mentioned that sops have this defect in common with support functions.

We shall discuss some ideas which may help us to overcome the problems mentioned in the last two items. The obvious idea is to ignore the potential state set explosion and let the user provide a maximal depth of iteration, leading to bounded reachability analysis. This approach has been taken for our prototypical implementation SoapBox and has the advantage that we maintain a maximum of exactness. This approach could easily be extended by methods which try to avoid the state set explosion by heuristically compacting several states into a single symbolic state using the closed convex hull operation. Such heuristics are beyond the topic of this thesis.

Another possible resort is the usage of alternative intermediate state set representations. For example, SpaceEx uses template polyhedra as an intermediate representa-

[2] Actually, to perform the fix-point check in Line 21 we have to decide whether a convex set is the subset of a union of convex sets.

**Algorithm 2** Reachability Algorithm for a Hybrid System with Incomplete $\text{post}_c$

**Input:** A disjunction of symbolic states $\mathbf{R}_0$ representing the initial states, a designated mode $u$ representing *Unsafe*

**Output:** **true** if *Unsafe* is not reachable, **false** otherwise

```
1: $k \leftarrow 0$
2: $\mathbf{S}_0 \leftarrow \emptyset$
3: repeat
4:    $\mathbf{C}_{k+1} \leftarrow \emptyset$;
5:    $\mathbf{R}_{k+1} \leftarrow \mathbf{S}_k$;
6:    $\mathbf{S}_{k+1} \leftarrow \emptyset$;
7:    /* continuous postimage computation */
8:    for each $(\mathbf{P}, m) \in \mathbf{R}_k$ do
9:       $(\mathbf{C}_{k+1}, \mathbf{S}_{k+1}) \leftarrow (\mathbf{C}_{k+1}, \mathbf{S}_{k+1}) \cup \text{post}_c(\mathbf{P}, m)$;
10:   end for;
11:   /* discrete postimage computation */
12:   for each $(\mathbf{P}, m) \in \mathbf{C}_{k+1}$ do
13:      $\mathbf{R}_{k+1} \leftarrow \mathbf{R}_{k+1} \cup \text{post}_d(\mathbf{P}, m)$;
14:   end for;
15:   /* check for intersections with Unsafe */
16:   for each $(\mathbf{P}, m) \in \mathbf{R}_{k+1}$ do
17:      if $m = u$ then
18:         output "Unsafe is reachable!";
19:         return false;
20:      end if
21:   end for;
22:   $k \leftarrow k + 1$;
23: until $\mathbf{R}_{k+1} \subseteq \bigcup_{i=1,\ldots,k} \mathbf{R}_i$;
24: return true;
```

tion. For template polyhedra and, more general, for $\mathcal{H}$-polyhedra one can efficiently decide subset relations. Since sops have support functions, we also may use template polyhedra or the more precise interpolations between sops and template polyhedra to overcome the subset problem similarly. Actually, the course of the thesis will show that coupling of sops, support functions, and template polyhedra is benefiting.

In the following chapter, we will discuss the continuous postimage computation. The proposed algorithm will be bounded as it computes the reachable states up to a given time bound. Hence, we may deal with an incomplete continuous postimage operator $\text{post}_c(\mathbf{P}, m)$ that returns the pair $(\bigcup_{i \in I}(\mathbf{P}_i, m), (\mathbf{P}', m))$ consisting of (i) a union of symbolic states $(\mathbf{P}_i, m)$ that are reachable by letting an arbitrary amount of time elapse[3], i. e., it returns the reachable states within some time interval $[0, t]$, possibly restricted to the mode-specific guards, (ii) a symbolic state $(\mathbf{P}', m)$ that represents the reachable state

[3] We have to specify a fixed positive lower bound to ensure progress.

exactly at the instant of time $t$. Algorithm 2 states a variant of the reachability algorithm that can cope with the incomplete continuous post operator. It extends Algorithm 1 by a special treatment of $(\mathbf{P}', m)$ which ensures that the continuous postimage computation is resumed correctly.

# 5. Reachability Analysis of Linear Systems

Linear systems can be classified in various ways, for example, whether they have constant derivatives only, whether they allow bounded input functions or not, or whether they have invariants or not. Depending on the classification we present different sop-based methods for the reachable set computation.

The trajectories of a linear system are highly related to the solutions of a linear differential equation with bounded input functions. We show that for simple linear systems with constant derivatives it suffices to compute a bundle of affine solutions. The bundle can be described exactly by a single sop.

The reachable set computation for general systems results in considerable expenditures. The computation is done in a step-wise manner. We start with an initial flow segment that contains the reachable states within a small time interval of length $\delta$. In each step the segment is moved forward in time by $\delta$ time units. In doing so, the influences of bounded inputs and the invariant have to be incorporated, leading to a monotonic growth of the sops assembling these influences. While the assembly can be done efficiently, any evaluation of the accrued sops by means of linear programs gets increasingly harder. By combining the purely sop-based algorithm with a well-known algorithm based on support functions (LGG-algorithm, Le Guernic and Girard (2009)), we obtain a fast and precise reachability algorithm.

The reader should be familiar with the theory of solving linear differential equations as it could be found in Heuser (1995). Parts of this chapter have been published as Hagemann (2014a).

**Convention.** The set of reachable states of a linear system is not convex in general. For a better distinction we denote the exact set of reachable states by the symbol $\mathcal{R}$. Over-approximations of $\mathcal{R}$ are denoted by the regular symbol $\mathbf{R}$.

## 5.1. Variants of Linear Systems

### 5.1.1. Non-autonomous Linear System

The linear differential equation with constant coefficients

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + \mathbf{u}(t) \quad \text{where } \mathbf{u}(t) \in \mathbf{U} \text{ for all } t \geq 0 \tag{5.1}$$

defines a *non-autonomous* linear system. Any measurable function $\mathbf{u}$ with $\mathbf{u}(t) \in \mathbf{U}$ for all $t \geq 0$ is called an *admissible input function.* Note that we assume that $\mathbf{U}$ is convex and bounded in every direction, not least to avoid discussions about the meaning of arbitrary large derivatives. The class of all admissible input functions is denoted by $\mathit{AdmInp}(\mathbf{U})$. For any *initial state* $\mathbf{x}_0$, any instant of time $t \geq 0$, and any admissible input function $\mathbf{u}$ the system (5.1) has the unique solution

$$\mathbf{y}^A(t, \mathbf{x}_0, \mathbf{u}) = \mathrm{e}^{tA}\mathbf{x}_0 + \mathrm{e}^{tA}\int_0^t \mathrm{e}^{-\tau A}\mathbf{u}(\tau)\,\mathrm{d}\tau. \tag{5.2}$$

The solution $\mathbf{y}^A(t, \mathbf{x}_0, \mathbf{u})$ describes – as a function over time – a *trajectory*, which emanates from the initial state $\mathbf{x}_0$ at $t = 0$ and evolves accordingly to the input function $\mathbf{u}$ and the dynamics given by (5.1). Any state $\mathbf{z} = \mathbf{y}^A(t, \mathbf{x}_0, \mathbf{u})$ is said to be *reachable* from $\mathbf{x}_0$ at time $t$ under the input function $\mathbf{u}$. The set of reachable states of (5.1) from a set of initial states $\mathbf{X}_0$ at a certain instant of time $t \geq 0$ under all admissible inputs is given by

$$\mathcal{R}_t^{A,\mathbf{U}}(\mathbf{X}_0) = \left\{\mathbf{z} \,\middle|\, \exists \mathbf{x}_0 \in \mathbf{X}_0, \exists \mathbf{u} \in \mathit{AdmInp}(\mathbf{U})\colon\ \mathbf{z} = \mathbf{y}^A(t, \mathbf{x}_0, \mathbf{u})\right\}.$$

We will often omit the superscripts $A$ and $A, \mathbf{U}$ when it is clear which system is meant or the given statement is of general significance.

The set $\mathcal{R}(\mathbf{X}_0)$ of the reachable states from $\mathbf{X}_0$ at any time $t \geq 0$ and the set $\mathcal{R}_{[t_1,t_2]}(\mathbf{X}_0)$ of the reachable states of (5.1) from $\mathbf{X}_0$ within the time interval $[t_1, t_2]$ are defined as follows:

$$\mathcal{R}(\mathbf{X}_0) = \bigcup_{t \geq 0} \mathcal{R}_t(\mathbf{X}_0), \qquad \mathcal{R}_{[t_1,t_2]}(\mathbf{X}_0) = \bigcup_{t \in [t_1,t_2]} \mathcal{R}_t(\mathbf{X}_0).$$

### 5.1.2. Autonomous Linear System

A linear system that has no inputs, i.e., $\mathbf{U} = \{\mathbf{0}\}$, is called an *autonomous* linear system. An autonomous system evolves accordingly to the homogeneous linear differential equation

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t), \tag{5.3}$$

and has for any initial state $\mathbf{x}_0 \in \mathbf{X}_0$ and $t \geq 0$ the solution

$$\mathbf{y}(t, \mathbf{x}_0) = \mathbf{y}(t, \mathbf{x}_0, \mathbf{0}) = \mathrm{e}^{tA}\mathbf{x}_0. \tag{5.4}$$

The reachable states of an autonomous linear system are given by

$$\mathcal{R}_t^A(\mathbf{X}_0) = \left\{\mathbf{z} \,\middle|\, \exists \mathbf{x}_0 \in \mathbf{X}_0\colon\ \mathbf{z} = \mathrm{e}^{tA}\mathbf{x}_0\right\} = \bigcup_{\mathbf{x}_0 \in \mathbf{X}_0} e^{tA}\mathbf{x}_0.$$

### 5.1.3. Superposition Principle

For linear systems without invariants the *superposition principle* allows us to compute the solution $\mathbf{y}(t, \mathbf{x}_0, \mathbf{u})$ of a non-autonomous system as the sum of the solution $\mathbf{y}(t, \mathbf{x}_0) = \mathrm{e}^{tA}\mathbf{x}_0$ of the related autonomous system (5.3) and the particular solution $\mathbf{y}(t, \mathbf{0}, \mathbf{u}) = \mathrm{e}^{tA} \int_0^t \mathrm{e}^{-\tau A}\mathbf{u}(\tau)\,\mathrm{d}\tau$ of the non-autonomous system (5.1). It enables us to decompose the set of reachable states into the following Minkowski sum:

$$\mathcal{R}^{A,\mathbf{U}}(\mathbf{X}_0) = \mathcal{R}^{A}(\mathbf{X}_0) + \mathcal{R}^{A,\mathbf{U}}(\{\mathbf{0}\}). \tag{5.5}$$

The first summand $\mathcal{R}^{A}(\mathbf{X}_0)$ is the set of reachable states of the related autonomous system. The latter summand $\mathcal{R}^{A,\mathbf{U}}(\{\mathbf{0}\})$, which accounts for the influences of all admissible input functions, is independent of the choice of $\mathbf{X}_0$.

### 5.1.4. Non-autonomous Linear System with Invariant

An invariant $\mathbf{I}$ restricts the set of reachable states. Only those states $\mathbf{z}$ are reachable from $\mathbf{x}_0$ for which a trajectory $\mathbf{y}(t, \mathbf{x}_0, \mathbf{u})$ of the unrestricted system exists such that

(i) $\mathbf{x}_0$ is the initial state of $\mathbf{y}$, and $\mathbf{z}$ is a reachable state of $\mathbf{y}$, i. e.,

$$\mathbf{x}_0 = \mathbf{y}(0, \mathbf{x}_0, \mathbf{u}), \qquad \mathbf{z} = \mathbf{y}(t_0, \mathbf{x}_0, \mathbf{u}) \text{ for some } t_0 \geq 0,$$

(ii) $\mathbf{y}(t, \mathbf{x}_0, \mathbf{u}) \in \mathbf{I}$ for all intermediate instants of time $0 \leq t \leq t_0$.

The dynamic of a non-autonomous system with invariant $\mathbf{I}$ is given by the linear differential equation

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + \mathbf{u}(t) \quad \text{where } \mathbf{u}(t) \in \mathbf{U} \text{ and } \mathbf{x}(t) \in \mathbf{I} \text{ for all } t \geq 0. \tag{5.6}$$

Any valid solution of (5.6) is given by

$$\mathbf{y}^{\mathbf{I}}(t, \mathbf{x}_0, \mathbf{u}) = \begin{cases} \mathbf{y}(t, \mathbf{x}_0, \mathbf{u}) & \text{if } \mathbf{y}(\lambda, \mathbf{x}_0, \mathbf{u}) \in \mathbf{I} \text{ for all } 0 \leq \lambda \leq t \\ \emptyset & \text{otherwise} \end{cases}$$

where $\mathbf{y}(t, \mathbf{x}_0, \mathbf{u})$ is a solution of the unrestricted system. The set of reachable states of a linear system with invariant $\mathbf{I}$ is defined as

$$\mathcal{R}^{\mathbf{I}}(\mathbf{X}_0) = \left\{ \mathbf{z} \;\middle|\; \begin{array}{c} \exists \mathbf{x}_0 \in \mathbf{X}_0,\ \exists \mathbf{u}(t) \in \mathit{AdmInp}(\mathbf{U}),\ \exists t \geq 0\colon\ \mathbf{z} = \mathbf{y}(t, \mathbf{x}_0, \mathbf{u}) \\ \text{and } \forall 0 \leq \lambda \leq t\colon\ \mathbf{y}(\lambda, \mathbf{x}_0, \mathbf{u}) \in \mathbf{I} \end{array} \right\}.$$

Challenges of restricted systems are that we have to consider all intermediate solutions to ascertain the validity of a trajectory of the restricted system, and that the validity of the superposition principle may be broken by the invariant.

### 5.1.5. Autonomous Systems with Constant Derivatives

Computing the reachable states of an autonomous system with constant derivatives is simple. The derivative is fixed to a constant:

$$\dot{\mathbf{x}}(t) = \mathbf{c} \quad \text{where } \mathbf{c} \text{ is constant.} \tag{5.7}$$

For any initial state $\mathbf{x}_0$ and any instant of time the unrestricted system (5.7) has the solution

$$\mathbf{y}(t, \mathbf{x}_0) = \mathbf{x}_0 + t\mathbf{c}.$$

The restricted system (5.7) under the invariant $\mathbf{I}$ has the solution

$$\mathbf{y}^{\mathbf{I}}(t, \mathbf{x}_0, \mathbf{u}) = \begin{cases} \mathbf{x}_0 + t\mathbf{c} & \text{if } \mathbf{x}_0 + \lambda\mathbf{c} \in \mathbf{I} \text{ for all } 0 \leq \lambda \leq t \\ \emptyset & \text{otherwise.} \end{cases}$$

Hence, the reachable states are given by

$$\begin{aligned} \mathcal{R}(\mathbf{X}_0) &= \{\mathbf{z} \mid \exists \mathbf{x}_0 \in \mathbf{X}_0, \exists t \geq 0\colon\ \mathbf{z} = \mathbf{x}_0 + t\mathbf{c}\} = \mathbf{X}_0 + \bigcup_{t \geq 0} t\mathbf{c} \quad \text{and} \\ \mathcal{R}^{\mathbf{I}}(\mathbf{X}_0) &= \{\mathbf{z} \mid \exists \mathbf{x}_0 \in \mathbf{X}_0, \exists t \geq 0\colon\ \mathbf{z} = \mathbf{x}_0 + t\mathbf{c} \text{ and } \forall 0 \leq \lambda \leq t\colon\ \mathbf{x}_0 + \lambda\mathbf{c} \in \mathbf{I}\}\,. \end{aligned} \tag{5.8}$$

If the sets $\mathbf{X}_0$ and $\mathbf{I}$ are given as Boolean combinations of linear expression over the state variables. Then application of quantifier elimination of linear arithmetic to $\mathcal{R}(\mathbf{X}_0)$ and $\mathcal{R}^{\mathbf{I}}$ results in Boolean combination of linear expression over the state variables again.

Assume, $\mathbf{I}$ is convex. Since $\{\mathbf{x}_0 + \lambda\mathbf{c} \mid 0 \leq \lambda \leq t\}$ is a line segment, all intermediate points are located in $\mathbf{I}$ provided that the starting and end point are in $\mathbf{I}$. Hence, $\mathcal{R}^{\mathbf{I}}(\mathbf{X}_0)$ is obtained from $\mathcal{R}(\mathbf{X}_0)$ by the intersection:

$$\mathcal{R}^{\mathbf{I}}(\mathbf{X}_0) = \mathcal{R}(\mathbf{X}_0) \cap \mathbf{I} = \left(\mathbf{X}_0 + \bigcup_{t \geq 0} t\mathbf{c}\right) \cap \mathbf{I}.$$

If $\mathbf{X}_0$ and $\mathbf{I}$ are given as polyhedra, then each of the sets $\mathcal{R}(\mathbf{X}_0)$ and $\mathcal{R}^{\mathbf{I}}(\mathbf{X}_0)$ may be represented exactly as single sops, see Proposition 3.12 on page 23.

**Proposition 5.1 (Continuous Post Operator)**
Let $H = (\mathit{Var}, \mathit{Mod}, \mathit{Trans})$ be a linear hybrid system with polyhedral state set representation. Further, let $m \in \mathit{Mod}$ be a mode encoding an autonomous system with constant derivatives $\dot{\mathbf{x}} = \mathbf{c}$ and invariant $\mathbf{I}$. Then the exact continuous postimage of any symbolic state $(\mathbf{P}, m)$ is given by

$$\mathrm{post}_c(\mathbf{P}, m) = \left(\mathbf{P} + \bigcup_{t \geq 0} t\mathbf{c}\right) \cap \mathbf{I}. \qquad \blacksquare$$

### 5.1.6. Non-autonomous Systems with Constant Derivatives

We discuss the reachable set computation for a non-autonomous system with constant derivatives

$$\dot{\mathbf{x}}(t) = \mathbf{c} + \mathbf{u}(t) \quad \text{where } \mathbf{c} \text{ is constant and } \mathbf{u}(t) \in \mathbf{U}' \text{ for all } t \geq 0.$$

As a start, we note that setting $\mathbf{U} = \mathbf{c} + \mathbf{U}'$ yields the equivalent formulation

$$\dot{\mathbf{x}}(t) = \mathbf{u}(t) \quad \text{where } \mathbf{u}(t) \in \mathbf{U} \text{ for all } t \geq 0. \tag{5.9}$$

For any initial state $\mathbf{x}_0$ and any instant of time the unrestricted system (5.9) has the solution

$$\mathbf{y}(t, \mathbf{x}_0, \mathbf{u}) = \mathbf{x}_0 + \int_0^t \mathbf{u}(\tau)\,\mathrm{d}\tau.$$

The solution of the restricted system (5.9) under the invariant $\mathbf{I}$ is given by

$$\mathbf{y}^{\mathbf{I}}(t, \mathbf{x}_0, \mathbf{u}) = \begin{cases} \mathbf{x}_0 + \int_0^t \mathbf{u}(\tau)\,\mathrm{d}\tau & \text{if } \mathbf{x}_0 + \int_0^\lambda \mathbf{u}(\tau)\,\mathrm{d}\tau \in \mathbf{I} \text{ for all } 0 \leq \lambda \leq t \\ \emptyset & \text{otherwise.} \end{cases}$$

The reachable states are given by

$$\mathcal{R}(\mathbf{X}_0) = \left\{\mathbf{z} \,\middle|\, \exists \mathbf{x}_0 \in \mathbf{X}_0, \exists \mathbf{u} \in \mathit{AdmInp}(\mathbf{U}), \exists t \geq 0\colon\ \mathbf{z} = \mathbf{x}_0 + \textstyle\int_0^t \mathbf{u}(\tau)\,\mathrm{d}\tau\right\} \quad \text{and}$$
$$\mathcal{R}^{\mathbf{I}}(\mathbf{X}_0) = \left\{\mathbf{z} \,\middle|\, \begin{array}{c} \exists \mathbf{x}_0 \in \mathbf{X}_0, \exists \mathbf{u} \in \mathit{AdmInp}(\mathbf{U}), \exists t \geq 0\colon\ \mathbf{z} = \mathbf{x}_0 + \int_0^t \mathbf{u}(\tau)\,\mathrm{d}\tau \\ \text{and } \forall 0 \leq \lambda \leq t\colon\ \mathbf{x}_0 + \int_0^\lambda \mathbf{u}(\tau)\,\mathrm{d}\tau \in \mathbf{I} \end{array}\right\}.$$

At first glance, these sets have a more complicated structure than the reachable sets of the autonomous system. In addition, quantifier elimination of linear arithmetic seems not to be applicable, since $\mathbf{u} \in \mathit{AdmInp}(\mathbf{U})$ is not linear in general. Though, the next proposition reveals the simple structure of $\mathcal{R}(\mathbf{X}_0)$.

**Proposition 5.2 (Non-Autonomous Systems with Constant Derivatives)**

(i) Any reachable state of the unrestricted non-autonomous system (5.9) is reachable under a constant input function. In fact, the reachable states of the unrestricted system (5.9) are given as

$$\mathcal{R}(\mathbf{X}_0) = \mathbf{X}_0 + \bigcup_{t \geq 0} t\mathbf{U}. \tag{5.10}$$

(ii) Let $\mathbf{I}$ be a convex set and $\mathbf{X}_0 \subseteq \mathbf{I}$. Then any reachable state of the system (5.9) with invariant $\mathbf{I}$ is reachable under a constant input function. The reachable states are given as

$$\mathcal{R}^{\mathbf{I}}(\mathbf{X}_0) = \mathcal{R}(\mathbf{X}_0) \cap \mathbf{I} = \left(\mathbf{X}_0 + \bigcup_{t \geq 0} t\mathbf{U}\right) \cap \mathbf{I}. \qquad \blacksquare$$

*Proof.* (i) We prove equality (5.10): For any initial state $\mathbf{x}_0 \in \mathbf{X}_0$, any admissible input function $\mathbf{u}$, any instant of time $t \geq 0$, and any vector $\mathbf{n}$ the following relations hold:

$$\begin{aligned} \mathbf{n}^T\mathbf{y}(t, \mathbf{x}_0, \mathbf{u}) &= \mathbf{n}^T\mathbf{x}_0 + \mathbf{n}^T \int_0^t \mathbf{u}(\tau)\,\mathrm{d}\tau = \mathbf{n}^T\mathbf{x}_0 + \int_0^t \mathbf{n}^T\mathbf{u}(\tau)\,\mathrm{d}\tau \\ &\leq \mathbf{n}^T\mathbf{x}_0 + \int_0^t \sup_{\mathbf{u}\in\mathbf{U}} \mathbf{n}^T\mathbf{u}\,\mathrm{d}\tau = \mathbf{n}^T\mathbf{x}_0 + \sup_{\mathbf{u}\in t\mathbf{U}} \mathbf{n}^T\mathbf{u}. \end{aligned} \tag{5.11}$$

Hence, $\mathbf{y}(t, \mathbf{x}_0, \mathbf{u}) \in \mathbf{X}_0 + \bigcup_{t\geq 0} t\mathbf{U}$ and $\mathcal{R}(\mathbf{X}_0) \subseteq \mathbf{X}_0 + \bigcup_{t\geq 0} t\mathbf{U}$.

In order to fully establish (5.10), we note that any state $\mathbf{z} \in \mathbf{X}_0 + \bigcup_{t\geq 0} t\mathbf{U}$ can be written as $\mathbf{z} = \mathbf{x}_0 + t\mathbf{u}_0$ for suitable $\mathbf{x}_0 \in \mathbf{X}_0$, $\mathbf{u}_0 \in \mathbf{U}$ and $t \geq 0$. Hence, $\mathbf{z}$ is reachable under the constant input function $\mathbf{u}_0$, i. e., $\mathbf{z} = \mathbf{y}(t, \mathbf{x}_0, \mathbf{u}_0)$.

We have shown that any reachable state of the restricted system is in $\mathbf{X}_0 + \bigcup_{t\geq 0} t\mathbf{U}$ and, vice versa, that any state in $\mathbf{X}_0 + \bigcup_{t\geq 0} t\mathbf{U}$ is reachable under a constant input function.

(ii) Alas, the property that any reachable state is reachable under a constant input function does not carry over to systems with invariants. Because of the invariant, a state $\mathbf{z}$ might not be reachable by a affine function but it may be reachable by some trajectory which bends inside the invariant. Fortunately, in case of a convex invariant $\mathbf{I}$, the line segment from $\mathbf{x}_0$ to $\mathbf{z} = \mathbf{y}^{\mathbf{I}}(t, \mathbf{x}_0, \mathbf{u})$ is also contained in $\mathbf{I}$. Hence, like in the unrestricted case, any reachable state is reachable under a constant input function, and $\mathcal{R}^{\mathbf{I}}(\mathbf{X}_0)$ is obtained by the intersection:

$$\mathcal{R}^{\mathbf{I}}(\mathbf{X}_0) = \mathcal{R}(\mathbf{X}_0) \cap \mathbf{I}. \qquad \square$$

Hence, we have

$$\begin{aligned} \mathcal{R}(\mathbf{X}_0) &= \{\mathbf{z} \mid \exists \mathbf{x}_0 \in \mathbf{X}_0,\ \exists \mathbf{u} \in \mathbf{U}, \exists t \geq \mathbf{0}\colon\ \mathbf{z} = \mathbf{x}_0 + t\mathbf{u}\}\,, \\ \mathcal{R}(\mathbf{X}_0) \cap \mathbf{I} &= \{\mathbf{z} \mid \exists \mathbf{x}_0 \in \mathbf{X}_0,\ \exists \mathbf{u} \in \mathbf{U}, \exists t \geq \mathbf{0}\colon\ \mathbf{z} = \mathbf{x}_0 + t\mathbf{u} \text{ and } \forall 0 \leq \lambda \leq t\colon\ \mathbf{x}_0 + \lambda\mathbf{u} \in \mathbf{I}\} \end{aligned} \tag{5.12}$$

If the sets $\mathbf{X}_0$ and $\mathbf{I}$ are given as Boolean combinations of linear expression over the state variables and $\mathbf{I}$ is additionally convex, we may use a trick of Alur et al. (1996) which helps us to get rid of the non-linear expressions $t\mathbf{u}$ and $\lambda\mathbf{u}$ in (5.12)[1], see also Damm et al. (2012).

If $\mathbf{X}_0$, $\mathbf{U}$ and $\mathbf{I}$ are given as a polyhedron, then each of the sets $\mathcal{R}(\mathbf{X}_0)$ and $\mathcal{R}^{\mathbf{I}}(\mathbf{X}_0)$ may be represented exactly as single sops, see Proposition 3.44 on page 40.

**Proposition 5.3 (Continuous Post Operator)**
Let $H = (\mathit{Var}, \mathit{Mod}, \mathit{Trans})$ be a linear hybrid system with polyhedral state set representation. Further, let $m \in \mathit{Mod}$ be a mode encoding a non-autonomous system with derivatives $\dot{\mathbf{x}} \in \mathbf{U}$ and invariant $\mathbf{I}$. Then the continuous postimage of any symbolic state $(\mathbf{P}, m)$ is given by

$$\mathrm{post}_c(\mathbf{P}, m) = \left(\mathbf{P} + \bigcup_{t\geq 0} t\mathbf{U}\right) \cap \mathbf{I}. \qquad \blacksquare$$

[1] The trick is related to the scaling of sops in Section 3.6 on page 27. As for sops, one has to be careful if the trick is applied to unbounded polyhedral sets.

## 5.2. Time Discretization of Linear Systems

In this section we return to the general case of linear systems. While the reachable states of a system with constant derivatives can be computed exactly, this is no longer possible for general linear systems. It turns out that, for a fixed time step $\delta > 0$, there is an exact recurrence relation which relates the reachable states $\mathcal{R}_{[(k+1)\delta,(k+2)\delta]}(\mathbf{X}_0)$ to the reachable states $\mathcal{R}_{[k\delta,(k+1)\delta]}(\mathbf{X}_0)$ and $\mathcal{R}_\delta(\{\mathbf{0}\})$. Since the recurrence relation only involves linear mappings and Minkowski sums, it may easily expressed in terms of sops. The recurrence relation shifts the computational work to the initial sets, i. e., the identification of $\mathcal{R}_{[0,\delta]}(\mathbf{X}_0)$ and $\mathcal{R}_\delta(\{\mathbf{0}\})$.

In addition to the spatial decomposition of the solution that is based on the superposition principle, we will derive a recurrence relation which is based on a temporal decomposition of the solutions. The matrix exponential[2] obeys the exponentiation identity

$$\mathrm{e}^{(t_1+t_2)A} = \mathrm{e}^{t_1 A}\mathrm{e}^{t_2 A}.$$

Hence, the solution $\mathbf{y}(\delta + k\delta, \mathbf{x}_0)$ of an autonomous linear system can be decomposed into

$$\mathbf{y}(\delta + k\delta, \mathbf{x}_0) = \mathrm{e}^{(\delta+k\delta)A}\mathbf{x}_0 = \mathrm{e}^{\delta A}\mathrm{e}^{k\delta A}\mathbf{x}_0 = \mathrm{e}^{\delta A}\mathbf{y}(k\delta, \mathbf{x}_0).$$

Therefore, the reachable states from $\mathbf{x}_0$ at the time instants $k\delta$, $k = 0, 1, 2, \ldots$, can be computed by repeated application of the matrix exponential $\mathrm{e}^{\delta A}$, yielding

$$\mathbf{y}(0, \mathbf{x}_0) = \mathbf{x}_0, \quad \mathbf{y}(\delta, \mathbf{x}_0) = \mathrm{e}^{\delta A}\mathbf{x}_0, \quad \mathbf{y}(2\delta, \mathbf{x}_0) = \mathrm{e}^{\delta A}(\mathrm{e}^{\delta A}\mathbf{x}_0), \quad \ldots$$

The next three lemmata yield a time discretization of $\mathcal{R}(\mathbf{X}_0)$ as a union of segments $\mathcal{R}_{[k\delta,(k+1)\delta]}(\mathbf{X}_0)$, $k = 0, 1, 2\ldots$, over time intervals of equal length $\delta > 0$. Lemma 5.4 and the first item of Lemma 5.5 are taken from Chutinan (1999) and slightly extended to non-constant input functions. Lemma 5.6 is widely used by several authors (Girard, 2005; Le Guernic, 2009; Althoff et al., 2010; Frehse et al., 2011).

**Lemma 5.4 (Temporal Decomposition I)**
Let $\mathbf{x}_0$ be an initial state, $\mathbf{u}$ an admissible input function, and $t$ an instant of time. Then for $h \geq 0$ the following equation holds:

$$\mathbf{y}(t + h, \mathbf{x}_0, \mathbf{u}) = \mathrm{e}^{tA}\mathbf{y}(h, \mathbf{x}_0, \mathbf{u}) + \mathrm{e}^{tA}\int_0^t \mathrm{e}^{-\tau A}\mathbf{u}(\tau + h)\,\mathrm{d}\tau. \qquad \blacksquare$$

[2] Note that we do not address the problem of computing the matrix exponential

$$\mathrm{e}^{\delta A} = \sum_{k=0}^{\infty} \frac{\delta^k A^k}{k!}$$

in this thesis. There are two problems in this context. First of all, the matrix exponential $\mathrm{e}^{\delta A}$ of a rational matrix $A$ has – in general – transcendental coefficients and, hence, it is impossible to represent the coefficients as floating point or rational numbers exactly. Secondly, due to catastrophic cancellation even the computation of an approximation of matrix exponential might be hard. A good survey on the computation of an approximation may be found in Moler and Van Loan (2003).

*Proof.*

$$\begin{aligned}\mathbf{y}(t+h,\mathbf{x}_0,\mathbf{u}) &= \mathrm{e}^{(t+h)A}\mathbf{x}_0 + \mathrm{e}^{(t+h)A}\int_0^{t+h}\mathrm{e}^{-\tau A}\mathbf{u}(\tau)\,\mathrm{d}\tau\\ &= \mathrm{e}^{(t+h)A}\mathbf{x}_0 + \mathrm{e}^{(t+h)A}\int_0^{h}\mathrm{e}^{-\tau A}\mathbf{u}(\tau)\,\mathrm{d}\tau + \mathrm{e}^{(t+h)A}\int_h^{t+h}\mathrm{e}^{-\tau A}\mathbf{u}(\tau)\,\mathrm{d}\tau\\ &= \mathrm{e}^{tA}\left(\mathrm{e}^{hA}\mathbf{x}_0 + \mathrm{e}^{hA}\int_0^{h}\mathrm{e}^{-\tau A}\mathbf{u}(\tau)\,\mathrm{d}\tau\right) + \mathrm{e}^{tA}\int_h^{t+h}\mathrm{e}^{-(\tau-h)A}\mathbf{u}(\tau)\,\mathrm{d}\tau\\ &= \mathrm{e}^{tA}\mathbf{y}(h,\mathbf{x}_0,\mathbf{u}) + \mathrm{e}^{tA}\int_0^{t}\mathrm{e}^{-\tau A}\mathbf{u}(\tau+h)\,\mathrm{d}\tau. \qquad \square\end{aligned}$$

**Lemma 5.5 (Temporal Decomposition II)**
For any time step $\delta \geq 0$ and any integer $k \geq 0$ the following equalities hold:

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

(ii) $\mathcal{R}_{(k+1)\delta}(\{\mathbf{0}\}) = \mathrm{e}^{\delta A}\,\mathcal{R}_{k\delta}(\{\mathbf{0}\}) + \mathcal{R}_\delta(\{\mathbf{0}\})$. ■

*Proof.* (i) Any state $\mathbf{z}$ is reachable from $\mathbf{X}_0$ in the time interval $[t, t+\delta]$ if and only if there exists an initial state $\mathbf{x}_0 \in \mathbf{X}_0$, an admissible input function $\mathbf{u}$, and an instant of time $h \in [0,\delta]$ such that $\mathbf{z} = \mathbf{y}(t+h,\mathbf{x}_0,\mathbf{u})$. By Lemma 5.4 we have the equality

$$\mathbf{z} = \mathrm{e}^{tA}\mathbf{y}(h,\mathbf{x}_0,\mathbf{u}) + \mathrm{e}^{tA}\int_0^{t}\mathrm{e}^{-\tau A}\mathbf{u}(\tau+h)\,\mathrm{d}\tau,$$

where $\mathbf{u}(t+h)$ is an admissible input function if and only if $\mathbf{u}(t)$ is an admissible input function. Hence,

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

(ii) A state $\mathbf{z}$ is reachable from $\{\mathbf{0}\}$ at time $(k+1)\delta$ if and only if there exists an admissible input function $\mathbf{u}$ such that $\mathbf{z} = \mathbf{y}((k+1)\delta,\mathbf{0},\mathbf{u})$. By Lemma 5.4 we have the equality

$$\mathbf{z} = \mathrm{e}^{\delta A}\mathbf{y}(k\delta,\mathbf{0},\mathbf{u}) + \mathrm{e}^{\delta A}\int_0^{\delta}\mathrm{e}^{-\tau A}\mathbf{u}(\tau+k\delta)\,\mathrm{d}\tau,$$

where $\mathbf{u}(t+k\delta)$ is an admissible input function if and only if $\mathbf{u}(t)$ is an admissible input function. Hence,

$$\mathbf{z} \in \mathcal{R}_{(k+1)\delta}(\{\mathbf{0}\}) \iff \mathbf{z} \in \mathrm{e}^{\delta A}\,\mathcal{R}_{k\delta}(\{\mathbf{0}\}) + \mathcal{R}_\delta(\{\mathbf{0}\}). \qquad \square$$

**Lemma 5.6 (Time Discretization of the Set of Reachable States)**
For all integers $k \geq 0$ and any time steps $\delta \geq 0$ the following recurrence relation on the segments $\mathcal{R}_{[k,(k+1)]}(\mathbf{X}_0)$ holds:

$$\mathcal{R}_{[(k+1)\delta,(k+2)\delta]}(\mathbf{X}_0) = \mathrm{e}^{\delta A}\,\mathcal{R}_{[k\delta,(k+1)\delta]}(\mathbf{X}_0) + \mathcal{R}_\delta(\{\mathbf{0}\}). \qquad ■$$

*Proof.*

$$\begin{aligned}\mathcal{R}_{[(k+1)\delta,(k+2)\delta]}(\mathbf{X}_0) &= \mathrm{e}^{(k+1)\delta A}\,\mathcal{R}_{[0,\delta]}(\mathbf{X}_0) + \mathcal{R}_{(k+1)\delta}(\{\mathbf{0}\}) && \text{(by Lemma 5.5 (i))}\\ &= \mathrm{e}^{(k+1)\delta A}\,\mathcal{R}_{[0,\delta]}(\mathbf{X}_0) + \mathrm{e}^{\delta A}\,\mathcal{R}_{k\delta}(\{\mathbf{0}\}) + \mathcal{R}_{\delta}(\{\mathbf{0}\}) && \text{(by Lemma 5.5 (ii))}\\ &= \mathrm{e}^{\delta A}(\mathrm{e}^{k\delta A}\,\mathcal{R}_{[0,\delta]}(\mathbf{X}_0) + \mathcal{R}_{k\delta}(\{\mathbf{0}\})) + \mathcal{R}_{\delta}(\{\mathbf{0}\})\\ &= \mathrm{e}^{\delta A}\,\mathcal{R}_{[k\delta,(k+1)\delta]}(\mathbf{X}_0) + \mathcal{R}_{\delta}(\{\mathbf{0}\}). && \text{(by Lemma 5.5 (i))}\end{aligned}$$

□

The set $\mathcal{R}_{[0,\delta]}(\mathbf{X}_0)$ contains all reachable states of the non-autonomous system from $\mathbf{X}_0$ within the time interval $[0,\delta]$, and the set $\mathcal{R}_{\delta}(\{\mathbf{0}\})$ accounts for the influences of all admissible input functions $\mathbf{u} \in \mathit{AdmInp}(\mathbf{U})$ at the instant of time $\delta$. Given the initial segment $\mathbf{R}_0 = \mathcal{R}_{[0,\delta]}(\mathbf{X}_0)$, the set $\mathbf{V} = \mathcal{R}_{\delta}(\{\mathbf{0}\})$, and the matrix exponential $\mathrm{e}^{\delta A}$, the time discretization lemma enables us to recursively compute the reachable states up to any instant of time $t = k\delta$, $k \geq 0$:

$$\mathbf{R}_{k+1} = \mathrm{e}^{\delta A}\mathbf{R}_k + \mathbf{V}.$$

In practice, we will not apply the recurrence relation to the exact initial sets $\mathcal{R}_{[0,\delta]}(\mathbf{X}_0)$ and $\mathcal{R}_{\delta}(\{\mathbf{0}\})$, but we apply it to bloated and convexified over-approximations of these sets, i. e.,

$$\operatorname{clconv}(\mathcal{R}_{[0,\delta]}(\mathbf{X}_0)) \subseteq \mathbf{R}_0 \quad \text{and} \quad \operatorname{clconv}(\mathcal{R}_{\delta}(\{\mathbf{0}\})) \subseteq \mathbf{V}.$$

All subsequent segments $\mathbf{R}_k$, $k > 0$, are computed according to the *exact* recurrence relation. Therefore, beside the initial bloating, no further approximations are made during the reachable set computation. In particular, the reachable set computation provides a *wrapping-free* over-approximation of $\mathcal{R}_{[0,k\delta]}(\mathbf{X}_0)$ (Le Guernic, 2009) if we have the matrix exponential $\mathrm{e}^{\delta A}$ at hand.[3]

### 5.2.1. Incorporating the Invariant

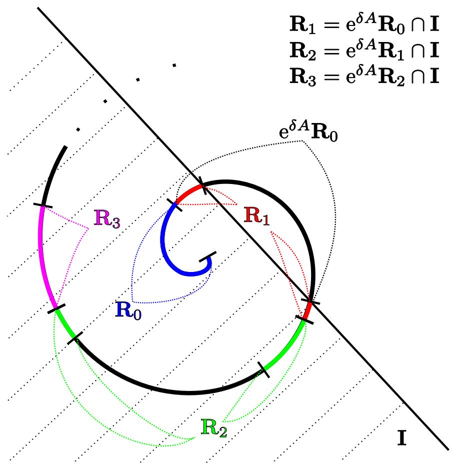
We would like to respect the influences of the invariant $\mathbf{I}$. Hence, we modify the recursive computation of the reachable states as follows

$$\mathbf{R}_{k+1} = (\mathrm{e}^{\delta A}\mathbf{R}_k + \mathbf{V}) \cap \mathbf{I}. \tag{5.13}$$

This discretization has been proposed and discussed by Le Guernic (2009) and Le Guernic and Girard (2009). Since support functions allow insufficient handling of intersections only, the authors propose further over-approximations of (5.13). The probably coarsest over-approximation of (5.13) may be found in Frehse et al. (2011) where the influence of the invariant is ignored almost completely. In fact, the following recurrence relation is used:

$$\mathbf{R}_{k+1} = \begin{cases} \emptyset & \text{if } (\mathrm{e}^{\delta A}\mathbf{R}_k + \mathbf{V}) \cap \mathbf{I} = \emptyset,\\ \mathrm{e}^{\delta A}\mathbf{R}_k + \mathbf{V} & \text{else.}\end{cases}$$

[3] See footnote 2 for the computation of the matrix exponential.



$\mathbf{R}_1 = e^{\delta A}\mathbf{R}_0 \cap \mathbf{I}$
$\mathbf{R}_2 = e^{\delta A}\mathbf{R}_1 \cap \mathbf{I}$
$\mathbf{R}_3 = e^{\delta A}\mathbf{R}_2 \cap \mathbf{I}$
$e^{\delta A}\mathbf{R}_0$
$\mathbf{R}_3$
$\mathbf{R}_1$
$\mathbf{R}_0$
$\mathbf{R}_2$
$\mathbf{I}$

*The figure shows artifacts of the time discretization* (5.13)*. The spiral is a valid trajectory of some system without invariant. The initial segment of the trajectory is* $\mathbf{R}_0$*. The actual set of reachable states of the corresponding system with invariant* $\mathbf{I}$ *consists of the initial segment of the trajectory until the invariant is left for the first time. The colored segments show the computed reachable sets according to the time discretization* (5.13) *respecting the invariant* $\mathbf{I}$*.*

Figure 5.1.: Artifacts of the Time Discretization with Invariants

Alas, even if we use the discretization (5.13) without further over-approximation, the discretization is not exact: The invariant is only applied at certain instants of time, namely at time $\delta$, $2\delta$, $3\delta$, .... Within the intermediate time instants, the influence of the invariant is ignored, which may result in an over-approximation. Even not connected trajectories could occur as artifacts of the discretization, as Figure 5.1 shows.

### 5.2.2. Initial Bloating

In this section we shall 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 the conservative over-approximation as given by Girard (2005)[4] to an accurate bloating of the convex hull $\operatorname{clconv}(\mathbf{X}_0 \cup e^{\delta A}\mathbf{X}_0)$ as proposed by Chutinan and Krogh (1998).

The novel method given below is inspired by a method proposed by Le Guernic (2009), which has been slightly improved and implemented in SpaceEx, see Frehse et al. (2011). The bloating method of Le Guernic is not applicable in our context, since we are dealing with polyhedral sets only, while Le Guernic's method involves piecewise quadratic functions to describe the support function of the bloated sets. Although we made no effort to give a precise comparison of both bloating methods, we expect the support function-

[4] Let $||\cdot||$ be a norm and let $\Box(\alpha)$ denotes the ball of radius $\alpha$ corresponding to $||\cdot||$. Then the conservative over-approximations $\mathcal{R}_\delta(\{\mathbf{0}\}) \subseteq \Box(\beta_\delta)$ and $\mathcal{R}_{[0,\delta]}(\mathbf{X}_0) \subseteq \operatorname{clconv}(\mathbf{X}_0 \cup e^{\delta A}\mathbf{X}_0) + \Box(\alpha_\delta + \beta_\delta)$ with $\beta_\delta = \frac{e^{\delta ||A||} - 1}{||A||} \sup_{\mathbf{u} \in \mathbf{U}} ||u||$ and $\alpha_\delta = (e^{\delta ||A||} - 1 - \delta\, ||A||) \sup_{\mathbf{x}_0 \in \mathbf{X}_0} ||x_0||$ hold.

based method to provide better results in general. However, since sops also have support functions, the following bloating procedure may also be applied to reachability analysis using support functions. A detailed comparison of both bloating methods is considered as future work.

We use the decomposition (5.5),

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

and over-approximate both summands separately. They are added afterwards to obtain an over-approximation of the reachable states of the non-autonomous system.

Let $\mathbf{e}_i$ the standard basis vector $\mathbf{e}_i = (0 \quad \dots \quad 0 \quad 1 \quad 0 \quad \dots \quad 0)^T$, whose only non-zero coefficient is at the $i$th place. For the following let $\boxdot(\mathbf{X})$ be the symmetric interval hull of $\mathbf{X}$, defined as

$$\boxdot(\mathbf{X}) = [-z_1, z_1] \times \cdots \times [-z_d, z_d] \text{ where } z_i = \max\left( \left| \inf_{\mathbf{x}\in\mathbf{X}} \mathbf{e}_i^T \mathbf{x} \right|, \left| \sup_{\mathbf{x}\in\mathbf{X}} \mathbf{e}_i^T \mathbf{x} \right| \right).$$

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 $\mathbf{x}$, matrices $A$ and $B$ we have (i) $\mathbf{x} \leq |\mathbf{x}|$, (ii) $|A + B| \leq |A| + |B|$, and (iii) $|A\mathbf{x}| \leq |A|\,|\mathbf{x}|$. We define the abbreviation $[\![\mathrm{e}^{\delta A}\mathbf{X}]\!] = \boxdot\left(\mathrm{e}^{\delta|A|}(\boxdot(\mathbf{X}))\right)$ and obtain an over-approximation of $\mathrm{e}^{tA}\mathbf{X}_0$ by the next small lemma.

**Lemma 5.7**
For all $t \in [0, \delta]$ the set inclusion $\mathrm{e}^{tA}\mathbf{X} \subseteq [\![\mathrm{e}^{\delta A}\mathbf{X}]\!]$ holds. ■

*Proof.* Let $\mathbf{y} = \mathrm{e}^{tA}\mathbf{x}$ for some $\mathbf{x} \in \mathbf{X}$ and $t \in [0, \delta]$. Then

$$\mathbf{y} \leq |\mathbf{y}| = \left|\mathrm{e}^{tA}\mathbf{x}\right| \leq \left|\mathrm{e}^{tA}\right| |\mathbf{x}| \leq \mathrm{e}^{t|A|}\,|\mathbf{x}| \leq \mathrm{e}^{\delta|A|}\,|\mathbf{x}|\,. \qquad \square$$

The next lemma is based on a Taylor approximation of $m$th order and an over-approximation of the Lagrange form of the remainder.

**Lemma 5.8**
For any $m \geq 0$ and any $t \in [0, \delta]$ the following set inclusions hold:

$$\mathrm{e}^{tA}\mathbf{X}_0 \subseteq \sum_{k=0}^{m} \frac{t^k}{k!} A^k \mathbf{X}_0 + \frac{t^{m+1}}{(m+1)!} A^{m+1} [\![\mathrm{e}^{\delta A}\mathbf{X}_0]\!], \tag{5.14}$$

$$\mathcal{R}_t(\{\mathbf{0}\}) \subseteq \sum_{k=0}^{m} \frac{t^{k+1}}{(k+1)!} A^k \mathbf{U} + \frac{t^{m+2}}{(m+2)!} A^{m+1} [\![\mathrm{e}^{\delta A}\mathbf{U}]\!]. \tag{5.15}$$

■

*Proof.* We show (5.14). For all $\mathbf{x}_0 \in \mathbf{X}_0$ and all $\mathbf{n} \in \mathbb{R}^d$ the following Taylor approximation with Lagrange remainder holds:

$$\mathbf{n}^T \mathrm{e}^{tA}\mathbf{x}_0 = \sum_{k=0}^{m} \frac{t^k}{k!} \mathbf{n}^T A^k \mathbf{x}_0 + \frac{t^{m+1}}{(m+1)!} \mathbf{n}^T A^{m+1} \mathrm{e}^{\tau A}\mathbf{x}_0 \quad \text{where } \tau \in [0, t).$$

Using the notation of support functions and Lemma 5.7 yields

$$\mathbf{n}^T \mathrm{e}^{tA}\mathbf{x}_0 \leq \sum_{k=0}^{m} \frac{t^k}{k!} h_{A^k(\mathbf{X}_0)}(\mathbf{n}) + \frac{t^{m+1}}{(m+1)!} h_{A^{m+1}[\![\mathrm{e}^{\delta A}\mathbf{X}_0]\!]}(\mathbf{n}).$$

Hence, $\mathrm{e}^{tA}\mathbf{X}_0 \subseteq \sum_{k=0}^{m} \frac{t^k}{k!} A^k \mathbf{X}_0 + \frac{t^{m+1}}{(m+1)!} A^{m+1}[\![\mathrm{e}^{\delta A}\mathbf{X}_0]\!]$. In order to show (5.15), we use the identity

$$\mathcal{R}_t(\{\mathbf{0}\}) = \left\{ \int_0^t \mathrm{e}^{(t-\tau)A}\mathbf{u}(\tau)\,\mathrm{d}\tau \,\middle|\, \mathbf{u} \in \mathit{AdmInp}(\mathbf{U}) \right\}.$$

Let $t \in [0, \delta]$ and $\mathbf{u}$ be an admissible input function. We set $\mathbf{x} = \int_0^t \mathrm{e}^{(t-\tau)A}\mathbf{u}(\tau)\,\mathrm{d}\tau$. For all $\mathbf{n} \in \mathbb{R}^d$ it holds

$$\mathbf{n}^T\mathbf{x} = \mathbf{n}^T \int_0^t \mathrm{e}^{(t-\tau)A}\mathbf{u}(\tau)\,\mathrm{d}\tau = \int_0^t \mathbf{n}^T \mathrm{e}^{(t-\tau)A}\mathbf{u}(\tau)\,\mathrm{d}\tau \leq \int_0^t \sup_{\mathbf{u}\in\mathbf{U}} \mathbf{n}^T \mathrm{e}^{(t-\tau)A}\mathbf{u}\,\mathrm{d}\tau.$$

By Taylor approximation with Lagrange remainder we obtain a function $\xi(t)$ with $\tau \leq \xi(t) \leq t$ and

$$\begin{aligned}
\mathbf{n}^T\mathbf{x} &\leq \int_0^t \sup_{\mathbf{u}\in\mathbf{U}} \left( \sum_{k=0}^{m} \frac{(t-\tau)^k}{k!} \mathbf{n}^T A^k \mathbf{u} + \frac{(t-\tau)^{m+1}}{(m+1)!} \mathbf{n}^T A^{m+1} \mathrm{e}^{(\xi(t)-\tau)A}\mathbf{u} \right) \mathrm{d}\tau \\
&\leq \int_0^t \sum_{k=0}^{m} \frac{(t-\tau)^k}{k!} \sup_{\mathbf{u}\in\mathbf{U}} \mathbf{n}^T A^k \mathbf{u} + \frac{(t-\tau)^{m+1}}{(m+1)!} \sup_{\mathbf{u}\in\mathbf{U}} \mathbf{n}^T A^{m+1} \mathrm{e}^{(\xi(t)-\tau)A}\mathbf{u}\,\mathrm{d}\tau \\
&= \int_0^t \sum_{k=0}^{m} \frac{(t-\tau)^k}{k!} \sup_{\mathbf{u}\in\mathbf{U}} \mathbf{n}^T A^k \mathbf{u}\,\mathrm{d}\tau \\
&\quad + \int_0^t \frac{(t-\tau)^{m+1}}{(m+1)!} \sup_{\mathbf{u}\in\mathbf{U}} \mathbf{n}^T A^{m+1} \mathrm{e}^{(\xi(t)-\tau)A}\mathbf{u}\,\mathrm{d}\tau \\
&= \sum_{k=0}^{m} \int_0^t \frac{(t-\tau)^k}{k!} \sup_{\mathbf{u}'\in A^k\mathbf{U}} \mathbf{n}^T \mathbf{u}'\,\mathrm{d}\tau \\
&\quad + \int_0^t \frac{(t-\tau)^{m+1}}{(m+1)!} \sup_{\mathbf{u}'\in \mathrm{e}^{(\xi(t)-\tau)A}\mathbf{U}} \mathbf{n}^T A^{m+1} \mathbf{u}'\,\mathrm{d}\tau.
\end{aligned}$$

We use the notation of support functions and rewrite to

$$\begin{aligned}
\mathbf{n}^T\mathbf{x} \leq \sum_{k=0}^{m} \int_0^t \frac{(t-\tau)^k}{k!} h_{A^k\mathbf{U}}(\mathbf{n})\,\mathrm{d}\tau \\
+ \int_0^t \frac{(t-\tau)^{m+1}}{(m+1)!} h_{\mathrm{e}^{(\xi(t)-\tau)A}\mathbf{U}}((A^{m+1})^T\mathbf{n})\,\mathrm{d}\tau.
\end{aligned}$$

Since $\xi(t) - \tau \in [0, t-\tau] \subseteq [0, \delta]$ holds, we may eliminate the reference to $\xi(t) - \tau$ using Lemma 5.7, $e^{tA}\mathbf{X} \subseteq [\![e^{\delta A}\mathbf{X}]\!]$ for all $t \in [0, \delta]$:

$$\begin{aligned}
\mathbf{n}^T\mathbf{x} &\le \sum_{k=0}^{m} \int_0^t \frac{(t-\tau)^k}{k!} h_{A^k\mathbf{U}}(\mathbf{n})\,\mathrm{d}\tau + \int_0^t \frac{(t-\tau)^{m+1}}{(m+1)!} h_{[\![e^{\delta A}\mathbf{U}]\!]}((A^{m+1})^T\mathbf{n})\,\mathrm{d}\tau \\
&= \sum_{k=0}^{m} \int_0^t \frac{(t-\tau)^k}{k!} h_{A^k\mathbf{U}}(\mathbf{n})\,\mathrm{d}\tau + \int_0^t \frac{(t-\tau)^{m+1}}{(m+1)!} h_{A^{m+1}[\![e^{\delta A}\mathbf{U}]\!]}(\mathbf{n})\,\mathrm{d}\tau \\
&= \sum_{k=0}^{m} \frac{t^{k+1}}{(k+1)!} h_{A^k\mathbf{U}}(\mathbf{n}) + \frac{t^{m+2}}{(m+2)!} h_{A^{m+1}[\![e^{\delta A}\mathbf{U}]\!]}(\mathbf{n}),
\end{aligned}$$

for all $\mathbf{x} = \int_0^t e^{(t-\tau)A}\mathbf{u}(\tau)$. Hence,

$$\begin{aligned}
\sup_{\mathbf{x}\in\mathcal{R}_t(\{\mathbf{0}\})} \mathbf{n}^T\mathbf{x} &= h_{\mathcal{R}_t(\{\mathbf{0}\})}(\mathbf{n}) \\
&\le \sum_{k=0}^{m} \frac{t^{k+1}}{(k+1)!} h_{A^k\mathbf{U}}(\mathbf{n}) + \frac{t^{m+2}}{(m+2)!} h_{A^{m+1}[\![e^{\delta A}\mathbf{U}]\!]}(\mathbf{n})
\end{aligned}$$

and

$$\mathcal{R}_t(\{\mathbf{0}\}) \subseteq \sum_{k=0}^{m} \frac{t^{k+1}}{(k+1)!} A^k\mathbf{U} + \frac{t^{m+2}}{(m+2)!} A^{m+1}[\![e^{\delta A}\mathbf{U}]\!]. \qquad \square$$

For $t = \delta$, (5.15) already provides an over-approximation of $\mathcal{R}_\delta(\{\mathbf{0}\})$. We choose $m = 0$ and obtain the first-order approximation

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

We use that for any $\mathbf{x}$, $k \ge 0$, and $t \in [0, \delta]$ the term $\frac{t^k}{k!}A^k\mathbf{x}$ may 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, $0 \le \lambda \le 1$. We introduce the notion $\lhd(\mathbf{X}) = \mathrm{clconv}(\{\mathbf{0}\} \cup \mathbf{X})$ and obtain a first-order approximations of $\bigcup_{t\in[0,\delta]} e^{tA}\mathbf{X}_0$ and $\bigcup_{t\in[0,\delta]} \mathcal{R}_t(\{\mathbf{0}\})$ as stated in the following lemma.

**Lemma 5.9**
The following set inclusions hold

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

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

**Proposition 5.10 (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:

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

## 5.3. Reachability Algorithm for Linear Systems with Invariants (Algorithm 3)

Having extensively discussed the various aspects of the step-wise reachable set computation, it is now time to state an algorithm which computes the output of the continuous post operator $\mathrm{post}_c(\mathbf{P}, m)$. Algorithm 3 computes the reachable states of a linear system, and it is a variant of the algorithms presented in Le Guernic (2009); Le Guernic and Girard (2009). If we modify the algorithm in such a way that it additionally returns the last segment $\mathbf{R}_{k+1}$ when the postimage computation was unfinished, we obtain an incomplete postimage computation which perfectly fits the needs of Algorithm 2, a reachability algorithm for hybrid systems. The inputs of the algorithm are the first flow

**Algorithm 3** Reachability Algorithm for a Linear System (SOP)

**Input:** a time step $\delta > 0$, 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 over-approximations of intersections of $\mathcal{R}_{[0,t]}(\mathbf{X}_0)$ and the guards in $\mathcal{G}$.

```
1: for $k \leftarrow 0, \dots, N$ 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;
```

segment $\mathbf{R}_0$, the set $\mathbf{V}$ – both obtained by the initial bloating procedure –, the invariant $\mathbf{I}$, and the set $\mathcal{G} = \{\mathbf{G}_1, \dots, \mathbf{G}_g\}$ of guards. The computation of the next flow segment in Line 6 is based on the discretization for linear systems with invariants (5.13). 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, or returning the convex hull of all intersections. We have implemented a collection strategy where the convex hull is built from all intersections of an individual guard traversal.

The following observations were made on an implementation of Algorithm 3.

1. The algorithm provides a new degree of exactness. The sources of inexactness are the computation of the matrix exponential $\mathrm{e}^{\delta A}$, over-approximations due to the initial bloating procedure, possible artifacts due to the invariant (Line 6), and 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 (Line 2 and Line 4). 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.
    d) The Minkowski sum in Line 6 has the highest influence: A non-trivial set $\mathbf{V}$ ($\mathbf{V} \neq \{\mathbf{0}\}$) leads to a linear growth of the representation matrices.

## 5.4. Le Guernic and Girard's Reachability Algorithm (Algorithm 4)

Compared in run-time, our prototypical implementation of Algorithm 3 is clearly behind the reachability algorithm of Le Guernic and Girard (2009). Algorithm 4 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. The template polyhedron $\mathbf{P}(T_0, \mathbf{b}_{k+1})$ is an over-approximation of the current flow segment and its computation can be done in constant time.[5] The quality of the over-approximation highly depends on the template

[5] This is a more 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

**Algorithm 4** Reachability Algorithm for a Linear System (LGG)

**Input:** $\delta$, $A$, $I$, $\mathcal{G}$, $\mathbf{R}_0$, $\mathbf{V}$, $N$ as specified in Algorithm 3 and an additional template polyhedron $\mathbf{P}(T_0, \mathbf{b}_0)$ over-approximating $\mathbf{R}_0$.

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

```
1:  $\mathbf{s}_0 \leftarrow \mathbf{0}$;
2:  for $k \leftarrow 0, \dots, N$ 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:      $T_{k+1} \leftarrow T_k e^{\delta A}$;
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;
```

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 (Frehse et al., 2011).

## 5.5. Comparison of Algorithm 3 and Algorithm 4

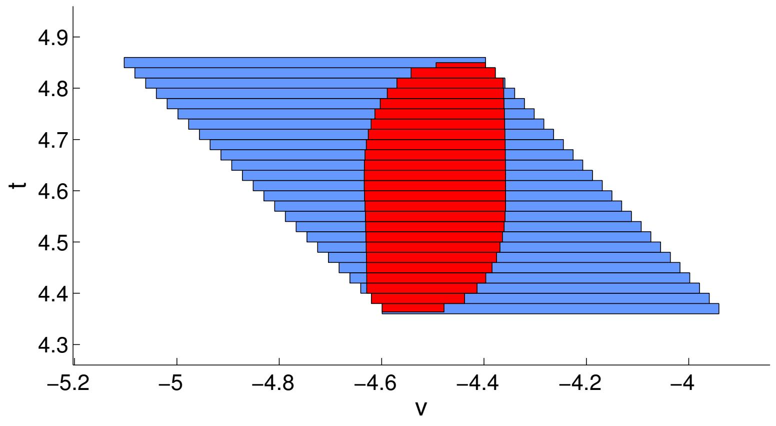


Figure 5.2.: Comparison of Algorithm 3 and 4

Figure 5.2 shows the first intersection of a bouncing ball with the guard (the floor).

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, see Schrijver (1986).

The model description can be found in Section 6.1.1. We use the time step $\delta = 0.02$. The outer slices show the intersections computed by Algorithm 4 using a rectangular template matrix. Each inner slice shows a tight rectangular over-approximation of the sops computed by Algorithm 3. Their representation matrices reach a size of about 3500 rows and 2000 columns with 9000 non-zero coefficients. The convex hull of all intersections has a size of 82652 rows, 46999 columns and 283006 non-zero coefficients.

Due to the proper handling of guard intersections and influences of invariants, the sop-based reachable set computation according to Algorithm 3 provides tighter over-approximations than the support function-based reachable set computation according to Algorithm 4. The increased precision comes along with a considerable growth of the representation size of the sops.

## 5.6. Combining Algorithm 3 and Algorithm 4 (Algorithm 5)

**Algorithm 5** Reachability Algorithm for a Linear System (SOP + LGG)

**Input:** $A$, $I$, $\mathcal{G}$, $\mathbf{R}_0$, $\mathbf{V}$, $N$ as specified in Algorithm 3 and an additional template polyhedron $\mathbf{P}(T_0, \mathbf{b}_0)$ over-approximating $\mathbf{R}_0$.

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

```
 1: $\mathbf{s}_0 \leftarrow \mathbf{0}$;
 2: for $k \leftarrow 0, \dots, N$ 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:   $T_{k+1} \leftarrow T_k e^{\delta A}$;
 9:   $\mathbf{b}_{k+1} \leftarrow \mathbf{h}_{\mathbf{R}_0}(T_{k+1}) + \mathbf{s}_{k+1}$;
10:   $\mathbf{R}_{k+1} \leftarrow e^{\delta A}\mathbf{R}_k + \mathbf{V}$;
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;
```

Algorithm 5 is a combination of Algorithm 3 and Algorithm 4. 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 Line 12 can be done in constant time. Lines 10–13 are an equivalent replacement for the assignment $\mathbf{R}_{k+1} \leftarrow (e^{\delta A}\mathbf{R}_k + \mathbf{V}) \cap \mathbf{I}$ with an additional redundancy removal, see also Item 2b in Section 5.3.

### 5.6.1. Fighting the Monotonic Growth by Interpolation

In practice we observe a similar behavior to Algorithm 4: The step computation of Algorithm 5 is done in constant time since the computational burden is shifted to the LGG-part. But the size of the sops still grows monotonically. While this growth can be handled during the collection and following discrete updates, latest in the next continuous iteration, when the discrete postimage of a symbolic state is passed to Algorithm 5 again, the enormous size of the sop has an effect: All involved linear programs of Algorithm 5 have to be solved over systems of linear inequalities of enormous size.[6] To overcome this problem, we use the ray shooting-based interpolation as described in Section 3.12. 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 may 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 4. Replacing $\mathbf{R}_k$ by the interpolating polyhedron still yields results which are at least as good as the results we would achieve by the pure LGG-algorithm. In our prototype SoapBox[7], 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.

### 5.6.2. The Impact of Empty Intersections

It may happen that some collected intersections $\mathbf{R}_k \cap \mathbf{G}_j$ are empty since we only check the feasibility of the over-approximation $\mathbf{P}(T_0, \mathbf{b}_K) \cap \mathbf{I} \cap \mathbf{G}_j$. Joining these intersections using the convex hull operation could lead to huge over-approximations, see the closed convex hull construction in Proposition 3.40 on page 36. To avoid this, we have to perform a feasibility check for each of the intersections, or we have to ensure that the normal cones of the sets agree. If the input sets $\mathbf{R}_0$ and $\mathbf{V}$ are bounded and no redundancy removal has been applied, then the normal cone of each of the intersections $\mathbf{R}_k \cap \mathbf{G}_j$ is equal to the state space, even if the $\mathbf{R}_k \cap \mathbf{G}_j$ is empty. Hence, an additional feasibility check for the intersections is not necessarily needed. Since our interpolation strategy builds the convex hull of at most `max_conv_hull` consecutive elements and interpolation reveals the emptiness of the interpolated sops, it could be interesting to experiment with delayed or disabled feasibility checks during the assembly of the intersections.

[6] Actually, the enormous size of the sops already effects the initial bloating procedure.
[7] SoapBox is available under https://vhome.offis.de/willemh/soapbox/

# 6. SoapBox and Experimental Results

The reachability algorithms for linear systems, Algorithm 3, Algorithm 4, and Algorithm 5, together with the embracing reachability algorithm for hybrid systems, Algorithm 2, have been implemented in our prototype SoapBox. In this chapter we outline some additional features of SoapBox and provide experimental results.

SoapBox was originally implemented in Matlab using the Matlab-interface of Gurobi Optimizer 5.6[1] for the linear programming tasks. Currently, the data-structures and algorithms of SoapBox are re-implemented in C++ with the aim to achieve a tighter interface to Gurobi and to provide an efficient library supporting various operations on sops, including the reachable set computation for linear systems. SoapBox has been successfully applied in the case study Damm et al. (2014a,b).

Parts of the given results have been published as Hagemann (2014a). Since then, the benchmark results for SoapBox have substantially improved due to the partial C++-implementation. The current results are reported below.

## 6.1. Experimental Results

To assess the differences between the combined Algorithm 5 (SOP) and the purely support function-based Algorithm 4 (LGG) we consider different examples. Since SpaceEx likewise implements the LGG-algorithm, we also compare our prototypical implementation against the productive implementation in SpaceEx, where we used the SpaceEx Virtual Machine Server v0.9.8b for the comparison. We turn our attention to the reachable set computation and provide graphical presentations of the computed reachable states. All computations were bounded in several respects:

(i) We restricted the number of continuous steps in each flow computation (`local time horizon` in SpaceEx). Due to the nature of our experiments, we were able to choose the bounds sufficiently large such that all continuous flow computation terminated correctly before exceeding the given bound.[2]

(ii) We restricted the number of iterations (`max. iterations` in SpaceEx), which is the maximal number of cycles of Algorithm 2. The bound was chosen sufficiently large such that the computation terminated correctly.

[1] http://www.gurobi.com

[2] Both tools, SoapBox and SpaceEx, display warnings if the continuous flow computation exceeds the given bound. Hence, we are able to verify this claim by inspecting the output.

(iii) The AVC experiment and the colliding pendula experiment were restricted by explicitly modeling the time $t$ with initial value $t = 0$ and invariant $t \leq T$, where $T$ is the bound on simulation time.

If not stated otherwise, we used a rectangular template matrix (`box`-option in SpaceEx) for the computations. We solely used ray shooting-based interpolation in Algorithm 5.

### 6.1.1. Bouncing Ball

For our benchmarks we have chosen a simple model of a bouncing ball. The variable $x$ represents the current height of the ball over the floor, which is located at $\mathbf{x} = 0$, and $v$ represents the current velocity of the 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 $t = 0$.

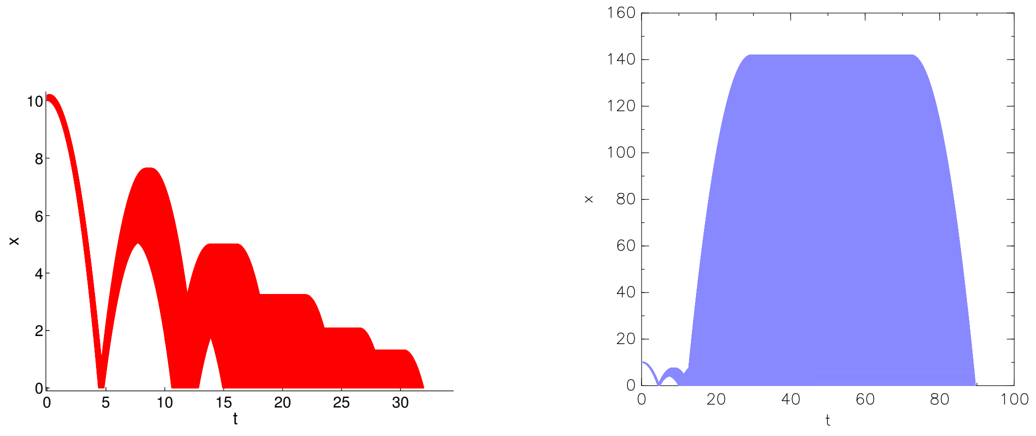


Figure 6.1.: Comparison of Algorithm 5 and SpaceEx

Throughout all computations the sop-specific configuration parameters were set as follows: `interpolate_after` $= 20$ and `max_conv_hull` $= 4$. Figure 6.1 shows the reachable positions $x$ over time $t$ for 6 iterations. The left hand side diagram shows the reachable states computed by Algorithm 5 and the right hand side diagram shows the reachable states computed by the LGG-algorithm[3]. Notice the different scalings of both figures. We observe a precise reproduction of the reachable states by Algorithm 5, while the reachable states computed by Algorithm 4 only describe a coarse over-approximation. Actually, the reachable states computed by Algorithm 5 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.

[3] The figure actually shows the SpaceEx output. The output of Algorithm 4 looks quite the same, but we think it is more impressive to compare with SpaceEx here.

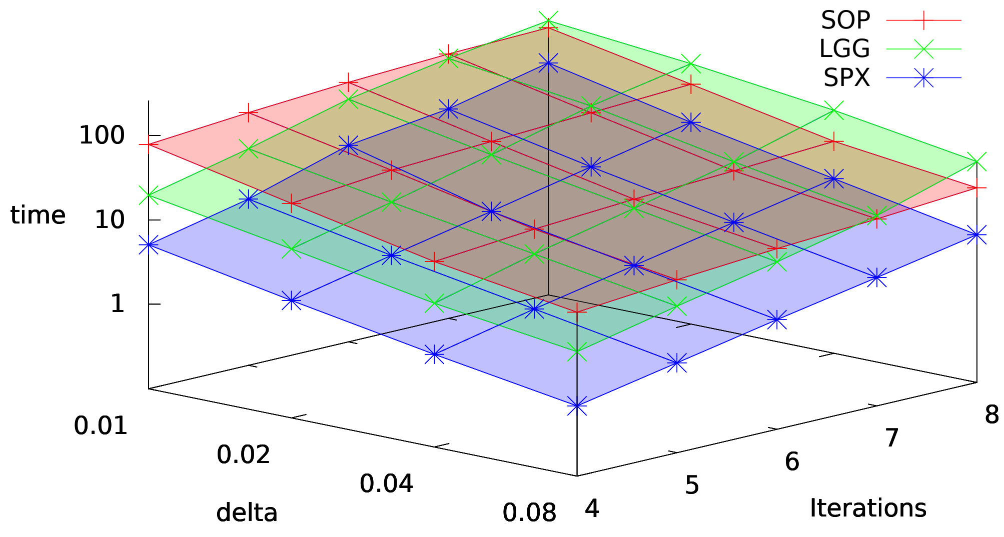


Figure 6.2.: Run-Time Comparison of Algorithm 5 (SOP), Algorithm 4 (LGG), and SpaceEx (SPX)

The computational effort due to the additional flow-segment computations is also reflected in the benchmarks. Figure 6.2 shows the run-times in seconds for different time steps $\delta$ and different numbers of iterations. Note that the time-axis and the $\delta$-axis use a logarithmic scale. We make the following observations:

- The run-times are located on nearly flat surfaces in the logarithmic scale, indicating a relationship of the form $\log(t) = -\alpha \log(\delta) + \beta i + \gamma$ between the run-time $t$, the step-width $\delta$ and the number of iterations $i$. The parameters $\alpha$, $\beta$ and $\gamma$ depend on the chosen algorithm. Hence, for all three algorithms we obtain a relationship of the form $t = a\delta^{-\alpha}b^i$ where $a$,$b$, and $\alpha$ are parameters specific to the algorithm.

  For a fixed number of iterations we obtain a relationship of the form $t = a'\delta^{-\alpha}$, and a relationship of the form $t = a''b^i$ for a fixed choice of $\delta$. While the latter relationship is clearly a model-specific property, we speculate that the relationship $t = a'\delta^{-\alpha}$ is a property specific to the algorithms.

- The run-times of our implementation of the LGG-algorithm and of SpaceEx are located on nearly parallel surfaces. The run-times differ by the factor 4.66 with standard deviation of 0.97. Since most of the computational time is spent in the continuous step computation, this indicates that our implementation is a faithful re-implementation of the LGG-algorithm.

- A comparison of our implementation of the LGG-algorithm 4 and the combined Algorithm 5 shows that Algorithm 5 outperforms Algorithm 4 despite the computational overhead for an increasing number of iterations, which can be explained by the coarse over-approximation of the reachable states by the LGG-algorithm.

## 6.1.2. Colliding Pendula

In order to assess the precision of SoapBox, we have chosen a simple model of two colliding pendula. Both pendula are described by mass points $m_1 = 1.0$ and $m_2 = 1.4$ that are fixed on the same anchor point with rods of length $l = 1$. $\phi_1$ and $\phi_2$ are the angular displacements of the respective pendulum with derivatives $\dot{\phi}_1 = \omega_1$ and $\dot{\phi}_2 = \omega_2$. We assume that always $\phi_2 \leq \phi_1$ holds. The bobs collide by the law of elastic collision if $\phi_1 \leq \phi_2$ and $\omega_1 < \omega_2$,[4] see Figure 6.3.

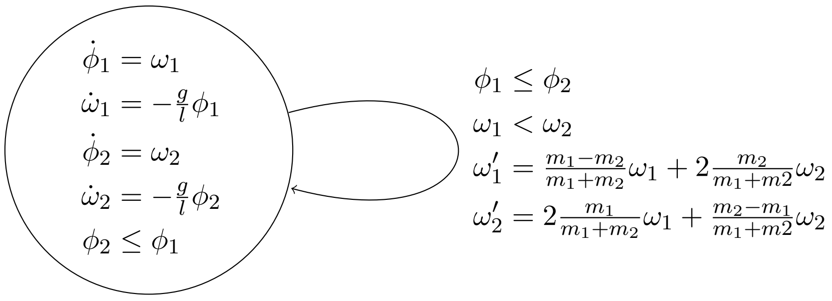

$\dot{\phi}_1 = \omega_1$
$\dot{\omega}_1 = -\frac{g}{l}\phi_1$
$\dot{\phi}_2 = \omega_2$
$\dot{\omega}_2 = -\frac{g}{l}\phi_2$
$\phi_2 \leq \phi_1$
$\phi_1 \leq \phi_2$
$\omega_1 < \omega_2$
$\omega_1' = \frac{m_1-m_2}{m_1+m_2}\omega_1 + 2\frac{m_2}{m_1+m2}\omega_2$
$\omega_2' = 2\frac{m_1}{m_1+m_2}\omega_1 + \frac{m_2-m_1}{m_1+m2}\omega_2$

Figure 6.3.: Colliding Pendula

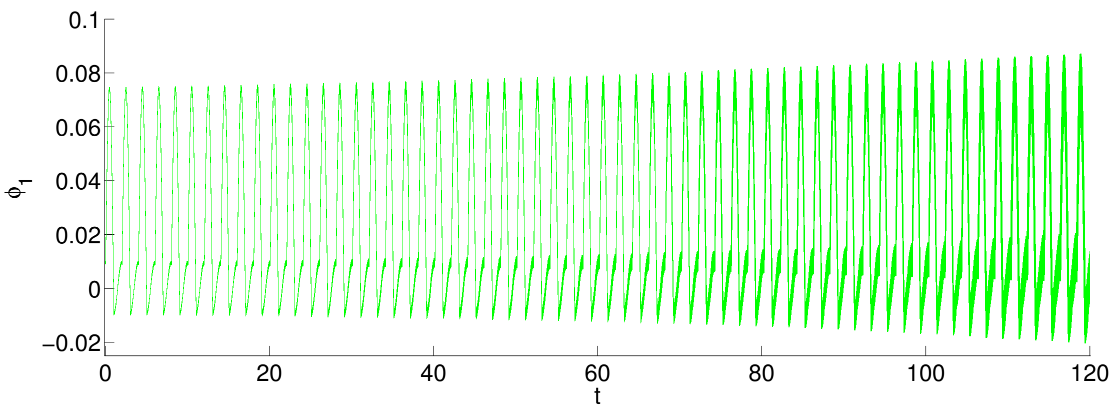


Figure 6.4.: Reachable States of Colliding Pendula (SoapBox)

We are interested in the reachable states, where we have chosen the initial values $0.009 \leq \phi \leq 0.01$, $\omega_1 = 0$, $\phi_2 = 0$, and $0.199 \leq \omega_2 \leq 0.2$. Figure 6.4 shows the

[4] See the following section for the usage of strict inequalities.

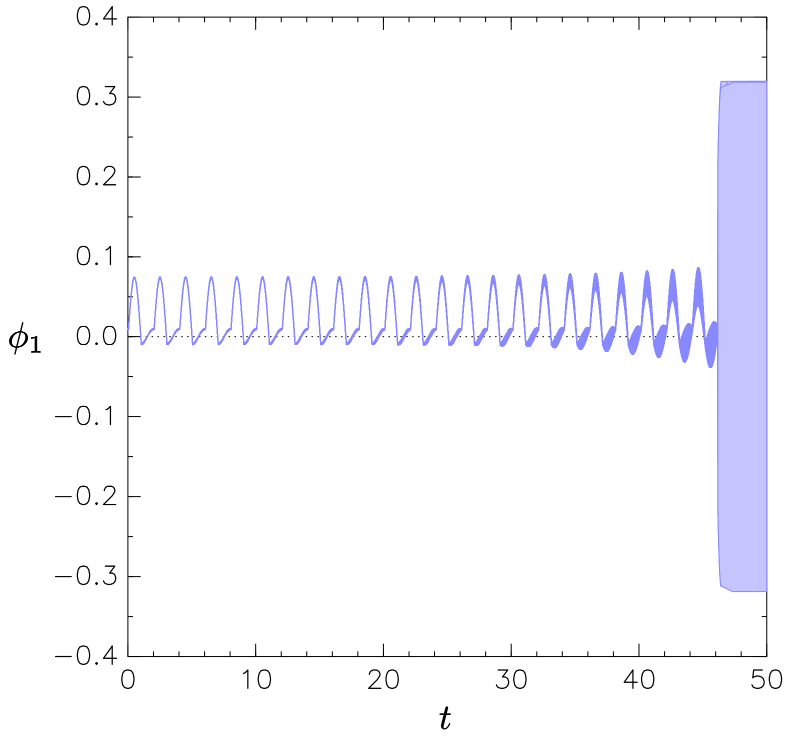


Figure 6.5.: Reachable States of Colliding Pendula (SpaceEx)

output of SoapBox for the simulation time $0 \leq t \leq 120$. The parameter were set as follows: $\delta = 0.01$, `max_conv_hull` $= 4$ and `interpolate_after` $= 20$. Figure 6.5 shows the reachable states computed by SpaceEx, where we have chosen the STC-scenario, an improved LGG-algorithm. Here, only the output of the first 51 iterations is shown, which suffices to see the catastrophic effects of the weak handling of invariants and guards in a scenario that is mainly based on the usage of support functions.

### 6.1.3. Approach Velocity Controller

The AVC controls the velocity $v$ of a following car in order to establish the desired distance $d_{\text{des}}$ to the leading car, which has velocity $v_a$. The current distance of the cars is given by the variable $d$. The dynamics are

$$\begin{aligned} \dot{d} &= v_a - v, & \dot{v} &= 0.29(v_a - v) + 0.01(d - d_{\text{des}}), & \dot{t} &= 1, \\ -0.5 &\leq \dot{v}_a \leq 0.5, & 0 &\leq v_a \leq 20. \end{aligned} \tag{6.1}$$

The inequalities (6.1) 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.

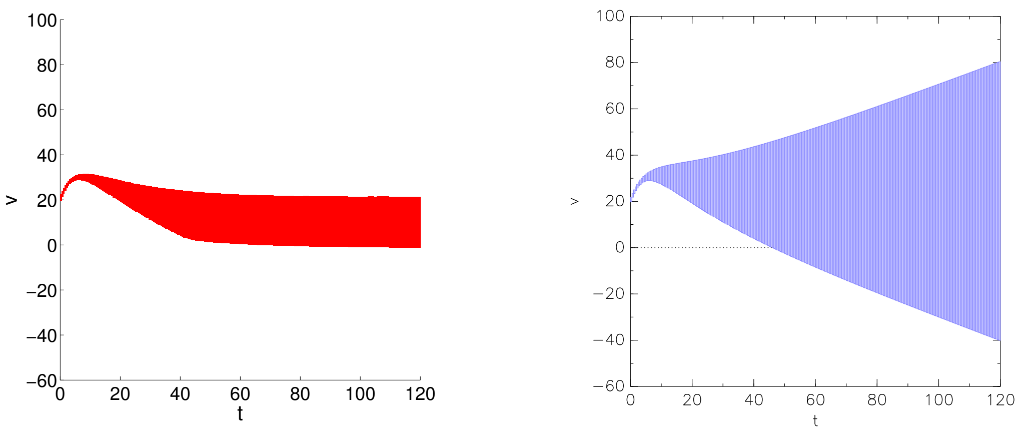
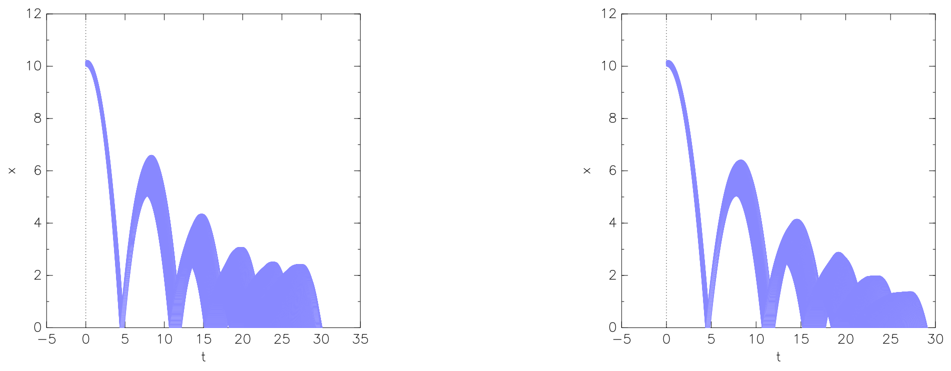


Figure 6.6.: Comparison of Algorithm 5 and SpaceEx

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 an interval. Figure 6.6 shows the reachable states computed by Algorithm 5 on the left and the reachable states computed by SpaceEx on the right. A comparison of both figures shows that SpaceEx is not able to establish any bound on the velocity of the follower while Algorithm 5 shows the desired behavior. For both algorithm we used the time step parameter $\delta = 0.5$, and for Algorithm 5 we used `interpolate_after` $= 40$.



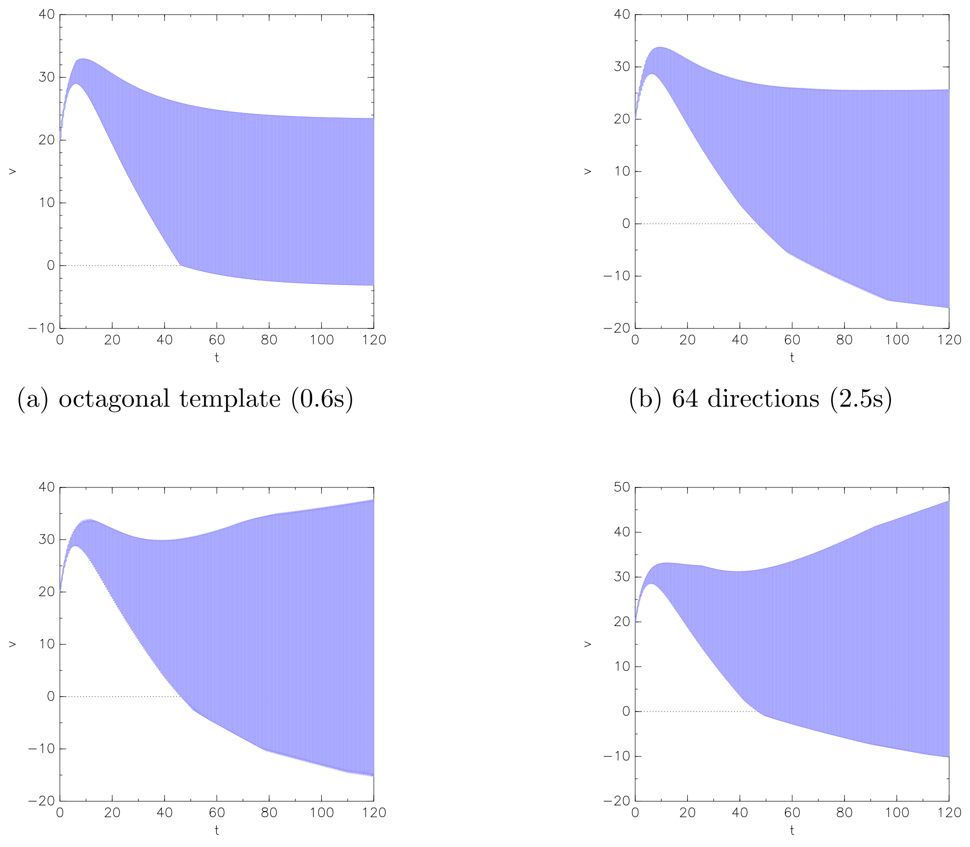
(a) 32 directions (52.1s)

(b) 64 directions (147.8s)

Figure 6.7.: Bouncing Ball revisited, SpaceEx (LGG)

### 6.1.4. Role of Template Directions

We explicitly note that, by increasing the number of directions, the LGG-algorithm computes much better over-approximations than it has been shown in Figure 6.1 on the right-hand side. Indeed, Figure 6.7 shows the reachable states computed by SpaceEx using 32 and 64 uniformly distributed directions.[5] Increasing the number of directions has also an impact on the run-times. While the computation took 15 seconds for the box template, the run-time increased to 52 seconds for 32 directions and increased to 148 seconds for 64 directions (SpaceEx Virtual Machine Server v0.9.8c).



(a) octagonal template (0.6s)

(b) 64 directions (2.5s)

(c) 128 directions (7.5s)

(d) 256 directions (19.7s)

Figure 6.8.: AVC revisited, SpaceEx (LGG)

Roughly, one should expect that increasing the number of template directions also increases the quality of the LGG-algorithm. Figure 6.8 shows that this idea might be misleading. While the octagonal template (corresponding to 32 directions) yields a good over-approximation of the reachable states, an increased number of directions does not

[5] For all those beady-eyed readers who compare Figure 6.7 with the SoapBox output in Figure 6.1 and wonder whether SpaceEx eventually computes a more precise over-approximation than SoapBox, please note that in Figure 6.1 we used a rectangular template for the output while in Figure 6.7 all 32 and 64 directions were used for the output. This is an issue of the output, not of the actual computed state sets.

necessarily increase the precision of the over-approximation. Interestingly, the octagonal template provides good results. So far, it is not clear why this template fits the dynamic of the example so well. Possibly, the reason could be buried in the fact that octagonal templates allow to assess linear inequalities relation of the form $v_a - v$, a form of linear expression which also occurs in the linear differential equation of the model.

## 6.2. Additional Features of SoapBox

### 6.2.1. Support for Safe Sets

In addition to the designated state sets *Init* and *Unsafe*, SoapBox supports *safe* sets. A safe set *Safe* is a set of states for which the evolution is already known or not of interest. Hence, we may stop reachability analysis for trajectories which have entered *Safe*. For example, *Safe* could include all initial states for which it is known that none of its trajectories will ever reach *Unsafe*. Note, that any subset of a safe set is again a safe set.

*Safe* has to be treated differently to guard sets. In order to prove that all trajectories emanating from the initial states have entered a safe set, it suffices to show that the current flow segment is a subset of *Safe*. While there is no known efficient method to decide the subset relation between two sops in general, the special case where *Safe* is given as an $\mathcal{H}$-polyhedron is efficiently decidable for $\mathcal{H}$-polyhedra, see Section 3.14.

The support for safe sets has been introduced in the context of the case study Damm et al. (2014a,b), where safe sets were computed by Stabhyli, a tool for stability verification of non-linear hybrid systems (Möhlmann and Theel, 2013), as follows: Stabhyli generates a mode specific Lyapunov function $V(\cdot)$ which assigns a value to any state of the mode. A *level set* for a level $l \geq 0$ is the set $\mathcal{L}_{V,l} = \{\mathbf{x} \,|\, V(\mathbf{x}) \leq s\}$. Since the values of a Lyapunov function along any trajectory do not increase, any purely continuous trajectories starting in a level set will not leave the level set. However, the trajectory could still enter *Unsafe* or leave the current mode due to a discrete transition if the trajectory hits its guard. Now, let $s$ be a value which is less than the value of the Lyapunov function of any state in *Unsafe* or any state in a mode specific discrete transition guard. Then $\mathcal{L}_{V,s}$ is a mode specific safe set, and any polyhedral subset of $\mathcal{L}_{V,s}$ can be used as a safe set for SoapBox.

### 6.2.2. Support for Guards with Strict Inequalities

We have added a proper handling of strict inequalities to SoapBox: A transition involving a strict guard $\mathbf{n}^T\mathbf{x} < c$ is disabled as long as the current flow segment $\mathbf{P}$ does not contain a witness point $\mathbf{x}$ that satisfies the strict guard. Otherwise, the transition is enabled and treated like a non-strict inequality, which results in a closed over-approximation of the actual intersection. The support for strict inequalities is useful for hybridization of modes whose dynamics are given by non-linear differential equations (Damm et al., 2014a).

# 7. Discussion and Outlook

## 7.1. Discussion

### 7.1.1. High Dimensional Reachability Analysis

The presented models have only a few variables. What do we have to expect for systems with more variables, say, hundreds of variables? First of all, the idea of the symbolic orthogonal projections is to introduce additional variables. Dealing with hundreds of variables is a primary feature of sops. The sops evolving during the reachability computations are indeed high dimensional. But we have learned that, after some point, the representation size becomes too large to be handled efficiently by linear programming. We introduced the notion of interpolation which allows us to shrink the representation size. Some experiments indicate that the presented heuristic for the interpolation might be a problem in higher dimensions since interpolation generates too many new facets, each accompanied by a large linear program. Possible resorts are:

- use interpolation after fewer steps,
- restrict the number of generated facets in the interpolation,
- generate no additional facets at all, instead fall back to the over-approximating template polyhedra of the sop. This method still respects the influence of the invariant.

Note that each of the suggested method still produces more precise results than the LGG-algorithm.

### 7.1.2. Backward Reachability

I mainly discussed forward reachability analysis by computing postimages. Nevertheless, several aspects of backward reachability analysis had been discussed at times. The presented postimage operations may either easily be reversed or a corresponding preimage operation was stated explicitly. Hence, the reader should be in the position to compile algorithms for a backward reachability analysis with little effort.

### 7.1.3. Numerical Issues

I should not conceal that my approach is subject to several numerical issues. First of all, the computation of the matrix exponential is an issue. I already addressed this problem in footnote 2 on page 65.

Due to the usage of floats, I encountered several other numerical issues. Since basic operations on sops consist of concatenations of matrices only and, hence, are unproblematic from a numerical point of view, numerical issues emerge mainly from the solver for the linear programs. During an early stage of the development of SoapBox, I used GLPK[1] as solver for the linear programming tasks. A major problem was the re-usage of the optimal solution of a linear program as input for another linear program. If some coefficients of the optimal solutions were sufficiently small, say, less than $1e^{-15}$, this led to catastrophic results. The obvious, but certainly improper, resort was to substitute small values by 0. The situation improved substantial after changing to Gurobi.

SoapBox uses the decomposition method to decompose the affine transformations of the discrete updates. The decomposed transformation matrices are computed a priori using a build-in Matlab algorithm which produces the row echelon form. So far, I did not encounter numerical problems with this approach, however it could be interesting to use the direct method as given in Proposition 3.13.

Aside from a few constants, which describe the lower bound from where on we regard a number as positive, the current version of SoapBox does not use any workaround for numerical issues, which results in a clean implementation.

## 7.2. Outlook

There are several starting point for further research related to the theory of sops. Some research is of interest per se, while other research is motivated by application.

- The block structure of sops appeals to investigate decomposition algorithms in the domain of linear programming like the Dantzig-Wolfe decomposition.

- It would also be interesting to extend the theory of sops such that it can cope with strict inequalities. I have already extended the theory of sops by a proper handling of equalities, at least for those operations which are implemented in SoapBox. This extension has been quite straightforward. Linear programming tasks over sops with strict inequalities can be handled using the fact that a system $A_1\mathbf{x}+L_1\mathbf{z} \leq \mathbf{a}_1$, $A_2\mathbf{x} + L_2\mathbf{z} < \mathbf{a}_2$ of strict and non-strict linear inequalities is feasible if and only if the linear program

  $$\text{maximize } \lambda \text{ subject to } A_1\mathbf{x} + L_1\mathbf{z} \leq \mathbf{a}_1,\ A_2\mathbf{x} + L_2\mathbf{z} + \mathbf{1}\lambda \leq \mathbf{a}_2,\ 0 \leq \lambda \leq 1$$

  has a positive optimal value $\lambda > 0$.

- It would be interesting to apply sops also in other areas of system analysis, especially in the area of program analysis. In my opinion there are two main problems: the lack of an efficient method to decide the subset relation of sops and the lack of a widening (or extrapolation) operator for sops. While there is little hope to find an efficient decision procedure for the subset relation of sops in general, it could

[1] https://www.gnu.org/software/glpk/

be worth to inspect the relationship of the structure of sops before and after application of a geometrical operation to find at least a decision procedure for some special cases. For the second problem there is also little hope to find an equivalent sop-based formulation of the standard widening operator since this widening operator is bound too much to the $\mathcal{H}$-representation of a polyhedron[2]. Nevertheless, it could be worth to look for some extrapolation operators, for example the operation that takes two polyhedra $\mathbf{P}$ and $\mathbf{Q}$ as an input and returns a polyhedron whose facets are all those facets of the closed convex hull that have at least a common point with $\mathbf{P}$, see Henzinger et al. (2001). While it is possible to compute at least the support function of this operation for arbitrary sops, I have not yet found any way to encode this operation as a sop.

However, the decision procedure for the subset relation and the widening operator can be realized for $\mathcal{H}$-polyhedra. If we are willing to accept the over-approximations due to template polyhedra and interpolation, we could integrate sops into existing system analysis tools that are based on polyhedra.

Also in the context of reachability analysis there are a lot of starting points for further research.

- The main motivation for my research in reachability analysis is to extend the symbolic mode-checker FOMC by a continuous preimage computation for linear systems. FOMC uses LinAIGs as a non-convex intermediate representation (Damm et al., 2012). By the key work of my colleagues in the FOMC team and, last but not least, by this thesis, the aim is sufficiently achieved from a theoretical perspective. It remains to re-implement the prototypical algorithms of SoapBox into a callable library, followed by an experimental evaluation phase.

- In order to deal with non-linear dynamics we have already experimented with hybridizations and bracketing systems. In this respect it also turns out that sops can be useful for pre-analysis of hybrid systems.

- How can we cope with the potential symbolic state set explosion? Certainly by joining several sops into a single sop, preferably using the convex hull operation. Usually, this results in over-approximations of the state set. Bemporad et al. (2001) provide an exact algorithm which checks whether the union of $\mathcal{H}$-polyhedra is convex and, hence, agrees with their convex hull. It would be interesting to have such an algorithm for sops. For the imprecise case, it would be nice to have some heuristics which help us to minimize the artifacts.

[2] To be more precise, the computation of the standard widening operator $\mathbf{P} \triangledown \mathbf{Q}$ is mainly bound to the $\mathcal{H}$-representation of the left argument. Hence, it could be worth to investigate whether it can be computed when the right argument is a sop.

# 8. Conclusion

I introduced a novel representation class for polyhedra, the symbolic orthogonal projections (sops), and provided a first insight into the theory of sops. Various geometrical operations can be performed efficiently and exactly on sops. Together with linear programming, sops provide a powerful framework for system analysis as it had been typified for the reachability analysis of linear hybrid systems.

I presented a reachability algorithm where all polyhedral operations are performed exactly (Algorithm 3). Due to the monotonic growth of the representation size, this algorithm is not suitable for practical applications. After combining Algorithm 3 with the LGG-algorithm, we achieved an efficient reachability algorithm (Algorithm 5). In order to show the applicability, accuracy, and efficiency of the resulting algorithm, I implemented the algorithms in a tool called SoapBox and provided experimental results.

The experiments indicate that Algorithm 5 allows a precise reachable set computation. Especially, if the invariant plays an important role, it is possible to discover the proper behavior of a system, where support function-based methods fail (AVC experiment). In the bouncing ball experiment and the colliding pendula experiment, we have seen that support function-based methods initially provide good insight into the behavior of a system, but suddenly end up in catastrophic over-approximations after certain iterations, while Algorithm 5 still provide accurate results. Moreover, a more precise reachable set computation can have positive effects on the run-time of the algorithm. Actually, the benchmarks of the bouncing ball experiment indicate that the inherently slower but precise Algorithm 5 overtakes the LGG-algorithm for a sufficiently high number of iterations.

# Bibliography

Matthias Althoff, Olaf Stursberg, and Martin Buss. Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. *Nonlinear Analysis: Hybrid Systems*, 4(2):233–249, 2010.

Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer, 1993.

Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A. Henzinger, Pei-Hsin Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

Rajeev Alur, Thomas A. Henzinger, and Pei-Hsin Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22(3):181–201, 1996.

Roberto Bagnara, Elisa Ricci, Enea Zaffanella, and Patricia M. Hill. Possibly not closed convex polyhedra and the Parma Polyhedra Library. In Manuel V. Hermenegildo and Germán Puebla, editors, *Static Analysis*, volume 2477 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2002.

Cedric Bastoul. Code generation in the polyhedral model is easier than you think. In *Proceedings of the 13$^{th}$ International Conference on Parallel Architectures and Compilation Techniques*, pages 7–16. IEEE Computer Society, 2004.

Alberto Bemporad, Komei Fukuda, and Fabio D. Torrisi. Convexity recognition of the union of polyhedra. *Computational Geometry*, 18(3):141–154, 2001.

Florence Benoy, Andy King, and Fred Mesnard. Computing convex hulls with a linear solver. *Theory and Practice of Logic Programming*, 5(1-2):259–271, 2005.

Endre Boros, Khaled Elbassioni, Vladimir Gurvich, and Hans Raj Tiwary. The negative cycles polyhedron and hardness of checking some polyhedral properties. *Annals of Operations Research*, 188(1):63–76, 2011.

Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification*, volume 8044 of *Lecture Notes in Computer Science*, pages 258–263. Springer, Heidelberg, 2013.

Alongkrit Chutinan. *Hybrid system verification using discrete model approximations.* PhD thesis, Carnegie Mellon University, 1999.

Alongkrit Chutinan and Bruce H. Krogh. Computing polyhedral approximations to flow pipes for dynamic systems. In *Proceedings of the 37th IEEE Conference on Decision and Control, 1998*, volume 2, pages 2089–2094, Dec 1998.

Patrick Cousot and Nicolas Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Proceedings of the 5th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 84–96. ACM, 1978.

Werner Damm, Henning Dierks, Stefan Disch, Willem Hagemann, Florian Pigorsch, Christoph Scholl, Uwe Waldmann, and Boris Wirtz. Exact and fully symbolic verification of linear hybrid automata with large discrete state spaces. *Science of Computer Programming*, 77(10-11):1122–1150, 2012.

Werner Damm, Willem Hagemann, Eike Möhlmann, and Astrid Rakow. Component based design of hybrid systems: A case study on concurrency and coupling. Technical Report 95, SFB/TR 14 AVACS, 2014a. ISSN: 1860-9821, `http://www.avacs.org`.

Werner Damm, Eike Möhlmann, and Astrid Rakow. 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 '14, pages 145–150, New York, 2014b. ACM.

George B. Dantzig. Linear programming: The story about how it began. *History of Mathematical Programming*, pages 19–31, 1991.

Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable verification of hybrid systems. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Computer Aided Verification*, volume 6806 of *Lecture Notes in Computer Science*, pages 379–395. Springer, Heidelberg, 2011.

Komei Fukuda. Lecture: Polyhedral computation, spring 2011, 2011. URL `http://stat.ethz.ch/ifor/teaching/lectures/poly_comp_ss11/lecture_notes`.

Antoine Girard. Reachability of uncertain linear systems using zonotopes. In Manfred Morari and Lothar Thiele, editors, *Hybrid Systems: Computation and Control*, volume 3414 of *Lecture Notes in Computer Science*, pages 291–305. Springer, Heidelberg, 2005.

Willem Hagemann. Reachability analysis of hybrid systems using symbolic orthogonal projections. In Armin Biere and Roderick Bloem, editors, *Computer Aided Verification*, volume 8559 of *Lecture Notes in Computer Science*, pages 406–422. Springer International Publishing Switzerland, 2014a.

Willem Hagemann. Reachability analysis of hybrid systems using symbolic orthogonal projections. Technical Report 98, SFB/TR 14 AVACS, 2014b. ISSN: 1860-9821, `http://www.avacs.org`.

Thomas L. Heath, editor. *The Thirteen Books of Euclid's Elements.* Dover Publication, New York, 1956.

Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 373–382. ACM, 1995.

Thomas A. Henzinger, Jörg Preußig, and Howard Wong-Toi. Some lessons from the HyTech experience. In *Proceedings of the 40th IEEE Conference on Decision and Control, 2001*, volume 3, pages 2887–2892. IEEE, 2001.

Harro Heuser. *Gewöhnliche Differentialgleichungen: Einführung in Lehre und Gebrauch.* B.G. Teubner, 1995.

Leonid Khachiyan, Endre Boros, Konrad Borys, Khaled Elbassioni, and Vladimir Gurvich. Generating all vertices of a polyhedron is hard. *Discrete & Computational Geometry*, 39(1-3):174–190, 2008.

Jean-Louis Lassez. Querying constraints. In *Proceedings of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 288–298. ACM, 1990.

Colas Le Guernic. *Reachability analysis of hybrid systems with linear continuous dynamics.* PhD thesis, Université Grenoble 1 - Joseph Fourier, 2009.

Colas Le Guernic and Antoine Girard. Reachability analysis of hybrid systems using support functions. In Ahmed Bouajjani and Oded Maler, editors, *Computer Aided Verification*, volume 5643 of *Lecture Notes in Computer Science*, pages 540–554. Springer, Heidelberg, 2009.

Jiří Matoušek and Bernd Gärtner. *Understanding and Using Linear Programming.* Universitext. Springer, 2007.

Eike Möhlmann and Oliver Theel. Stabhyli: A tool for automatic stability verification of non-linear hybrid systems. In *Proceedings of the 16th international conference on Hybrid systems: Computation and Control*, pages 107–112. ACM, 2013.

Cleve Moler and Charles Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49, 2003.

Theodore S. Motzkin, Howard Raiffa, Gerald L. Thompson, and Robert M. Thrall. The double description method. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games*, volume 2, pages 51–73. Princeton University Press, 1953.

R. Tyrrell Rockafellar and Roger J.-B. Wets. *Variational Analysis*, volume 317 of *Grundlehren der mathematischen Wissenschaften.* Springer, 1998.

Sriram Sankaranarayanan, Michael A. Colón, Henny Sipma, and Zohar Manna. Efficient strongly relational polyhedral analysis. In E. Allen Emerson and Kedar S. Namjoshi, editors, *Verification, Model Checking, and Abstract Interpretation*, volume 3855 of *Lecture Notes in Computer Science*, pages 111–125. Springer, Heidelberg, 2006.

Sriram Sankaranarayanan, Thao Dang, and Franjo Ivančić. Symbolic model checking of hybrid systems using template polyhedra. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 4963 of *Lecture Notes in Computer Science*, pages 188–202. Springer, Heidelberg, 2008.

Alexander Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, 1986.

Hans Raj Tiwary. On the hardness of computing intersection, union and Minkowski sum of polytopes. *Discrete & Computational Geometry*, 40(3):469–479, 2008.

Günter M. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics.* Springer, 1995.