Hybrid Tools for Hybrid Systems – Proving Stability and Safety at Once

by
Willem Hagemann, Eike Möhlmann and Oliver Theel

AVACS Technical Report No. 108
July 2015
ISSN: 1860-9821

Publisher: Sonderforschungsbereich/Transregio 14 AVACS
(Automatic Verification and Analysis of Complex Systems)
Editors: Bernd Becker, Werner Damm, Bernd Finkbeiner, Martin Fränzle,
Ernst-Rüdiger Olderog, Andreas Podelski
ATRs (AVACS Technical Reports) are freely downloadable from `www.avacs.org`

Hybrid Tools for Hybrid Systems – Proving Stability and Safety at Once

Extended Version

Willem Hagemann, Eike Möhlmann, Oliver Theel
Carl von Ossietzky University of Oldenburg
Department of Computer Science
26111 Oldenburg, Germany

{willem.hagemann,eike.moehlmann,theel}@informatik.uni-oldenburg.de

August 3, 2015

Industrial applications usually require safety and stability properties. The safety property guarantees that "something bad" never happens, and the stability property guarantees that "something good" eventually happens. The analyses of both properties are usually performed in isolation. In this work, we consider analyzing both properties by a single automatic approach for hybrid systems. We basically merge analyses of both properties to exploit the knowledge gained from the analysis of each of them in the analysis of the other. We show how both analyses can be divided into multiple steps and interlocked such that both benefit from each other. In fact, we compute single-mode Lyapunov functions, unroll the hybrid system's automaton via repeated reachability queries, and, finally, compute a global Lyapunov function. Each reachability query is simplified by exploiting the knowledge gained from the single-mode Lyapunov functions. The final computation of the global Lyapunov function is simplified by a precise characterization of the reachable states and reuses the single-mode Lyapunov functions.

We provide automated tools necessary to link the analyses and report on promising experiments we performed using our new prototype tool.

Keywords:
Hybrid Systems, Automatic Verification, Stability, Safety, Reachability, Lyapunov Theory, Geometry, Unrolling

1 Introduction

We present an approach to verify safety and stability properties of hybrid systems at once. The theory of hybrid systems provides a well-suited and natural framework for the analysis of interacting discrete and continuous behavior of a system. Examples are cyber-physical systems like vehicles or chemical processes. A hybrid system is called *safe* if some "bad states" cannot be reached, and it is called *(asymptotically) stable* if the system converges to some "good states".

Although there has been a lot of progress in recent time, verifying these types of properties is hard, especially in industrial applications where typical hybrid systems are often large. Consequently, abstractions or compositional frameworks are used to obtain simplified systems for which safety and stability can be proven and which are still powerful enough to establish the respective property of the original system. While the verification of safety and stability is usually done separately, we propose to integrate both analyses in a symbiotic fashion.

From the safety perspective, we use Lyapunov functions to detect regions that are guaranteed to be safe. Exploiting these regions helps us to shorten the reachability analysis. Indeed, if all trajectories eventually enter a safe region, then there is no need to compute infinite traces of the trajectories.

From the stability perspective, we obtain a more precise description of the feasible trajectories of a system. By discovering implicit knowledge and making it explicit, we lower the computational burden to obtain Lyapunov functions.

2 Related Work

Safety and Reachability analysis of hybrid systems has to deal with two problems: how to tackle the dynamics of the system, and how to represent the reachable states systematically. Both problems are related since the choice of the admissible dynamics has an impact on the required operations for post-image computation. For reachability analysis we will consider systems whose dynamics are given by linear differential inclusions [17, 21, 26]. Differential inclusions allows us to approximate systems with richer dynamics [3, 4, 15].

For state representation we focus on convex approaches where reachable states are usually represented by unions of convex sets. Different representations, like polyhedra [9], template polyhedra [43], zonotopes [21], ellipsoids [26], and support functions [27], are commonly used. The choice of the representation has wide influence on the approximations of the underlying sets and on the efficiency of operations required for reachability analysis. Depending on the choice of the representation, some operations may be difficult, e. g. zonotopes and support functions are challenging for intersections with guard sets [2, 22].

Recently, the support function-based tool SpaceEx [17] has become popular for reachability analysis. With respect to reachability, this paper considers the same class of hybrid automata that SpaceEx can deal with.

Stability analysis of hybrid systems has – in contrast to safety analysis – not yet received that much attention with respect to automatic proving, and, therefore, only a few tools are available. A tool by Podelski and Wagner, presented in [36], proves region stability via the search for a decreasing sequence of snapshots. Oehlerking et al. [33] implemented a powerful state space partitioning scheme for linear hybrid systems. The RSolver by Ratschan and She [42] computes Lyapunov-like functions for continuous systems. Duggirala and Mitra [16] combined Lyapunov functions with searching for a well-foundedness relation for symmetric linear hybrid systems.

Prabhakar and García [40] presented a technique for proving stability of hybrid systems with constant derivatives. YALMIP [28] and SOSTools [35] are convenient Matlab toolboxes assisting in the manual search for Lyapunov functions.

Related theoretical works are the decompositional technique by Oehlerking and Theel [34] and the work on pre-orders for reasoning about stability in a series of papers by Prabhakar et al. [38, 37, 39] whose aim is a precise characterization of the soundness of abstractions for stability properties. In contrast, our vision is an automatic computational engine for proving safety and stability. The technique and tool presented in [40] is based on abstractions. Unfortunately, their technique is restricted to hybrid systems whose differential equations have constant right hand sides while our technique is more general. However, the techniques are not mutually exclusive and have the potential to be combined.

Reach-Avoid problems are related to the investigated problem. In reach-avoid one is interested in finding a control strategy or initial condition to reach a certain set of desired states while avoiding another set of undesired states. In [30] Mitchell and Tomlin propose an exact algorithm for this problem which – like our algorithm – makes use of level sets. In [1] Abate et al. give a specification of the corresponding probabilistic problem. However, reach-avoid is an *existential problem*, while we ask *all* trajectories to converge and avoid undesired states.

3 Preliminaries

In this section we give definitions of the hybrid system model, global asymptotic stability, and Lyapunov functions. Furthermore, we briefly present two tools: 1. Stabhyli [31] which automatically computes Lyapunov functions and thereby certifies stability and 2. SoapBox [23] which automatically computes reachable state sets and, thus, allows us to check for (non-)reachable states.

Definition 1. $\mathcal{H} = (\mathcal{V}, \mathcal{M}, \mathcal{T}, \mathit{Flow}, \mathit{Inv}, \mathit{Inits})$ *is a* Hybrid Automaton *where*

- $\mathcal{V}$ *is a finite set of* variables *and* $\mathcal{S} = \mathbb{R}^{|\mathcal{V}|}$ *is the corresponding* continuous state space*,*
- $\mathcal{M}$ *is a finite set of* modes*,*
- $\mathcal{T}$ *is a finite set of* transitions (m_1, G, U, m_2) *where*
  - $m_1, m_2 \in \mathcal{M}$ *are the* source and target mode *of the transition, respectively,*
  - $G \subseteq \mathcal{S}$ *is a* guard *which restricts the valuations of the variables for which this transition can be taken,*
  - $U : \mathcal{S} \to \mathcal{S}$ *is the* update function *which might update some valuations of the variables,*
- $\mathit{Flow} : \mathcal{M} \to [\mathcal{S} \to \mathcal{P}(\mathcal{S})]$ *is the* flow function *which assigns a* flow *to every mode. A flow* $f : \mathcal{S} \to \mathcal{P}(\mathcal{S})$ *in turn assigns a closed subset of* $\mathcal{S}$ *to each* $\mathbf{x} \in \mathcal{S}$*, which can be seen as the right-hand side of a differential inclusion* $\dot{\mathbf{x}} \in f(\mathbf{x})$*,*
- $\mathit{Inv} : \mathcal{M} \to \mathcal{P}(\mathcal{S})$ *is the* invariant function *which assigns a closed subset of the continuous state space to each mode* $m \in \mathcal{M}$*, and therefore restricts valuations of the variables for which this mode can be active.*
- $(\mathit{Init}, m) \in \mathit{Inits} \subseteq \mathcal{S} \times \mathcal{M}$ *is a closed set of* initial (hybrid) states *where* m *is the* discrete state *and* Init *is the* continuous state*.*

A trajectory *of $\mathcal{H}$ is an infinite solution in form of a function $\tau(t) = (\mathbf{x}(t), m(t))$ over time t where $\mathbf{x}(\cdot)$ describes the evolution of the continuous variables and $m(\cdot)$ the corresponding evolution of the modes [32, p.35].*

3.1 Checking Stability via Stabhyli

Stabhyli can be used to obtain Lyapunov functions which certify stability of hybrid systems whose behavior is expressible in forms of polynomials. Stabhyli can be used to obtain common Lyapunov functions, piecewise Lyapunov functions and performing the (de-)compositional proof schemes presented in [12, 34]. These features are fully automatized and combined with pre- and postprocessing steps that simplify the design and counteract numerical problems. Furthermore, in case stability cannot be proven, Stabhyli returns a hint to the user. In the sequel we sketch the theoretical basis of Stabhyli.

Roughly speaking, stability is a property basically expressing that all trajectories of the system eventually reach an equilibrium point of the sub-state space and stay in that point forever. Usually, for technical reasons the equilibrium point is assumed to be the origin $\mathbf{0}$ of the continuous state space. This is not a restriction since a system can always be shifted such that the equilibrium point is $\mathbf{0}$ via a coordinate transformation. In the sequel we focus on *asymptotic stability* which does not require the equilibrium point to be reached in finite time, but only requires every trajectory to converge. This property is weaker than *exponential stability* where the existence of an exponential convergence rate is additionally required.

In the following we refer to $\mathbf{x}_{\downarrow\mathcal{V}'} \in \mathbb{R}^{|\mathcal{V}'|}$ as the sub-vector of a vector $\mathbf{x} \in \mathbb{R}^{|\mathcal{V}|}$ containing only values of variables in $\mathcal{V}' \subseteq \mathcal{V}$.

Definition 2 (Global Asymptotic Stability [32]). *Let $\mathcal{H}$ be a hybrid automaton with $\mathcal{H} = (\mathcal{V}, \mathcal{M}, \mathcal{T}, \mathit{Flow}, \mathit{Inv})$, and let $\mathcal{V}' \subseteq \mathcal{V}$ be a set of variables that are required to converge to the equilibrium point $\mathbf{0}$. A continuous-time dynamic system $\mathcal{H}$ is called* Lyapunov stable (LS) with respect to $\mathcal{V}'$ *if for all functions $\mathbf{x}_{\downarrow\mathcal{V}'}(\cdot)$ of $\mathcal{H}$,*

$$\forall \epsilon > 0 : \exists \delta > 0 : \forall t \geq 0 : ||\mathbf{x}(0)|| < \delta \Rightarrow \left|\left|\mathbf{x}_{\downarrow\mathcal{V}'}(t)\right|\right| < \epsilon.$$

$\mathcal{H}$ is called globally attractive (GA) with respect to $\mathcal{V}'$ *if for all functions $\mathbf{x}_{\downarrow\mathcal{V}'}(\cdot)$ of $\mathcal{H}$,*

$$\lim_{t\to\infty} \mathbf{x}_{\downarrow\mathcal{V}'}(t) = \mathbf{0}, \text{ i. e.,} \forall \epsilon > 0 : \exists t_0 \geq 0 : \forall t > t_0 : \left|\left|\mathbf{x}_{\downarrow\mathcal{V}'}(t)\right|\right| < \epsilon,$$

where $\mathbf{0}$ is the origin of $\mathbb{R}^{|\mathcal{V}'|}$. If a system is both Lyapunov stable with respect to $\mathcal{V}'$ and globally attractive with respect to $\mathcal{V}'$, then it is called globally asymptotically stable (GAS) with respect to $\mathcal{V}'$.

Intuitively, LS means that trajectories starting δ-close to the origin remains ϵ-close to the origin. GA means that for each ϵ-distance to the origin, there exists a point in time t_0 such that a trajectory always remains within this distance. It follows that each trajectory is eventually always approaching the origin. This property can be proven using Lyapunov theory [29]. Lyapunov theory was originally restricted to continuous systems, but has been lifted to hybrid systems.

Theorem 1 (Discontinuous Lyapunov Functions [32]). *Let $\mathcal{H}$ be a hybrid automaton with $\mathcal{H} = (\mathcal{V}, \mathcal{M}, \mathcal{T}, \mathit{Flow}, \mathit{Inv})$, and let $\mathcal{V}' \subseteq \mathcal{V}$ be a set of variables that are required* *to converge. If for each $m \in \mathcal{M}$, there exists a set of variables $\mathcal{V}_m$ with $\mathcal{V}' \subseteq \mathcal{V}_m \subseteq \mathcal{V}$ and a continuously differentiable function $V_m : \mathcal{S} \to \mathbb{R}$ such that*

1. *for each $m \in \mathcal{M}$, there exist two class K^∞ functions α and β such that*
$$\forall \mathbf{x} \in \mathit{Inv}(m) : \alpha(||\mathbf{x}_{\downarrow \mathcal{V}_m}||) \leq V_m(\mathbf{x}) \leq \beta(||\mathbf{x}_{\downarrow \mathcal{V}_m}||),$$
2. *for each $m \in \mathcal{M}$, there exists a class K^∞ function γ such that*
$$\forall \mathbf{x} \in \mathit{Inv}(m) : \dot{V}_m(\mathbf{x}) \leq -\gamma(||\mathbf{x}_{\downarrow \mathcal{V}_m}||)$$
for each $\dot{V}_m(\mathbf{x}) \in \left\{ \left\langle \frac{dV_m(\mathbf{x})}{d\mathbf{x}} \,\middle|\, f(\mathbf{x}) \right\rangle \,\middle|\, f(\mathbf{x}) \in \mathit{Flow}(m) \right\}$,
3. *for each $(m_1, G, U, m_2) \in \mathcal{T}$,*
$$\forall \mathbf{x} \in G : V_{m_2}(U(\mathbf{x})) \leq V_{m_1}(\mathbf{x}),$$

then $\mathcal{H}$ is globally asymptotically stable with respect to $\mathcal{V}'$. Each V_m is called a Local Lyapunov Function (LLF) *of m, and the function $V(\mathbf{x}, m) = V_m(\mathbf{x})$ is called the* Global Lyapunov Function *(GLF).*

In Theorem 1, $\left\langle \frac{dV(\mathbf{x})}{d\mathbf{x}} \,\middle|\, f(\mathbf{x}) \right\rangle$ denotes the inner product of the gradient of a Lyapunov function V and a flow function $f(\mathbf{x})$. Please refer to [32, p.43] for the details of K^∞ functions. Throughout the paper we denote by *mode constraints* the constraints of Type 1 and Type 2 and by *transition constraints* the constraints of Type 3.

Note 1. *For each LF V (local or global), it holds that for any trajectory $\mathbf{x}(\cdot)$ the LF's value does not increase, i. e., $\forall t_0, t \geq 0 : V(\mathbf{x}(t_0 + t)) \leq V(\mathbf{x}(t_0))$ (or $\forall t_0, t \geq 0 : V(\mathbf{x}(t_0 + t), m(t_0 + t)) \leq V(\mathbf{x}(t_0), m(t_0))$ for the GLF).*

Stabhyli generates constraint systems which – using the so called *sums-of-squares* method [41] and the S-Procedure [7] – are relaxed to a linear matrix inequality (LMI). The LMI can then be solved by a *semi-definite program (SDP)*. For this purpose, we use the numerical solver CSDP [6] or SDPA [18] is in charge. If the LMI is feasible, then each solution represents a valid Lyapunov function.

Note 2. *Since* Stabhyli *employs the sums-of-squares method, every computed Lyapunov function is representable as a sum-of-squares. However, in the following we restrict ourselves to* quadratic *Lyapunov functions as they are sufficient for many applications.*

3.2 Checking Reachability via SoapBox

SoapBox is a tool for reachability analysis of hybrid systems. It is implemented in Matlab. SoapBox handles hybrid systems with continuous dynamics described by linear differential inclusions and arbitrary affine maps for discrete updates. The invariants, guards, and sets of reachable states are given as convex polyhedra. Internally, the reachability algorithm of SoapBox is based on symbolic orthogonal projections (sops) [23]. Sops extend the half-space representation of polyhedra ($\mathcal{H}$-polyhedra) such that the operations required for the post-image computation, including convex hulls, Minkowski sums, affine maps, and intersections, can be performed efficiently and exactly. Hence, using sops yields tighter overapproximation than support functions.

A drawback, which sops and support functions have in common, is that there is no efficient and exact method for deciding subset-relations in general. Hence, an exact fix-point check is hard to achieve. However, at least the decision whether a given set – represented as a sop or by support functions – is contained in an $\mathcal{H}$-polyhedron or not can be done efficiently.

Although SoapBox is a fully functional model checker handling continuous and discrete updates of a hybrid system, for our approach it suffices to present the mode-specific continuous post-image computation provided by the method

$$\texttt{Reach}(\mathit{Init}, \mathit{Flow}(m), \mathit{Inv}(m), \mathit{Safe}, T).$$

It expects as input an $\mathcal{H}$-polyhedron representing the initial states *Init*, a differential inclusion $\mathit{Flow}(m)$ describing the differential inclusion $\dot{\mathbf{x}} = A\mathbf{x} + \mathbf{E}$, where $\mathbf{E}$ is an $\mathcal{H}$-polytope modeling the bounded input, an $\mathcal{H}$-polyhedron representing the invariant $\mathit{Inv}(m)$, an $\mathcal{H}$-polyhedron representing the target set[1] T, and an $\mathcal{H}$-polyhedron *Safe*. The method computes a tight overapproximation of all reachable states on trajectories solving

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + \mathbf{E},\ \mathbf{x}(0) \in \mathit{Init},\ \forall t \geq 0\colon\ \mathbf{x}(t) \in \mathit{Inv}(m),$$

until either all states leave the invariant or all states have entered the set *Safe*. Then, it returns a polyhedral set representing a tight polyhedral overapproximation of the intersection of the reachable states with the target set T.[2]

4 Hybrid Stability and Reachability Tool

We present our algorithm that combines safety and stability analysis. We show how the combination of both analyses mutually simplifies the verification task.

While GAS is defined over trajectories, Theorem 1 argues over states and searches for LLFs which have to be compatible with respect to the transition constraints. We observed that, often, we are able to find the LLFs but fail to establish the necessary compatibility with respect to the transition constraints. Among other reasons, this may be due to cyclic dependencies imposed by the transitions of the hybrid automaton.

Our approach is to unroll the hybrid automaton to an equivalent hybrid automaton for which we can verify GAS via Theorem 1. Here, "equivalence" denotes that both automata exhibit an identical continuous behavior, i. e., for each trajectory of either automaton there is a trajectory of the other automaton with the same continuous evolution. Hence, the original hybrid automaton is also GAS.

We define the following notion of the depth of a reachable state. The set $\mathcal{R}_0 = \mathit{Inits}$ is the set of all reachable states of depth 0. Given the set $\mathcal{R}_n$ of all reachable states at depth n, we recursively define $\mathcal{R}_{n+1}$ as the set of all tuples $(\mathbf{x}_{n+1}, m_{n+1})$ which are reachable from any tuple $(\mathbf{x}_n, m_n) \in \mathcal{R}_n$ by a combination of a continuous evolution and a subsequent discrete transition, i. e., there exists a flow $f \in \mathit{Flow}(m_n)$,

[1] Usually, `Reach`() handles finite unions of polyhedral target sets at once. For the sake of simplicity, our presentation is restricted to a single target set.

[2] As before, `Reach`() returns unions of polyhedra, each representing a single traversal of a target set.

an instant of time $t \geq 0$, and a transition $(m_n, G, U, m_{n+1}) \in \mathcal{T}$ such that for $\mathbf{y}(t) = \int_0^t f(\mathbf{y})\ d\tau$ it holds $\mathbf{y}(\tau) \in \mathit{Inv}(m_n)$ for all $\tau \in [0,t]$, $\mathbf{y}(0) = \mathbf{x}_n$, $\mathbf{y}(t) \in G$, and $U(\mathbf{y}(t)) = \mathbf{x}_{n+1}$. Clearly, $(\mathbf{x}, m)$ is reachable if and only if there exists some n with $(\mathbf{x}, m) \in \mathcal{R}_n$.

Our idea is as follows: First we compute for each mode a Lyapunov function, called a "single-mode Lyapunov Function" (SMLF). Then, starting with $\mathcal{R}_0$, we successively compute the sets $\mathcal{R}_n$. For each mode this reduces to the reachability problem of computing the *reachset*, that is the set obtained by computing all reachable states in the transition guards and subsequent application of the respective discrete updates. During each reachset computation, we additionally assess safety, i. e., we decide whether *Unsafe* can be reached or not. Moreover, the SMLFs enable us to compute safe sets which substantially simplify the reachability analysis: An appropriate safe set has no intersection with neither the unsafe set nor any guard of an outgoing transition and it is invariant under the flow. Thus, we may stop the reachability analysis as soon as a safe set is entered.

If this approach successfully terminates, then it does not only help to ensure safety, but we also obtain an unrolled (or unfolded) version of the original hybrid automaton. The unrolled automaton is free of cyclic dependencies. Additionally, guards are restricted to their intersection with the reachable states. Both facts enormously simplify the task of establishing compatibility of the SMLFs with the transition constraints, which finally yields a global Lyapunov function.

4.1 Single-mode Lyapunov Function Computation

As mentioned above, in the first step, we compute Lyapunov functions for each mode in isolation. We call a Lyapunov function *single-mode Lyapunov function* (SMLF), if it satisfies the mode constraints, but not necessarily the transition constraints of Theorem 1.

Stabhyli can compute Lyapunov functions automatically. To obtain the *single-mode Lyapunov functions*, we simply "feed" Stabhyli with a single-mode automaton $\mathcal{H}_m$ for every $m \in \mathcal{M}$, where the *single-mode hybrid automaton* is defined as

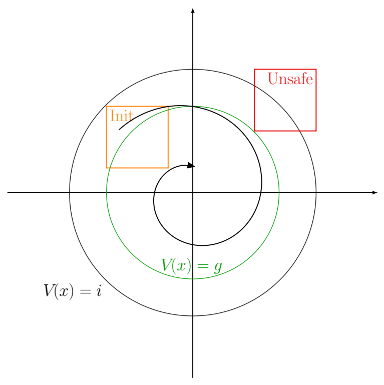
$$\mathcal{H}_m = (\mathcal{V}, \{m\}, \varnothing, \{m \mapsto \mathit{Flow}(m)\}, \{m \mapsto \mathit{Inv}(m)\}, \{(\mathit{Inv}(m), m)\}).$$

Computing the single-mode Lyapunov functions certifies that each single-mode hybrid automaton is GAS. Note that this is not sufficient to conclude that the full hybrid automaton is GAS. However, we exploit single-mode Lyapunov function during the following reachability analysis. In a final step, we combine the single-mode Lyapunov functions with the results of the reachability analyses to verify GAS of the full hybrid automaton (see Section 4.5).

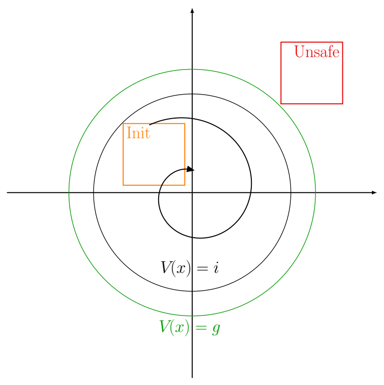
4.2 SafeSet Computation

As in the previous section, we restrict our attention to a single mode m of a hybrid automaton $\mathcal{H}$. Firstly, we introduce the notion of a level set.

Definition 3 (Level Set)**.** *Let $V(\cdot)$ be a Lyapunov function of mode m. For any $s \geq 0$, we call $\mathcal{L}_{V,s} = \{\mathbf{x} \mid V(\mathbf{x}) \leq s\}$ a* level set *of mode m.*



(a) Safe after entering $\mathcal{L}_{V,g}$



(b) Initially safe

Figure 1: Sketch of safe sets

A Lyapunov function assigns a value to any state of the state space, and its values along any trajectory does not increase. Hence, all trajectories starting in a level set will not leave the level set. In general, we call any subset of states which cannot be left by the continuous trajectories an *invariant set* (for the formal definition see Definition 4). This property allows a strong prediction on the future of the trajectory. However, the prediction does not suffice to establish safety if the intersection of the invariant set with *Unsafe* or some guards is not empty, since then the mode could be unsafe or it could be left by a discrete transition.

Definition 4 (Safe Set, Avoid Set, and Invariant Set)**.** *Let $\mathcal{A}$ and $\mathcal{S}'$ be two subset of $\mathcal{S}$ such that $\mathcal{S}' \subset Inv(m)$. If for all trajectories $\mathbf{x}(\cdot)$ with $\mathbf{x}(0) \in \mathcal{S}'$ and for all $t \geq 0$ it holds*

$$\forall\tau\ (0 \leq \tau \leq t \rightarrow \mathbf{x}(\tau) \in Inv(m)) \implies \forall\tau\ (0 \leq \tau \leq t \rightarrow \mathbf{x}(\tau) \in \mathcal{S}') \tag{1}$$
$$\forall\tau\ (0 \leq \tau \leq t \rightarrow \mathbf{x}(\tau) \in Inv(m)) \implies \forall\tau\ (0 \leq \tau \leq t \rightarrow \mathbf{x}(\tau) \notin \mathcal{A}), \tag{2}$$

then $\mathcal{A}$ is called an avoid set *and $\mathcal{S}'$ is called a* safe set *for $\mathcal{A}$ of a mode m. A safe set for $\mathcal{A}$ is denoted by $Safe_{\mathcal{A}}$.*
The set $\mathcal{S}' \subseteq Inv(m)$ is an invariant set *if condition (1) holds.*

Proposition 1. *Any level set of a mode is also an invariant set of the mode.*

Proof. Let $\mathcal{L}_{V,s}$ be a level set of some mode, $\mathbf{x}(\cdot)$ a trajectory with $\mathbf{x}(0) \in \mathcal{L}_{V,s}$ and $t \geq 0$. Assume, $\mathbf{x}(\tau)$ with $0 \leq \tau \leq t$ is a trajectory segment that does not leave the invariant. A Lyapunov function assigns a value to any state of the state space, and the values along any trajectory do not increase. Hence, $\mathbf{x}(\tau)$ will not leave the level set. □

Proposition 2. *Let $\mathcal{S}'$ be an invariant set of a mode. If $\mathcal{A} \cap \mathcal{S}' = \varnothing$ holds for some set $\mathcal{A}$, then $\mathcal{S}'$ is a safe set for the avoid set $\mathcal{A}$.*

Proof. $\mathcal{S}'$ is an invariant set. Hence, each point on any trajectory which emanates from $\mathcal{S}'$ is either within $\mathcal{S}'$ or it violates the invariant. Since $\mathcal{S}'$ and $\mathcal{A}$ have no points in common, none of the points in $\mathcal{A}$ can be reached by a trajectory emanating from $\mathcal{S}'$ without violating the invariant. □

Corollary 1. *Any level set of a mode which has an empty intersection with some set $\mathcal{A}$ is a safe set for the avoid set $\mathcal{A}$.*

Proof. Follows immediately from Proposition 2 and Proposition 1. □

Now, the alleviating argument for the reachability analysis is that a certain set, like *Unsafe* or a guard, cannot be reached as soon as a safe set for the respective set is entered.

Following Corollary 1 we aim for maximizing the extent of the level set $\mathcal{L}_{V,s} = \{\mathbf{x} \mid V(\mathbf{x}) \leq s\}$ for a given avoid set $\mathcal{A}$. Let $g < \inf_{\mathbf{x}\in\mathcal{A}} V(\mathbf{x})$ be a strict lower bound for the values of the Lyapunov function over $\mathcal{A}$. Since all states with a lower value are guaranteed not to be in $\mathcal{A}$, the set $\mathit{Safe}_{\mathcal{A}} = \mathcal{L}_{V,g}$ is a safe set for $\mathcal{A}$. Furthermore, if we find an upper bound $i \geq \sup_{\mathbf{x}\in \mathit{Init}} V(\mathbf{x})$ with $g \geq i$, then $\mathcal{A}$ is unreachable from all initial states, and we can omit the reachset computation entirely. Both cases are visualized in Figure 1.

This yields the following basic algorithm:

1. Determine the lowest LF's value $b = \inf\{V(\mathbf{x}) \mid \mathbf{x} \in \mathcal{A}\}$ of all states in the avoid set.
2. Determine the highest LF's value $i = \sup\{V(\mathbf{x}) \mid \mathbf{x} \in \mathit{Init}\}$ of all states in the initial set.
3. If $i < b$, then the *Init* is safe and otherwise subtract a safety margin ϵ on the LF's value, $g = b - \epsilon$, to build the safe set $\mathit{Safe}_{\mathcal{A}} = \mathcal{L}_{V,g}$.

Although finding the LF's minimum (resp. maximum) values in a compact set can be done via numerical optimization, our implementation performs a bisectioning combined with Z3. The procedure is as follows:

1. Guess an initial interval $[a = 0, b]$ of the LF's infimum (resp. supremum) value.
2. Refine the initial interval:
   Ask Z3 to find an $\mathbf{x} \in \mathcal{A}$ (resp. $\mathbf{x} \in \mathit{Init}$) such that $V(\mathbf{x}) \leq b$ (resp. $V(\mathbf{x}) \geq b$). While such an $\mathbf{x}$ cannot (resp. can) be found increase b.
3. If the initial $[a, b]$ is found we enter the bisectioning:
   Ask Z3 to find an $\mathbf{x} \in \mathcal{A}$ (resp. $\mathbf{x} \in \mathit{Init}$) such that $V(\mathbf{x}) \leq \frac{b-a}{2}$ (resp. $V(\mathbf{x}) \geq \frac{b-a}{2}$). While $b - a \geq \epsilon$ continue with $[a, \frac{b-a}{2}]$ (resp. $[\frac{b-a}{2}, b]$) if such an $\mathbf{x}$ can be found and continue with $[\frac{b-a}{2}, b]$ (resp. $[a, \frac{b-a}{2}]$) otherwise.
4. A safe approximation of the LF's infimum (resp. supremum) value is given by a (resp. b).

4.3 SafeBox Conversion

In order to use safe sets for trajectory truncation in a polyhedral-based tool like SoapBox, we generate polyhedral underapproximatons of safe sets. In this section we shortly describe the idea of our method. Details of this method can be found in Section 7. Stabhyli generates quadratic Lyapunov functions. Hence, a level set in our context is a quadric $\left\{\mathbf{x} \mid \mathbf{x}^T V \mathbf{x} \leq c^2\right\}$ with $c > 0$ and a symmetric matrix V. Projectively principal axis transformation yields an invertible matrix L and a diagonal matrix E, whose coefficients are equal to -1, 1, or 0, and sorted in descending order such that $\tilde{V} = \begin{pmatrix} V & \mathbf{0} \\ \mathbf{0}^T & -c^2 \end{pmatrix} = L\tilde{E}L^T$. By Sylverster's law of inertia, the numbers of negative, positive, and zero coefficients are uniquely determined. Furthermore, since $\tilde{V}$ contains the block matrix $-c^2$, the last coefficient of $\tilde{E}$ is -1, i. e., $\tilde{E} = \begin{pmatrix} E & \mathbf{0} \\ \mathbf{0}^T & -1 \end{pmatrix}$.

Function 1: The Prepare Function

```
    input  : A hybrid automaton $\mathcal{H}$, a set $Unsafe$.
    output: A set of LFs $LFs$, and a prepared version of $\mathcal{H}$.
1   $Inits', \mathcal{T}', LFs \leftarrow \varnothing$;
    // compute Lyapunov functions for each mode in isolation
2   foreach $m \in \mathcal{H}.\mathcal{M}$ do $LFs(m) \leftarrow$ computeLF($Flow(m), Inv(m)$);
3   $\mathcal{T}' \leftarrow \mathcal{H}.\mathcal{T}$;                                   // copy transitions
4   foreach $(Init, m) \in \mathcal{H}.Inits$ do            // separate each initial mode
5       $m' \leftarrow$ copyMode($\mathcal{H}, Unsafe, LFs, m$);                  // copy the mode
6       $Inits' \leftarrow Inits' \cup \{(Init, m')\}$ ;                   // add the new init
        // copy each outgoing transition to the new mode
7       foreach $(m, G, U, m_2) \in \mathcal{T}'$ do $\mathcal{H}.\mathcal{T} \leftarrow \mathcal{H}.\mathcal{T} \cup \{(m', G, U, m_2)\}$;
8   $\mathcal{H}.Inits \leftarrow Inits'$ ;                             // replace initial states
```

This yields the implication $\mathbf{y}^T E' \mathbf{y} \leq 1 \Rightarrow \mathbf{y}^T E \mathbf{y} \leq 1$, where E' is obtained from E by replacing all occurrences of -1 by 0. Hence, the cylinder $\{\mathbf{y} \mid \mathbf{y}^T E' \mathbf{y} \leq 1\}$ over a lower-dimensional unit sphere is the largest inscribed convex and cylindrical set of $\{\mathbf{y} \mid \mathbf{y}^T E \mathbf{y} \leq 1\}$. Now, given template $\mathcal{H}$-polyhedra with circumspheres of arbitrary dimension, it is easy to generate an inscribed $\mathcal{H}$-polyhedron of the spherical cylinder. For our experiments we used hypercubes and cross-polytopes. It remains to compute the image of the resulting polyhedra under the inverse transformation to $\binom{\mathbf{y}}{\mu} = L^T \binom{\mathbf{x}}{\lambda}$, which is computationally easy but involves non-trivial insights in projective geometry.[3]

4.4 Unrolling Algorithm

Function 1 and Algorithm 2 show the unrolling algorithm. Function 1 is a *preparation function* that creates copies of the modes as well as all outgoing edges for each initial state set. The function `copyMode` creates a fresh copy of a mode with the same flow, invariant, SMLF, and unsafe set as the original mode. Algorithm 2 is the main *unrolling algorithm.* It is executed after the preparation function. It maintains a job queue which is initialized with the initial state sets. Until the job queue is empty, it selects a job, computes safe sets and reachsets with respect to *Unsafe* and the guards of the outgoing transition. An intersection of the reachset with *Unsafe* shows that the hybrid system is unsafe. Intersections with guards are used to tighten the guards of transitions and are enqueued for further exploration. This unrolls the hybrid automaton in a breath first manner. If the job queue is empty, then the unrolling is followed by a post-processing. The post-processing removes all nodes that are not connected to a mode of the initial set. The result is a forest of hybrid automata describing the trajectories abstractly. The unrolled hybrid automaton can be proven stable very efficiently according to Corollary 2. In fact, since the unrolled hybrid automaton is acyclic, the single-mode Lyapunov functions computed may be reused.

[3]Actually, we use a projective generalization of polyhedra similar to the notion of *projective polyhedra* as it has been introduced in [19].

Algorithm 2: The Unrolling Algorithm

```
input  : A hybrid automaton $\mathcal{H}$, a set $Unsafe$, and a set of LFs $LFs$.
output: An unrolled version of $\mathcal{H}$.
1  Jobs $\leftarrow \mathcal{H}.Inits$;                // start from each initial state set
2  while Jobs $\neq \varnothing$ do
3      $(Init, m) \leftarrow$ pop(Jobs);                     // select a job
       // compute safe sets with respect to unsafe and convert them to
          safe boxes
4      $Safe_{Unsafe} \leftarrow$ convertToBoxes(safeSets($LFs(m), Init, Unsafe(m)$));
       // compute reachset with respect to unsafe
5      $R \leftarrow$ Reach($Init, Flow(m), Inv(m), Safe_{Unsafe}, Unsafe(m)$);
6      if $R \neq \varnothing$ then markUnsafe($m, R$);          // model is unsafe
7      $\mathcal{T}' \leftarrow \mathcal{H}.\mathcal{T}$;                          // copy transitions
       // check reachability of each outgoing transition
8      foreach $(m, G, U, m_2) \in \mathcal{T}'$ do
9          $\mathcal{H}.\mathcal{T} \leftarrow \mathcal{H}.\mathcal{T} \setminus \{(m, G, U, m_2)\}$;        // remove old transition
           // compute safe sets with respect to guard and convert them
              to safe boxes
10         $Safe_G \leftarrow$ convertToBoxes(safeSets($LFs(m), Init, G$));
           // compute reachset with respect to guard
11         $R \leftarrow$ Reach($Init, Flow(m), Inv(m), Safe_G, G$);
12         if $R = \varnothing$ then continue;               // guard unreachable
13         $m' \leftarrow$ copyMode($\mathcal{H}, Unsafe, LFs, m_2$);           // copy the mode
14         $\mathcal{H}.\mathcal{T} \leftarrow \mathcal{H}.\mathcal{T} \cup \{(m, R, U, m')\}$; // add refined incoming transition
           // copy each outgoing transitions to the new mode
15         foreach $(m_2, G_2, U_2, m_3) \in \mathcal{T}'$ do $\mathcal{H}.\mathcal{T} \leftarrow \mathcal{H}.\mathcal{T} \cup \{(m', G_2, U_2, m_3)\}$;
16         Jobs $\leftarrow$ Jobs $\cup\{$(apply($R, U$)$, m'$)$\}$;      // append updated postset
```

4.5 Global Lyapunov Function Computation

Now that we have established safety, we verify GAS of the unrolled hybrid automaton reusing the single-mode Lyapunov functions. Since both automata are equivalent, this implies GAS of the original hybrid automaton.

For completeness we introduce notations as well as a theorem from [32] which we will adapt to our needs in Corollary 2. Let us start with some basic graph notations.

Definition 5 (Directed Graph)**.** *A* directed graph $\mathcal{G}$ *is a tuple* $(\mathcal{V}, \mathcal{E})$*, where* $\mathcal{V}$ *is a finite set of vertices and* $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ *is a finite set of edges. A* subgraph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ *is a graph where* $\mathcal{V}' \subseteq \mathcal{V}$ *and* $\mathcal{E}' \subseteq \mathcal{E}$*.*

Definition 6 (Strongly Connected Component)**.** *A* strongly connected component *(SCC) of a directed graph* $\mathcal{G}$ *is a maximal subgraph* $\mathcal{G}'$ *such that for each pair of nodes* $n_1 \neq n_2$ *in* $\mathcal{G}'$ *there exists a forward path from* n_1 *to* n_2*. An edge* (n_1, n_2) *connecting two SCCs is called a* bridge*.*

In Definition 6 *maximality* means that no vertex may be added without violating the existence of the forward paths.

In order to reason about the graph structure of a hybrid automaton, we define the underlying or corresponding graph as follows:

Definition 7 (Underlying Graph)**.** *An* underlying graph $\mathcal{G}(\mathcal{H}) = (\mathcal{V}, \mathcal{E})$ *of a hybrid automaton* $\mathcal{H} = (\mathcal{V}, \mathcal{M}, \mathcal{T}, \mathit{Flow}, \mathit{Inv}, \mathit{Inits})$ *is a directed graph where* $\mathcal{V} = \mathcal{M}$ *is a finite set of vertices and* $\mathcal{E} = \{(m_1, m_2) \mid (m_1, G, U, m_2) \in \mathcal{T}\}$ *is a finite set of edges.*

A decomposition of the underlying graph into SCCs allows us to apply the following theorem:

Theorem 2 (Decomposition into SCCs [32, Theorem 4.1, Remark 4.3])**.** *Let* $\mathcal{H}$ *be a hybrid automaton with* $\mathcal{H} = (\mathcal{V}, \mathcal{M}, \mathcal{T}, \mathit{Flow}, \mathit{Inv}, \mathit{Inits})$*. If all sub-automata pertaining to the SCCs of* $\mathcal{G}(\mathcal{H})$ *are globally attractive then so is* $\mathcal{H}$*. If all SCCs are Lyapunov stable and if for all transitions* $(m_1, G, U, m_2) \in \mathcal{T}$ *corresponding to bridges of* $\mathcal{G}(\mathcal{H})$ *it holds that*

$$\exists c > 0 : \forall \mathbf{x} \in G : V_{C_2}(U(\mathbf{x}), m_2) \leq c \cdot V_{C_1}(\mathbf{x}, m_1)$$

where C_i *is the SCC containing* m_i *for* $i \in 1, 2$*, then* $\mathcal{H}$ *is Lyapunov stable. Therefore,* $\mathcal{H}$ *is GAS.*

Therefore, if a hybrid automaton consists of SCCs which contain at most one mode – as it is the case for a unrolled hybrid automaton – then we can compute Lyapunov functions for each mode in isolation, and we can check satisfiability of the transition constraints afterwards.

This leads to the following corollary which is basically a reformulation of Theorem 2 in the context of unrolled hybrid automata.

Corollary 2 (GAS of an unrolled Hybrid Automaton)**.** *Let* $\mathcal{H}$ *be an unrolled hybrid automaton with* $\mathcal{H} = (\mathcal{V}, \mathcal{M}, \mathcal{T}, \mathit{Flow}, \mathit{Inv}, \mathit{Inits})$*. If all modes are globally attractive, then so is* $\mathcal{H}$*. If all modes are Lyapunov stable and for all transitions* $(m_1, G, U, m_2) \in \mathcal{T}$ *it holds that*

$$\exists c > 0 : \forall \mathbf{x} \in G : V_{m_2}(U(\mathbf{x})) \leq c \cdot V_{m_1}(\mathbf{x}),$$

then $\mathcal{H}$ *is Lyapunov stable. Consequently,* $\mathcal{H}$ *is GAS.*

Proof. Follows from Theorem 2 because, since $\mathcal{H}$ is unrolled and, in turn, acyclic, every mode of $\mathcal{H}$ belongs to exactly one SCC. □

Since we already have single-mode Lyapunov functions, it remains to show that for each transition the factor c as used in Corollary 2 actually exists. If this is successful, then we can conclude that the hybrid automaton is GAS.

5 Benchmarking

In this section we present four benchmarks and compare the time needed for verification of their respective properties. The benchmark set contains one example where we verify stability and safety (actually, it is part of a case study on parallel composition of hybrid systems), two examples with more complex discrete behavior for which we only verify pure stability properties,[4] and one example for which it is impossible to prove stability without further reachability analysis.

[4]Note that the unrolling algorithm performs a complete reachability analysis. Hence, extending the examples by an unsafe set is rather trivial.

Example 1: The Velocity Controller (VC) – visualized in Figure 2 – is part of the Advanced Driver Assistance System (ADAS) presented and analyzed in [13, 14, 25]. The ADAS consists of two concurrent controllers (as well as helper components) that cooperatively achieve the following objectives:

(o1) maintain a centrifugal force comfortable for a driver,
(o2) bring and then keep the car on the center of its lane,
(o3) control the speed whereby also considering driver requests for a certain speed value.

The VC contributes to Objective (o1) and Objective (o3).

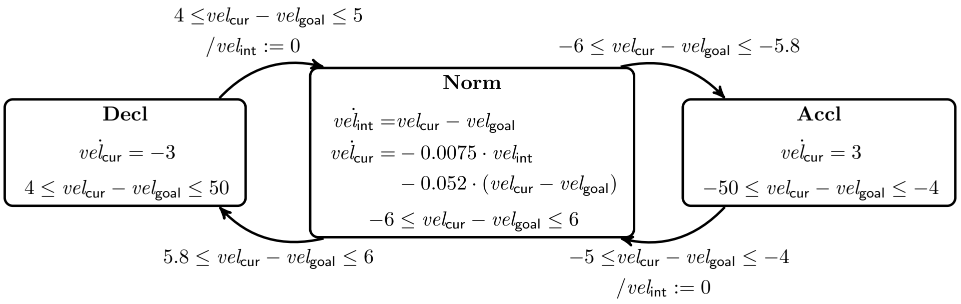

$4 \leq vel_{\mathsf{cur}} - vel_{\mathsf{goal}} \leq 5$
$/vel_{\mathsf{int}} := 0$
$-6 \leq vel_{\mathsf{cur}} - vel_{\mathsf{goal}} \leq -5.8$
Norm
$\dot{vel}_{\mathsf{int}} = vel_{\mathsf{cur}} - vel_{\mathsf{goal}}$
$\dot{vel}_{\mathsf{cur}} = -0.0075 \cdot vel_{\mathsf{int}} - 0.052 \cdot (vel_{\mathsf{cur}} - vel_{\mathsf{goal}})$
$-6 \leq vel_{\mathsf{cur}} - vel_{\mathsf{goal}} \leq 6$
Decl
$\dot{vel}_{\mathsf{cur}} = -3$
$4 \leq vel_{\mathsf{cur}} - vel_{\mathsf{goal}} \leq 50$
Accl
$\dot{vel}_{\mathsf{cur}} = 3$
$-50 \leq vel_{\mathsf{cur}} - vel_{\mathsf{goal}} \leq -4$
$5.8 \leq vel_{\mathsf{cur}} - vel_{\mathsf{goal}} \leq 6$
$-5 \leq vel_{\mathsf{cur}} - vel_{\mathsf{goal}} \leq -4$
$/vel_{\mathsf{int}} := 0$

Figure 2: The Velocity Controller [13]

The model has three modes: one mode with a constant acceleration, one mode with a constant deceleration and one mode doing the fine tuning via a PI controller. The VC's task is to drive the current velocity of the vehicle vel_{cur} to a desired velocity vel_{goal}. This desired velocity is given by an external input that might be updated discretely. The verification task is to show that

- vel_{cur} converges to vel_{goal},
- if $vel_{\mathsf{cur}} - vel_{\mathsf{goal}} \leq -3$ initially holds then always holds $vel_{\mathsf{cur}} - vel_{\mathsf{goal}} \leq 3$,
- if $vel_{\mathsf{cur}} - vel_{\mathsf{goal}} \in [-3, -2]$ initially holds then always holds $vel_{\mathsf{cur}} - vel_{\mathsf{goal}} \leq 2$,
- if $vel_{\mathsf{cur}} - vel_{\mathsf{goal}} \in [-2, 0]$ initially holds then always holds $vel_{\mathsf{cur}} - vel_{\mathsf{goal}} \leq 1$.

The later three are safety properties and restrict the peak-overshoot. For the verification, all properties are considered under the assumption that vel_{goal} remains constant once set.

Example 2: The Automatic Cruise Controller (ACC) regulates the velocity of a vehicle. Figure 3 shows the controller as an automaton. The task of the controller is to approach a user-chosen velocity – here the variable v represents the velocity relative to the desired velocity. The ACC is a bit more complex than the VC as it has two different brakes: a service break for minor corrections and an emergency break for huge differences. Both brakes have an activation phase, in which the deceleration is continuously increased.

Example 3: The Spidercam is a movable robot equipped with a camera used at sport events such as a football matches. The robot is connected to four cables. Each cable is attached to a motor that is placed in a corner high above the playing field of a sports arena. By winding and unwinding the cables, the spidercam is able to reach nearly any position in the three-dimensional space above the playing field.

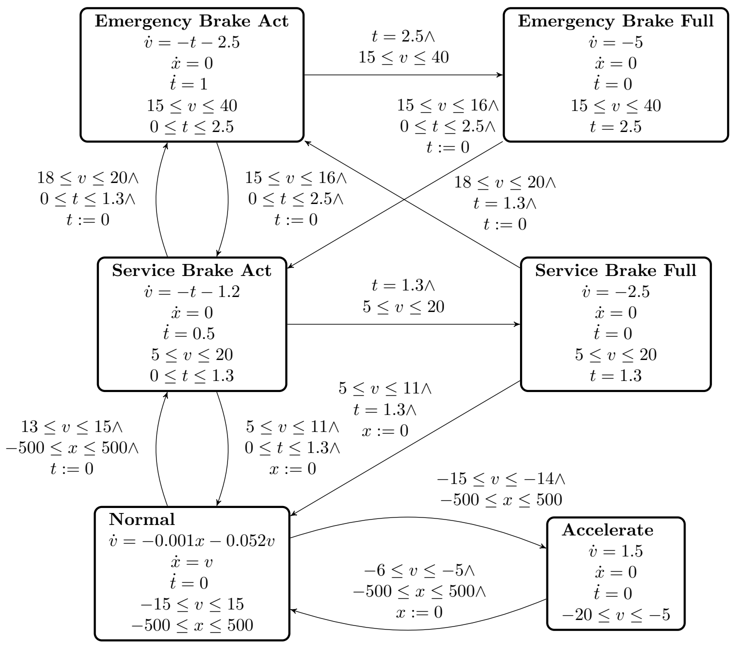


Figure 3: The Automatic Cruise Controller [32]

Figure 4 shows a very simple model of such a spidercam in the plane. The target is to stabilize the camera at a certain position. The continuous variables x and y denote the distance relative to the desired position on the axis induced by the cables. In the model, we assume a high-level control of the motor engines, i. e., the movement is on x and y axes instead of a low-level control of each individual motor. The model has nine modes: one mode that controls the behavior while being close to the desired position, four modes corresponding to nearly straight movements along one of the axes, and four modes cover the quadrants between the axes. The maximal velocity in the direction of each axis is limited from above by $0.6\frac{m}{s}$. Thus, in the four modes corresponding to the quadrants, the movement in each direction is at full speed. In the four modes corresponding to the axes, the movement on the particular axis is at full speed while the movement orthogonal to the axis is proportional to the distance. In the last mode, the speed in both directions is proportional to the distance.

Example 4: The Artificial Example is a model (see Figure 5a) which cannot be proven stable without further reachability information. The reason is that the guard of the transition from mode `Turbo Fast` to `Wait` does not restrict values of x. Thus, naïvely generating constraints due to Theorem 1 leads to the following snippet of constraints

$$\begin{aligned} \forall x, t : x \in [-100, 100] \wedge t \in [0, 5] \Rightarrow 0 < V_{\texttt{Wait}}(x, t) \\ \forall t : t \in [0, 5] \Rightarrow V_{\texttt{Turbo Fast}}(0, t) = 0 \\ \forall x : V_{\texttt{Wait}}(0.9x + 0.1, 0) \le V_{\texttt{Turbo Fast}}(x, 5). \end{aligned}$$

Obviously, no such $V_{\texttt{Wait}}, V_{\texttt{Turbo Wait}}$ exist. On the other hand, due to the unrolling, we can conclude that the transition may only be taken with $x \in [10.1, 38.9]$ which

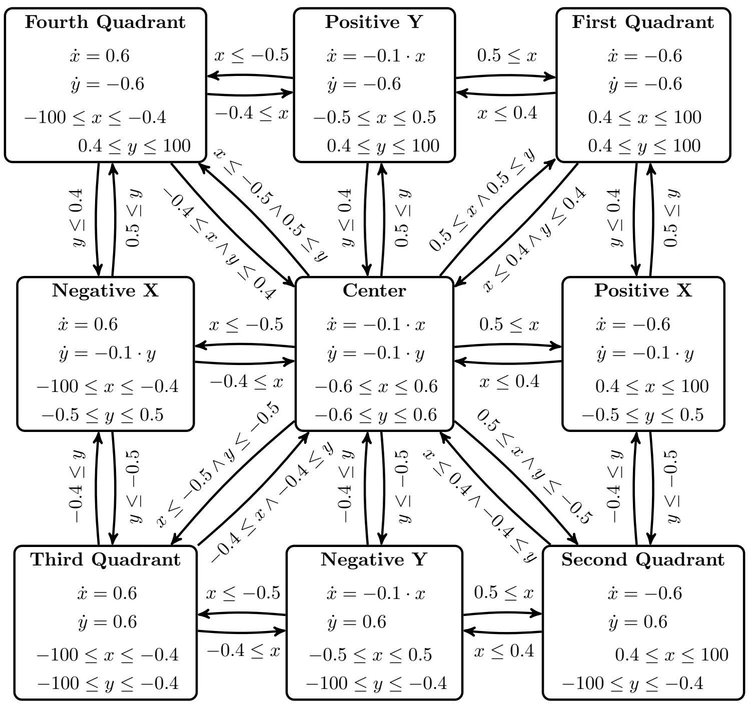


Figure 4: The Simple Planar Spidercam

allows us to replace the last constraint by

$$\forall x : x \in [10.1, 38.9] \Rightarrow V_{\texttt{Wait}}(0.9x + 0.1, 0) \leq V_{\texttt{Turbo Fast}}(x, 5),$$

and, indeed, for $x \geq 10$ the Lyapunov function value may not increase.[5] The unrolled automaton is sketched in Figure 5b.

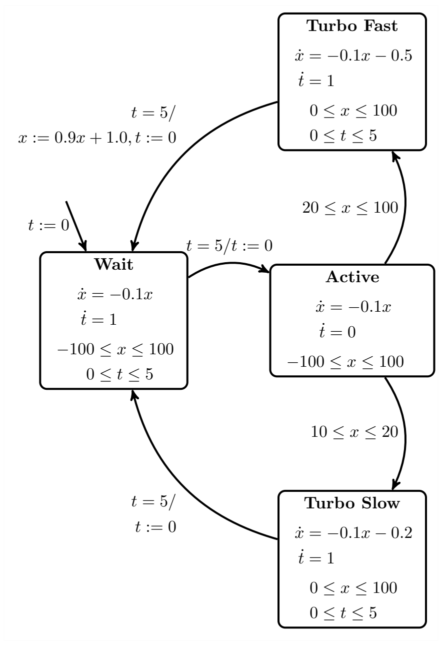
Results

Tabular 1 shows, in order, the depth of the unrolled hybrid automaton and the time needed to compute the SMLFs, the safe sets, the inscribed polygon (hypercube and cross-polytope), the reachability information, and, in the last column, the total runtime.[6] Since SoapBox is written in Matlab and our current prototype tool runs SoapBox for each reachset computation, we have included two times: with and without the time for the Matlab reinitialization (once for each computation).

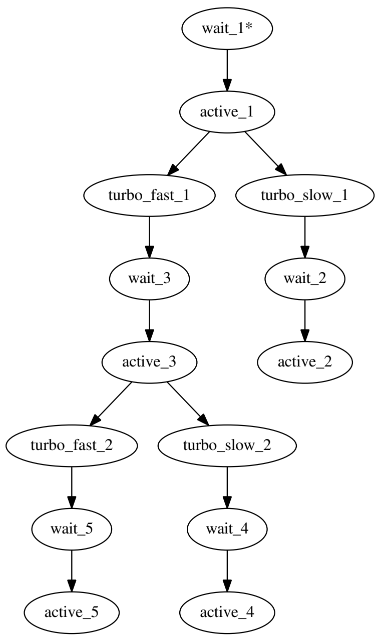
In Tabular 2 we compare the runtime of the proposed approach with the runtime of Stabhyli searching for a common Lyapunov function, Stabhyli searching for a piecewise Lyapunov function, and Stabhyli using the decompositional approach. We can conclude that although the proposed unrolling technique is not the fastest, its runtime is comparable to the runtime of the decompositional approach and may handle examples that other approaches cannot handle. Furthermore, if a benchmark

[5]Note that the update increases x in case the transition is taken with $x < 10$. This would render the system non-LS (cf. Definition 2). However, reachability information reveals that no such trajectory exists.

[6]An Intel© Core$^{\text{TM}}$ i7-3770T CPU with 2.50GHz and 8GB of RAM (in single-core mode) was used to run the benchmarks.



(a) unmodified



(b) unrolled

Figure 5: An Artificial Example

exposes a safety property, too (like the VC), then additional time is required to verify the safety properties which nullifies the advantage of the shorter runtime.

6 Summary

We have presented an approach to verify both, safety and stability properties of hybrid systems. In contrast to the simple approach of verifying the properties separately, we merge the verification procedures such that it makes either verification task simpler. Our approach allows us to exploit the knowledge that is gained during the verification of one problem in the verification of the other one. From the safety perspective, we use the fact that level sets – which are obtained from Lyapunov functions – reveal subsets of the state space which are known to be safe. We stop

	Depth	Stabhyli	SafeSet	BoxConvert	SoapBox	Total Time
Example 1	3	0.39s	0.16s	2.4s	38.6s	41.55s
Example 2	6	0.82s	1s	57.2s	291.6s	350.62s
Example 3	5	2.14s	580.3s	11s	3814.7s	4408.14s
Example 4	8	1.65s	5.67s	0.06s	58.04s	65.42s

(a) including Matlab startup

	Depth	Stabhyli	SafeSet	BoxConvert	SoapBox	Total Time
Example 1	3	0.39s	0.16s	2.4s	14.6s	17.55s
Example 2	6	0.82s	1s	57.2s	39.6s	98.62s
Example 3	5	2.14s	580.3s	11s	1098.7s	1692.14s
Example 4	8	1.65s	5.67s	0.06s	10.04s	17.42s

(b) excluding Matlab startup

Table 1: Detailed Computation Times

	Stabhyli (common)	Stabhyli (piecewise)	Stabhyli (decomp.)	Unrolling
Example 1	X	0.21s	22.29s	17.55
Example 2	X	1.70s	112.99s	98.62
Example 3	1.37s	6.97s	X	1692.14
Example 4	X	X	X	17.42

Table 2: Comparison of Computation Times

the reachability analysis as soon as the safe set is entered. From the stability perspective, we use an unrolled version of the hybrid system's automaton – which is obtained by repeated reachset computations – to have a more precise characterization of the feasible trajectories. Obtaining Lyapunov functions for this unrolled hybrid automaton reduces the computational effort. It allows us to find LFs for systems where the system's representation contains implicit information that is needed to successfully prove stability.

In the future we plan to investigate sufficient and necessary termination criteria for the unrolling algorithm. Even if it turns out that termination of the unrolling algorithm will fail on a particular automaton, we believe that the presented techniques still can fruitfully be applied to parts of the automaton. Reachability analysis profits from safe sets even when the stability of the overall automaton cannot be established. Additionally, knowledge gained by partial reachability computations on sub-components of the automaton helps us to relax the transition constraints for the computation of the GLF. However, the usefulness of our approach has already been shown by some promising experiments (see Section 5).

7 Appendix: The Geometry behind the SafeBox Conversion

Given a Lyapunov function $V(\mathbf{x})$ with a symmetric matrix V and a level set $\mathcal{L}_{V,c^2} = \left\{\mathbf{x} \,\middle|\, V(\mathbf{x}) \leq c^2\right\} \subseteq \mathbb{R}^N$, this section describes how to compute an inner polyhedral approximation of the level set, see also [24].

By construction the Lyapunov function can be written as $V(\mathbf{x}) = \mathbf{x}^T Q \mathbf{x}$ with a symmetric matrix Q. Hence, the level set $\mathcal{L}_{V,c^2} = \left\{\mathbf{x} \,\middle|\, \mathbf{x}^T Q \mathbf{x} \leq c^2\right\}$ describes a quadric. Projective principal axis transformation provides a basis in which the quadric $\mathcal{L}_{V,c^2}$ has the shape of a hyperboloid

$$\mathbf{H} = \left\{\mathbf{x} \,\middle|\, \mathbf{x}^T \begin{pmatrix} \mathrm{I} & \mathrm{O} & \mathrm{O} \\ \mathrm{O} & \mathrm{O} & \mathrm{O} \\ \mathrm{O} & \mathrm{O} & -\mathrm{I} \end{pmatrix} \mathbf{x} \leq 1 \right\}.$$

By setting the negative entries on the diagonal to zero, we obtain the inscribed spherical cylinder

$$\mathbf{C} = \left\{\mathbf{x} \,\middle|\, \mathbf{x}^T \begin{pmatrix} \mathrm{I} & \mathrm{O} & \mathrm{O} \\ \mathrm{O} & \mathrm{O} & \mathrm{O} \\ \mathrm{O} & \mathrm{O} & \mathrm{O} \end{pmatrix} \mathbf{x} \leq 1 \right\} = \mathbf{S}^k \times \mathbb{R}^{N-k},$$

where $\mathbf{S}^k$ is the unit hyperball in $\mathbb{R}^k$. Now, the problem of finding an inner polyhedral approximation of $\mathcal{L}_{V,c^2}$ reduces to providing an inner polyhedral approximation

$\mathbf{P} \subseteq \mathbf{S}^k$ of the $(N-k)$-dimensional unit hyperball and transforming the resulting polyhedron $\mathbf{P} \times \mathbb{R}^k$ back to the initial basis.

The main difficulty of this procedure is to find an adequate representation of polyhedra in the projective space. In the following we introduce the needed geometrical concepts. The conceptions of projective spaces and of polyhedra are introduced for any vector space $\mathbb{K}^N$ over an ordered field $\mathbb{K}$. To capture the basis transformation that transforms the level set $\mathcal{L}_{V,c^2}$ to the hyperboloid $\mathbf{H}$, we need the additional postulate that any positive element has a square root, i. e., we migrate to the Euclidean vector space $\mathbb{E}^N$. A closed convex polyhedron $\mathbf{P}$ is the set $\mathbf{P} = \mathbf{P}(A, \mathbf{a}) = \{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{a}\}$. In the following the term *polyhedron* will always refer to a closed convex polyhedron. Vectors and sets of vectors are denoted by bold letters. All coefficients of the vectors $\mathbf{0}$ and $\mathbf{1}$ are 0 or 1, respectively. We use the superscript ^{T} to indicate the transpose of a vector or matrix. The notion A^{-T} denotes the transpose of the inverse of the matrix A. The coefficients of a N-dimensional vector $\mathbf{x}$ are denoted by $x_1, \ldots, x_N$.

7.1 Projective Space

We give a short introduction into projective geometry as it can be found in many textbooks like [20, 5]. The *projective space* $\mathrm{Proj}^N(\mathbb{K})$ *induced by* $\mathbb{K}$ can be identified with the set $\mathbb{K}^{N+1} \setminus \{\mathbf{0}\}$ where two vectors $\mathbf{x}$, $\mathbf{y}$ are *projectively equal* if and only if there exists some $\lambda \neq 0$ such that $\mathbf{x} = \lambda \mathbf{y}$, formally

$$\mathbf{x} =_p \mathbf{y} \iff \exists \lambda \neq 0\colon\ \mathbf{x} = \lambda \mathbf{y}. \tag{3}$$

The injective mapping $\iota\colon \mathbb{K}^N \to \mathrm{Proj}^N(\mathbb{K})$, $\mathbf{x} \mapsto \binom{\mathbf{x}}{1}$ defines the *canonical embedding* of $\mathbb{K}^N$ into the projective space $\mathrm{Proj}^N(\mathbb{K})$. On the other hand, any point $\binom{\mathbf{x}}{\lambda} \in \mathrm{Proj}^N(\mathbb{K})$ either represents the point $\frac{1}{\lambda}\mathbf{x}$ in $\mathbb{K}^N$ if and only if $\lambda \neq 0$, or $\binom{\mathbf{x}}{\lambda}$ represents a point at infinity, which is a point that has no corresponding point in $\mathbb{K}^N$. Actually, $\binom{\mathbf{x}}{0}$ represents $\lim_{\lambda \to 0^+} \frac{1}{\lambda}\mathbf{x}$, which coincides with $\lim_{\lambda \to 0^-} \frac{1}{\lambda}\mathbf{x}$ by projective equality.

7.2 Complete Projective Embeddings

The classical approach allows us to treat projective sets as subsets of $\mathbb{K}^{N+1} \setminus \{\mathbf{0}\}$, but it has a drawback: Due to the existential quantifier in (3) it is laborious to assess projective subset relations in $\mathbb{K}^{N+1} \setminus \{\mathbf{0}\}$. We shall introduce the notion of *projective completeness*, which allows us to assess projective subset relations in a simple way. The projective vector space $\mathrm{Proj}^N(\mathbb{K})$ is canonically embedded in the vector space $\mathbb{K}^{N+1} \setminus \{\mathbf{0}\}$ and, hence, also embedded in $\mathbb{K}^{N+1}$.

Definition 8. *Let* $\mathbf{P}$ *be a subset of* $\mathbb{K}^{N+1}$*. If (i)* $\mathbf{0} \in \mathbf{P}$ *and (ii)* $\mathbf{x} \in \mathbf{P}$ *implies that* $\lambda\mathbf{x} \in \mathbf{P}$ *for all* $\lambda \in \mathbb{K}$*, then we call* $\mathbf{P}$ projectively complete*.*

Definition 9. *Let* $\mathbf{P}$ *be a closed set, and let* $\tilde{\mathbf{P}}$ *be the least closed and projectively complete set with* $\iota(\mathbf{P}) \subseteq \tilde{\mathbf{P}}$*. The set* $\tilde{\mathbf{P}}$ *is called the* complete projective embedding *of* $\mathbf{P}$*.*

Hence, the projectively complete embedding $\tilde{\mathbf{P}}$ of $\mathbf{P}$ always contains $\mathbf{0}$ and all projective representations $\binom{\mathbf{x}}{\lambda}$, $\lambda \neq 0$, of a point $\mathbf{x} \in \mathbf{P}$. Furthermore, since the union of two closed sets $\mathbf{P}_1$ and $\mathbf{P}_2$ is closed again, the complete projective embedding of $\mathbf{P}_1 \cup \mathbf{P}_2$ is the union $\tilde{\mathbf{P}}_1 \cup \tilde{\mathbf{P}}_2$ of the complete projective embeddings $\tilde{\mathbf{P}}_1$ and $\tilde{\mathbf{P}}_2$.

Proposition 3. *Let* $\mathbf{P}$, $\mathbf{Q}$ *be two closed sets and* $\tilde{\mathbf{P}}$, $\tilde{\mathbf{Q}}$ *their complete projective embeddings. Then* $\mathbf{P} \subseteq \mathbf{Q}$ *if and only if* $\tilde{\mathbf{P}} \subseteq \tilde{\mathbf{Q}}$.

Proof. Assume $\mathbf{P} \subseteq \mathbf{Q}$ holds. Let $\binom{\mathbf{x}}{\lambda} \in \tilde{\mathbf{P}}$. In the case $\lambda \neq 0$ we have $\frac{1}{\lambda}\mathbf{x} \in \mathbf{P}$. Since $\mathbf{P} \subseteq \mathbf{Q}$, we also have $\frac{1}{\lambda}\mathbf{x} \in \mathbf{Q}$. The set $\tilde{\mathbf{Q}}$ is the complete projective embedding of $\mathbf{Q}$. Hence, $\binom{\mathbf{x}}{\lambda} \in \tilde{\mathbf{Q}}$. In the case $\lambda = 0$ we either have $\mathbf{x} = \mathbf{0}$ and $\binom{\mathbf{x}}{\lambda} = \binom{\mathbf{0}}{0}$ is trivially contained in $\tilde{\mathbf{Q}}$, or there exists a sequence $\left(\binom{\mathbf{x}_i}{\lambda_i}\right)_{i\in\mathbb{N}} \subseteq \tilde{\mathbf{P}}$ with $\lim_{i\to\infty} \binom{\mathbf{x}_i}{\lambda_i} = \binom{\mathbf{x}}{0}$ and $\lambda_i \neq 0$ for all $i \in \mathbb{N}$ since $\tilde{\mathbf{P}}$ is the least closed and projective complete set containing $\iota(\mathbf{P})$. We already have seen that any $\binom{\mathbf{x}_i}{\lambda_i} \in \tilde{\mathbf{P}}$ with $\lambda_i \neq 0$ is also in $\tilde{\mathbf{Q}}$. Furthermore, $\tilde{\mathbf{Q}}$ is closed. Hence, $\binom{\mathbf{x}}{0} \in \tilde{\mathbf{Q}}$. We have shown that $\mathbf{P} \subseteq \mathbf{Q}$ implies $\tilde{\mathbf{P}} \subseteq \tilde{\mathbf{Q}}$.

On the other hand, assume $\tilde{\mathbf{P}} \subseteq \tilde{\mathbf{Q}}$. Let $\mathbf{x} \in \mathbf{P}$. Then $\binom{\mathbf{x}}{1} \in \tilde{\mathbf{P}}$ and $\tilde{\mathbf{P}} \subseteq \tilde{\mathbf{Q}}$ implies $\binom{\mathbf{x}}{1} \in \tilde{\mathbf{Q}}$. Hence, $\mathbf{x} \in \mathbf{Q}$. We have shown that $\tilde{\mathbf{P}} \subseteq \tilde{\mathbf{Q}}$ implies $\mathbf{P} \subseteq \mathbf{Q}$. □

Proposition 4 (Complete Projective Embedding of a Polyhedron)**.** *Let* $\mathbf{P} = \mathbf{P}(A, \mathbf{a}) = \left\{\mathbf{x} \in \mathbb{K}^N \,\middle|\, A\mathbf{x} \leq \mathbf{a}\right\}$ *be a polyhedron, and let* $\tilde{\mathbf{P}} = \mathbf{P}_1 \cup \mathbf{P}_2$, *where*

$$\mathbf{P}_1 = \mathbf{P}\left(\begin{pmatrix} A & -\mathbf{a} \\ \mathbf{0}^T & -1 \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ 0 \end{pmatrix}\right),$$
$$\mathbf{P}_2 = \mathbf{P}\left(\begin{pmatrix} -A & \mathbf{a} \\ \mathbf{0}^T & 1 \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ 0 \end{pmatrix}\right).$$

Then $\tilde{\mathbf{P}}$ *is the complete projective embedding of* $\mathbf{P}$.

Proof. Obviously, we have $\iota(\mathbf{P}) \subseteq \mathbf{P}_1 \subseteq \tilde{\mathbf{P}}$.

We show that $\tilde{\mathbf{P}}$ is projectively complete. Obviously, $\binom{\mathbf{0}}{0} \in \tilde{\mathbf{P}}$. Let $\binom{\mathbf{x}}{\lambda} \in \tilde{\mathbf{P}}$, i. e., $A\mathbf{x} - \lambda\mathbf{a} \leq \mathbf{0}$, $-\lambda \leq 0$ or $-A\mathbf{x} + \lambda\mathbf{a} \leq \mathbf{0}$, $\lambda \leq 0$. Let $\mu \in \mathbb{K}$. Either we have $\mu \geq 0$ or $\mu < 0$. In both cases we obtain $A\mu\mathbf{x} - \mu\lambda\mathbf{a} \leq \mathbf{0}$, $-\mu\lambda \leq 0$ or $-A\mu\mathbf{x} + \mu\lambda\mathbf{a} \leq \mathbf{0}$, $\mu\lambda \leq 0$. Hence, $\binom{\mathbf{x}}{\lambda} \in \tilde{\mathbf{P}}$ implies $\mu\binom{\mathbf{x}}{\lambda} \in \tilde{\mathbf{P}}$ for all $\mu \in \mathbb{K}$.

We show that $\tilde{\mathbf{P}}$ is a closed set. Let $\left(\binom{\mathbf{x}_i}{\lambda_i}\right)_{i\in\mathbb{N}}$ be a sequence with $\binom{\mathbf{x}_i}{\lambda_i} \in \tilde{\mathbf{P}}$ for all $i \in \mathbb{N}$ and $\lim_{i\to\infty} \binom{\mathbf{x}_i}{\lambda_i} = \binom{\mathbf{x}}{\lambda}$. We have to show that $\binom{\mathbf{x}}{\lambda} \in \tilde{\mathbf{P}}$. In the case $\lambda \neq 0$ let J be the set of all indices i where $\lambda_i \neq 0$. Then $(\frac{1}{\lambda_i}\mathbf{x}_i)_{i\in J}$ is an infinite sequence of members in $\mathbf{P}$ which converges to $\frac{1}{\lambda}\mathbf{x}$. The polyhedron $\mathbf{P}$ is closed, hence $\frac{1}{\lambda}\mathbf{x} \in \mathbf{P}$ and $\binom{\mathbf{x}}{\lambda} \in \tilde{\mathbf{P}}$. Let $\lambda = 0$. Either we have an infinite subset $J \subseteq \mathbb{N}$ such that $\lambda_i = 0$ for all $i \in J$. Then $A\mathbf{x}_i \leq \mathbf{0}$ or $-A\mathbf{x}_i \leq \mathbf{0}$ for all $i \in J$. The sets $\{\mathbf{y} \mid A\mathbf{y} \leq \mathbf{0}\}$ and $\{\mathbf{y} \mid -A\mathbf{y} \leq \mathbf{0}\}$ are closed, and so is their union. It follows $A\mathbf{x} \leq \mathbf{0}$ or $-A\mathbf{x} \leq \mathbf{0}$ and by definition of $\tilde{\mathbf{P}}$ also $\binom{\mathbf{x}}{0} \in \tilde{\mathbf{P}}$. Otherwise, there exists an index i_0 with $\lambda_i \neq 0$ for all $i \geq i_0$. The remaining sequence $\left(\binom{\mathbf{x}_i}{\lambda_i}\right)_{i\geq i_0}$ has the same limes. Hence, without loss of generality, we may assume $\lambda_i \neq 0$ for all $i \in \mathbb{N}$. Setting $\mathbf{y}_i = \mathbf{x}_i - \frac{\lambda_i}{\lambda_0}\mathbf{x}_0$ yields the equalities $\binom{\mathbf{y}_i}{0} = \binom{\mathbf{x}_i}{\lambda_i} - \frac{\lambda_i}{\lambda_0}\binom{\mathbf{x}_0}{\lambda_0}$ for all $i \in \mathbb{N}$. Since $\lim_{i\to\infty} \lambda_i = 0$ and $\lim_{i\to\infty} \mathbf{x}_i = \mathbf{x}$, we obtain $\lim_{i\to\infty} \frac{\lambda_i}{\lambda_0}\mathbf{x}_0 = \mathbf{0}$ and $\lim_{i\to\infty} \binom{\mathbf{y}_i}{0} = \lim_{i\to\infty} \binom{\mathbf{x}_i}{\lambda_i} - \lim_{i\to\infty} \frac{\lambda_i}{\lambda_0}\binom{\mathbf{x}_0}{\lambda_0} = \binom{\mathbf{x}}{0}$. Hence, $\binom{\mathbf{y}_i}{0}$ is a sequence with $\lim_{i\to\infty} \binom{\mathbf{y}_i}{0} = \binom{\mathbf{x}}{0}$ where the last coefficient of all members is equal to zero. For this case we already have shown that $\binom{\mathbf{x}}{0} \in \tilde{\mathbf{P}}$.

It remains to show that $\tilde{\mathbf{P}}$ is the least closed set which includes $\iota(\mathbf{P})$ and is projectively complete. Assume, $\tilde{\mathbf{P}}'$ is an arbitrary closed set that includes $\iota(\mathbf{P})$ and is projectively complete. We have to show that any $\binom{\mathbf{x}}{\lambda} \in \tilde{\mathbf{P}}$ is also contained in $\tilde{\mathbf{P}}'$.

Therefore, let $\binom{\mathbf{x}}{\lambda} \in \tilde{\mathbf{P}}$. In the case $\lambda \neq 0$ we have $\frac{1}{\lambda}\mathbf{x} \in \mathbf{P}$. Then $\frac{1}{\lambda}\mathbf{x} \in \iota(\mathbf{P})$ and, hence, $\frac{1}{\lambda}\mathbf{x} \in \tilde{\mathbf{P}}'$. Since $\tilde{\mathbf{P}}'$ is projectively complete, it follows $\binom{\mathbf{x}}{\lambda} \in \tilde{\mathbf{P}}'$. In the case $\lambda = 0$ we have $A\mathbf{x} \leq \mathbf{0}$ or $-A\mathbf{x} \leq \mathbf{0}$. Hence, for all $\mathbf{y} \in \mathbf{P}$, i. e., $A\mathbf{y} \leq \mathbf{a}$, we have $\mathbf{y} + \mu\mathbf{x} \in \mathbf{P}$ for all $\mu \geq 0$ or $\mathbf{y} + \mu\mathbf{x} \in \mathbf{P}$ for all $\mu \leq 0$. The set $\tilde{\mathbf{P}}'$ includes $\iota(\mathbf{P})$, hence, $\binom{\mathbf{y}+\mu\mathbf{x}}{1} \in \tilde{\mathbf{P}}'$ for all $\mu \geq 0$ or $\binom{\mathbf{y}+\mu\mathbf{x}}{1} \in \tilde{\mathbf{P}}'$ for all $\mu \leq 0$. In the former case let $\mu_i = i$ and in the latter case let $\mu_i = -i$. Since $\tilde{\mathbf{P}}'$ is projectively complete, we have $\frac{1}{\mu_i}\binom{\mathbf{y}+\mu_i\mathbf{x}}{1} \in \tilde{\mathbf{P}}'$ for all $i \geq 1$. Hence, $\lim_{i\to\infty} \frac{1}{\mu_i}\binom{\mathbf{y}+\mu_i\mathbf{x}}{1} = \lim_{i\to\infty} \binom{\frac{1}{\mu_i}\mathbf{y}+\mathbf{x}}{\frac{1}{\mu_i}} = \binom{\mathbf{x}}{0}$. Since $\tilde{\mathbf{P}}'$ is closed, it follows $\binom{\mathbf{x}}{0} \in \tilde{\mathbf{P}}'$. □

Proposition 5. *Let $\tilde{\mathbf{P}}$ be the union of two cones*

$$\tilde{\mathbf{P}} = \mathbf{P}\big((A \quad -\mathbf{a}), \mathbf{0}\big) \cup \mathbf{P}\big((-A \quad \mathbf{a}), \mathbf{0}\big).$$

Then $\tilde{\mathbf{P}}$ is the complete projective embedding of $\mathbf{P}(A, \mathbf{a}) \cup \mathbf{P}(-A, -\mathbf{a})$.

Proof. Let $\tilde{\mathbf{R}}$ be the complete projective embedding of $\mathbf{P}(A, \mathbf{a}) \cup \mathbf{P}(-A, -\mathbf{a})$, i. e.,

$$\begin{aligned}\tilde{\mathbf{R}} = \mathbf{P}\left(\begin{pmatrix} A & -\mathbf{a} \\ \mathbf{0}^T & -1 \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ 0 \end{pmatrix}\right) \cup \mathbf{P}\left(\begin{pmatrix} -A & \mathbf{a} \\ \mathbf{0}^T & 1 \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ 0 \end{pmatrix}\right) \cup \\ \mathbf{P}\left(\begin{pmatrix} -A & \mathbf{a} \\ \mathbf{0}^T & -1 \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ 0 \end{pmatrix}\right) \cup \mathbf{P}\left(\begin{pmatrix} A & -\mathbf{a} \\ \mathbf{0}^T & 1 \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ 0 \end{pmatrix}\right) \\ = \mathbf{P}\big((A \quad -\mathbf{a}), \mathbf{0}\big) \cup \mathbf{P}\big((-A \quad \mathbf{a}), \mathbf{0}\big).\end{aligned}$$

Hence, $\tilde{\mathbf{R}} = \tilde{\mathbf{P}}$. □

Any complete projective embedding $\tilde{\mathbf{P}}$ of the form

$$\tilde{\mathbf{P}} = \mathbf{P}\big((A \quad -\mathbf{a}), \mathbf{0}\big) \cup \mathbf{P}\big((-A \quad \mathbf{a}), \mathbf{0}\big)$$

is a double cone. Gallier calls these double cones *projective polyhedra* [19]. In the following we shall make use of his nomenclature. Note that the transition from the projective polyhedron $\tilde{\mathbf{P}} = \mathbf{P}\big((A \quad -\mathbf{a}), \mathbf{0}\big) \cup \mathbf{P}\big((-A \quad \mathbf{a}), \mathbf{0}\big) \subseteq \mathbb{K}^{N+1}$ to $\mathbf{P}(A, \mathbf{a}) \cup \mathbf{P}(-A, -\mathbf{a}) \subseteq \mathbb{K}^N$ geometrically corresponds to the intersection of $\tilde{\mathbf{P}}$ with the hyperplane $\left\{\binom{\mathbf{x}}{\lambda} \,\middle|\, \lambda = 1\right\} \subseteq \mathbb{K}^{N+1}$.

Lemma 1. *Any linear map $\phi : \mathbb{K}^{N+1} \to \mathbb{K}^{N'+1}$ is compatible with projective equality $=_p$.*

Proof. Let $\mathbf{u}, \mathbf{v} \in \mathrm{Proj}^N(\mathbb{K})$ with $\mathbf{u} =_p \mathbf{v}$, i. e., there exists some $\lambda \neq 0$ with $\mathbf{u} = \lambda\mathbf{v}$. Then $\phi(\mathbf{u}) = \phi(\lambda\mathbf{v}) = \lambda\phi(\mathbf{v})$, hence $\phi(\mathbf{u}) =_p \phi(\mathbf{v})$. □

Definition 10. *Let $\phi : \mathbb{K}^{N+1} \to \mathbb{K}^{N+1}$ be a bijective linear map. Then ϕ is called a* projectivity.

Proposition 6. *The class of projective polyhedra is closed under projectivities. It is also closed under linear maps $\phi : \mathbb{K}^{N+1} \to \mathbb{K}^{N'+1}$.*

Proof. The proof is obvious since the image of a closed convex cone is a closed convex cone again. □

In case of a projectivity ϕ let M be its transformation matrix. Then we can explicitly state the following formula for the image of a projective polyhedron $\tilde{\mathbf{P}} = \mathbf{P}\big((A \quad -\mathbf{a}), \mathbf{0}\big) \cup \mathbf{P}\big((-A \quad \mathbf{a}), \mathbf{0}\big)$:

$$M\tilde{\mathbf{P}} = \mathbf{P}\big((A \quad -\mathbf{a})M^{-1}, \mathbf{0}\big) \cup \mathbf{P}\big((-A \quad \mathbf{a})M^{-1}, \mathbf{0}\big).$$

Notes on Projective Polyhedra. During our research we found the following interesting properties of projective polyhedra:

(i) The class of projective polyhedra is not closed under intersections [19].

(ii) There is a close relationship between projective polyhedra and the set of feasible solutions of the linear fractional program

$$\text{maximize } \frac{\mathbf{c}^T\mathbf{x} + \alpha}{\mathbf{d}^T\mathbf{x} + \beta} \text{ subject to } A\mathbf{x} \leq \mathbf{a}.$$

Under the regularity conditions that $\{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{a}\}$ is bounded and not empty, Charnes and Cooper [8] show that it suffices to solve the two linear programs

$$\begin{aligned}&\text{maximize } \mathbf{c}^T\mathbf{y} + \alpha\lambda \text{ subject to}\\ &\qquad A\mathbf{y} - \mathbf{a}\lambda \leq \mathbf{0}, \mathbf{d}^T\mathbf{y} + \beta\lambda = 1, \lambda \geq 0\\ &\text{maximize } \mathbf{c}^T\mathbf{y} + \alpha\lambda \text{ subject to}\\ &\qquad - A\mathbf{y} + \mathbf{a}\lambda \leq \mathbf{0}, \mathbf{d}^T\mathbf{y} + \beta\lambda = 1, \lambda \leq 0.\end{aligned}$$

It is not hard to see that the intersection of the hyperplane $\{\binom{\mathbf{x}}{\lambda} \mid \binom{\mathbf{d}}{\beta}^T \binom{\mathbf{x}}{\lambda} = 1\}$ with the projective embedding $\mathbf{P}\big((A, -\mathbf{a}), \mathbf{0}\big) \cup \mathbf{P}\big((-A, \mathbf{a}), \mathbf{0}\big)$ of $\mathbf{P}(A, \mathbf{a})$ corresponds to the set of feasible solutions of the linear programs above. Moreover, projective polyhedra allow us to drop the regularity conditions.

In the remainder of this section we return to general, non-polyhedral complete projective embeddings.

Proposition 7 (Complete Projective Embedding of a Quadric)**.** *Let* $\mathbf{Q}$ *be a quadric* $\mathbf{Q} = \{\mathbf{x} \mid \mathbf{x}^T Q \mathbf{x} \leq c^2\}$. *Further, let*

$$\tilde{\mathbf{Q}} = \left\{ \begin{pmatrix}\mathbf{x}\\ \lambda\end{pmatrix} \middle| \begin{pmatrix}\mathbf{x}\\ \lambda\end{pmatrix}^T \begin{pmatrix} Q & \mathbf{0}\\ \mathbf{0}^T & -c^2\end{pmatrix} \begin{pmatrix}\mathbf{x}\\ \lambda\end{pmatrix} \leq 0 \right\}.$$

Then $\tilde{\mathbf{Q}}$ *is the complete projective embedding of* $\mathbf{Q}$.

Proof. Obviously, $\tilde{\mathbf{Q}}$ includes $\iota(\mathbf{Q})$. Furthermore, $\binom{\mathbf{x}}{\lambda}^T \begin{pmatrix} Q & \mathbf{0}\\ \mathbf{0}^T & -c^2\end{pmatrix} \binom{\mathbf{x}}{\lambda} \leq 0$ implies $\mu\binom{\mathbf{x}}{\lambda}^T \begin{pmatrix} Q & \mathbf{0}\\ \mathbf{0}^T & -c^2\end{pmatrix} \mu\binom{\mathbf{x}}{\lambda} \leq 0$ for all $\mu \in \mathbb{K}$. Hence, $\tilde{\mathbf{Q}}$ is projectively complete.

We show that $\tilde{\mathbf{Q}}$ is closed. Let $\big(\binom{\mathbf{x}_i}{\lambda_i}\big)_{i\in\mathbb{N}}$ be a convergent sequence with $\binom{\mathbf{x}_i}{\lambda_i} \in \tilde{\mathbf{Q}}$ for all $i \in \mathbb{N}$ and $\lim_{i\to\infty} \binom{\mathbf{x}_i}{\lambda_i} = \binom{\mathbf{x}}{\lambda}$. Assume, $\lambda \neq 0$. Let J be the set of all indices where $\lambda_i \neq 0$. Then $\big(\binom{\frac{1}{\lambda_i}\mathbf{x}_i}{1}\big)_{i\in J}$ is an infinite sequence which converges to $\binom{\frac{1}{\lambda}\mathbf{x}}{1}$. Furthermore, for all $i \in J$ we have $\frac{1}{\lambda_i} \in \mathbf{Q}$. Since $\mathbf{Q}$ is closed, we have $\frac{1}{\lambda}\mathbf{x} \in \mathbf{Q}$.

Hence, $\binom{\frac{1}{\lambda}\mathbf{x}}{1} \in \iota(\mathbf{Q})$ and, by projective completeness, $\binom{\mathbf{x}}{\lambda} \in \tilde{\mathbf{Q}}$. Assume, $\lambda = 0$. Either we have an infinite subset $J \subseteq \mathbb{N}$ such that $\mathbf{x}_i^T Q \mathbf{x}_i \leq 0$ for all $i \in J$. The set $\{\mathbf{y} \mid \mathbf{y}^T Q \mathbf{y} \leq 0\} \subseteq \mathbf{Q}$ is closed. Hence, $\mathbf{x}^T Q \mathbf{x} \leq 0$ and $\binom{\mathbf{x}}{0}^T \begin{pmatrix} Q & \mathbf{0} \\ \mathbf{0}^T & -c^2 \end{pmatrix} \binom{\mathbf{x}}{0} \leq 0$. That is, $\binom{\mathbf{x}}{0} \in \tilde{\mathbf{Q}}$. Otherwise, there exists an index i_0 with $\lambda_i \neq 0$ for all $i \geq i_0$. The remaining infinite sequence $\left(\binom{\mathbf{x}_i}{\lambda_i}\right)_{i \geq i_0}$ has the same limes and without loss of generality we may assume that $i = 0$. Setting $\mathbf{y}_i = \mathbf{x}_i - \frac{\lambda_i}{\lambda_0}\mathbf{x}_0$ yields the equalities $\binom{\mathbf{y}_i}{0} = \binom{\mathbf{x}_i}{\lambda_i} - \frac{\lambda_i}{\lambda_0}\binom{\mathbf{x}_0}{\lambda_0}$ for all $i \in \mathbb{N}$. Since $\lim_{i\to\infty} \lambda_i = 0$ and $\lim_{i\to\infty} \mathbf{x}_i = \mathbf{x}$, we obtain $\lim_{i\to\infty} \frac{\lambda_i}{\lambda_0}\mathbf{x}_0 = \mathbf{0}$ and $\lim_{i\to\infty} \binom{\mathbf{y}_i}{0} = \lim_{i\to\infty} \binom{\mathbf{x}_i}{\lambda_i} - \lim_{i\to\infty} \frac{\lambda_i}{\lambda_0}\binom{\mathbf{x}_0}{\lambda_0} = \binom{\mathbf{x}}{0}$. Hence, $\binom{\mathbf{y}_i}{0}$ is a sequence with $\lim_{i\to\infty} \binom{\mathbf{y}_i}{0} = \binom{\mathbf{x}}{\lambda}$ where the last coefficient of the members is equal to zero. For this case we already have shown that $\binom{\mathbf{x}}{0} \in \tilde{\mathbf{Q}}$.

It remains to show that $\tilde{\mathbf{Q}}$ is the least closed projectively complete set which contains $\iota(\mathbf{Q})$. Assume $\tilde{\mathbf{Q}}'$ is another closed projectively complete set that contains $\iota(\mathbf{Q})$. We have to show that any $\binom{\mathbf{x}}{\lambda} \in \tilde{\mathbf{Q}}$ is also in $\tilde{\mathbf{Q}}'$. Therefore, let $\binom{\mathbf{x}}{\lambda} \in \tilde{\mathbf{Q}}$, i.e., $\mathbf{x}^T Q \mathbf{x} \leq \lambda^2 c^2$. In the case $\lambda \neq 0$ we have $\frac{1}{\lambda}\mathbf{x} \in \mathbf{Q}$ and, hence $\binom{\frac{1}{\lambda}\mathbf{x}}{1} \in \iota(\mathbf{Q})$. The set $\tilde{\mathbf{Q}}'$ includes $\iota(\mathbf{Q})$ and is projectively complete. Hence, $\binom{\mathbf{x}}{\lambda} \in \tilde{\mathbf{Q}}'$. Otherwise, we have $\lambda = 0$, i.e., $\mathbf{x}^T Q \mathbf{x} \leq 0$. Then for any μ we have $\mu\mathbf{x}^T Q \mu\mathbf{x} \leq 0$. Let $\mu_i = i$. Since $0 \leq c^2$, we have $\mu_i \mathbf{x} \in \mathbf{Q}$ for all $i \geq 1$. Further, since $\iota(\mathbf{Q}) \in \tilde{\mathbf{Q}}'$ and $\tilde{\mathbf{Q}}'$ is projectively complete, we also have $\frac{1}{\mu_i}\binom{\mu_i \mathbf{x}}{1} = \binom{\mathbf{x}}{\frac{1}{\mu_i}} \in \tilde{\mathbf{Q}}'$. We obtain $\lim_{i\to\infty} \binom{\mathbf{x}}{\frac{1}{\mu_i}} = \binom{\mathbf{x}}{0} \in \mathbb{K}^{N+1}$. Since $\tilde{\mathbf{Q}}'$ is closed, it follows $\binom{\mathbf{x}}{\lambda} = \binom{\mathbf{x}}{0} \in \tilde{\mathbf{Q}}'$. □

7.3 Inner Approximation of a Quadric

In the following we show how to find an inner polyhedral approximation of a quadric $\mathbf{Q} = \{\mathbf{x} \mid \mathbf{x}^T Q \mathbf{x} \leq c^2\} \subseteq \mathbb{E}^N$. Without loss of generality we may assume that Q is a symmetric matrix.[7] We will show that it suffices to provide an inscribed polyhedron of the unit sphere in any dimension to find an inner approximation of the quadric. For example, for any dimension N the polyhedron

$$\mathbf{B}(N) = \mathbf{P}\left(\left(\begin{smallmatrix} \mathrm{I} \\ -\mathrm{I} \end{smallmatrix}\right), \tfrac{1}{\sqrt{N}}\left(\begin{smallmatrix} \mathbf{1} \\ \mathbf{1} \end{smallmatrix}\right)\right) = \left\{\mathbf{x} \,\middle|\, \mathbf{x} \leq \tfrac{1}{\sqrt{N}}\mathbf{1}, -\mathbf{x} \leq \tfrac{1}{\sqrt{N}}\mathbf{1}\right\}$$

is an inscribed hypercube of the hyperball

$$\mathbf{S}^N = \{\mathbf{x} \mid \mathbf{x}^T\mathbf{x} \leq 1\} = \{\mathbf{x} \mid x_1^2 + x_2^2 + \cdots + x_N^2 \leq 1\}.$$

Proposition 8 (Principal Axis Transformation). *Let* $\mathbf{Q} = \{\mathbf{x} \mid \mathbf{x}^T Q \mathbf{x} \leq c^2\}$ *be a quadric in* $\mathbb{E}^N$ *where* Q *is a symmetric matrix. Further, let*

$$\tilde{\mathbf{Q}} = \left\{\begin{pmatrix}\mathbf{x}\\ \lambda\end{pmatrix} \,\middle|\, \begin{pmatrix}\mathbf{x}\\ \lambda\end{pmatrix}^T \tilde{Q} \begin{pmatrix}\mathbf{x}\\ \lambda\end{pmatrix} \leq 0\right\} \text{with } \tilde{Q} = \begin{pmatrix} Q & \mathbf{0} \\ \mathbf{0}^T & -c^2 \end{pmatrix}$$

be the complete projective embedding of $\mathbf{Q}$ *in* $\mathbb{E}^{N+1}$*. Then there exists an invertible matrix* L *and a diagonal matrix*

$$E = \begin{pmatrix} \mathrm{I} & \mathrm{O} & \mathrm{O} \\ \mathrm{O} & \mathrm{O} & \mathrm{O} \\ \mathrm{O} & \mathrm{O} & -\mathrm{I} \end{pmatrix} \quad \textit{such that} \quad \tilde{Q} = LEL^T.$$

[7]Otherwise, let $\hat{Q} = \frac{1}{2}(Q + Q^T)$, i.e., $\mathbf{x}^T\hat{Q}\mathbf{x} = \frac{1}{2}(\mathbf{x}^T Q\mathbf{x} + \mathbf{x}^T Q^T \mathbf{x}) = \frac{1}{2}(\mathbf{x}^T Q \mathbf{x} + (\mathbf{x}^T Q^T \mathbf{x})^T) = \frac{1}{2}(\mathbf{x}^T Q\mathbf{x} + \mathbf{x}^T Q \mathbf{x}) = \mathbf{x}^T Q \mathbf{x}$ for all $\mathbf{x}$, and use $\hat{Q}$ instead of Q.

While the matrix L is in general not unique, the matrix E is uniquely determined by Sylvester's law of inertia. If Q is positive semidefinite, then E has exactly one negative entry.

Proof. Symmetric Gaussian elimination yields an invertible diagonal matrix $D = \operatorname{diag}(d_1, \ldots, d_{N+1}) = U\tilde{Q}U^T$ where U represents the row operations and the transposed matrix U^T represents the corresponding column operations. We normalize the elements of D by multiplying with the matrix $S = \operatorname{diag}(s_1, \ldots, s_{N+1}) = S^T$, where each s_i is defined as $s_i = 1$ if $d_i = 0$, or $s_i = \frac{1}{\sqrt{|d_i|}}$ if $d_i \neq 0$. Then $SDS^T = SU\tilde{Q}U^TS^T$ is a diagonal matrix whose entries are 1, -1 or 0. Finally, we sort the entries of the diagonal matrix SDS^T in descending order, yielding a row permutation matrix P and a corresponding column permutation matrix P^T and $E = \left(\begin{smallmatrix} \mathrm{I} & \mathrm{O} & \mathrm{O} \\ \mathrm{O} & \mathrm{O} & \mathrm{O} \\ \mathrm{O} & \mathrm{O} & -\mathrm{I} \end{smallmatrix}\right) = PSDS^TP^T = PSU\tilde{Q}U^TS^TP^T$. Since P, S, and U are invertible, we define $L = (PSU)^{-1}$ and obtain the factorization $\tilde{Q} = LEL^T$. □

Note that the decomposition $\tilde{Q} = LEL^T$ as given above is only possible in a projective vector space. In a non-projective setting we would obtain a richer variety of resulting forms.

Let $\mathbf{Q}$, $\tilde{\mathbf{Q}}$, Q, $\tilde{Q}$, and L, E be given as in Proposition 8. We define $\binom{\mathbf{y}}{\mu} = L^T\binom{\mathbf{x}}{\lambda}$ and obtain the identity $\mathbf{x}^TQ\mathbf{x} - \lambda^2c^2 = \binom{\mathbf{x}}{\lambda}^T\tilde{Q}\binom{\mathbf{x}}{\lambda} = \binom{\mathbf{x}}{\lambda}^TLEL^T\binom{\mathbf{x}}{\lambda} = \binom{\mathbf{y}}{\mu}^TE\binom{\mathbf{y}}{\mu}$. Hence, the base transformation matrix L transforms the projective quadric $\tilde{\mathbf{Q}}$ into a projective hyperboloid of the form $\tilde{\mathbf{H}} = \left\{\binom{\mathbf{y}}{\mu} \,\middle|\, \binom{\mathbf{y}}{\mu}^TE\binom{\mathbf{y}}{\mu}\right\}$.

We explain how to find an inner approximation of the projective hyperboloid $\tilde{\mathbf{H}}$. Application of the inverse base transformation finally yields (a union of two) inscribed polyhedra of $\mathbf{Q}$. Let k the number of 1s, l the numbers of 0s, and m the number of -1s in E, with $k+l+m = N+1$. According to Sylvester's law of inertia, k, l, and m are uniquely determined by Q.

We characterize the solutions $\binom{\mathbf{y}}{\mu}$ of $\binom{\mathbf{y}}{\mu}^TE\binom{\mathbf{y}}{\mu} \leq 0$, or equivalently:

$$y_1^2 + \cdots + y_k^2 - y_{k+l+1}^2 - \cdots - y_N^2 - \mu^2 \leq 0. \tag{4}$$

We have the following mutual exclusive cases:

(i) If $k = 0$, then $\tilde{\mathbf{H}} = \mathbb{E}^{N+1}$, i. e., $\tilde{\mathbf{H}}$ is the complete projective embedding of the entire vector space $\mathbb{E}^N$.

(ii) If $k = N+1$, then $\tilde{\mathbf{H}} = \{\mathbf{0}\}$, i. e., $\tilde{\mathbf{H}}$ is the complete projective embedding of the empty set.

(iii) If $0 < k < N+1$ and $m = 0$, then $\tilde{\mathbf{H}} = \{0\}^k \times \mathbb{E}^{N-k} \times \mathbb{E}$, i. e., $\tilde{\mathbf{H}}$ is the complete projective embedding of the subspace $\{0\}^k \times \mathbb{E}^{N-k}$.

(iv) If $0 < k < N+1$ and $m = 1$, then $\tilde{\mathbf{H}} = \{\binom{\mathbf{y}}{\mu} \mid \mathbf{y}^T\left(\begin{smallmatrix} \mathrm{I} & \mathrm{O} \\ \mathrm{O} & \mathrm{O} \end{smallmatrix}\right)\mathbf{y} \leq \mu^2\}$ is the complete projective embedding of the cylinder $\mathbf{S}^k \times \mathbb{E}^{N-k}$. Let $\mathbf{T}(k) = \mathbf{P}(A, \mathbf{a})$ be an inscribed polyhedron of $\mathbf{S}^k$. The cylindrification $\mathbf{T}(k) \times \mathbb{E}^{N-k}$ of $\mathbf{T}(k)$ is given by $\mathbf{P} = \mathbf{P}\big(\begin{pmatrix} A & \mathrm{O} \end{pmatrix}, \mathbf{a}\big)$. We have $\mathbf{P} \subseteq \mathbf{S}^k \times \mathbb{E}^{N-k}$. According to Proposition 3 the subset relation carries over to the complete projective embeddings

$\tilde{\mathbf{P}} \subseteq \tilde{\mathbf{H}}$, where $\tilde{\mathbf{P}} = \mathbf{P}\Big(\big(\begin{smallmatrix} A & \mathrm{O} & -\mathbf{a} \\ \mathbf{0}^T & \mathbf{0}^T & -1 \end{smallmatrix}\big), \big(\begin{smallmatrix} \mathbf{0} \\ 0 \end{smallmatrix}\big)\Big) \cup \mathbf{P}\Big(\big(\begin{smallmatrix} -A & \mathrm{O} & \mathbf{a} \\ \mathbf{0}^T & \mathbf{0}^T & 1 \end{smallmatrix}\big), \big(\begin{smallmatrix} \mathbf{0} \\ 0 \end{smallmatrix}\big)\Big)$. Hence, after application of the base transformation L^{-T} on $\tilde{\mathbf{P}}$ we obtain a union of polyhedra $L^T(\tilde{\mathbf{P}}) = \mathbf{P}\big((A_1 \quad -\mathbf{a}_1), \mathbf{0}\big) \cup \mathbf{P}\big((-A_1 \quad \mathbf{a}_1), \mathbf{0}\big)$ with $L^T(\tilde{\mathbf{P}}) \subseteq \tilde{\mathbf{Q}}$ and the matrices are given by $\big(A_1 \quad -\mathbf{a}_1\big) = \Big(\begin{smallmatrix} A & \mathrm{O} & -\mathbf{a} \\ \mathbf{0}^T & \mathbf{0}^T & -1 \end{smallmatrix}\Big) L^T$ and $\big(-A_1 \quad \mathbf{a}_1\big) = \Big(\begin{smallmatrix} -A & \mathrm{O} & \mathbf{a} \\ \mathbf{0}^T & \mathbf{0}^T & 1 \end{smallmatrix}\Big) L^T$. According to Proposition 5, $L^T(\tilde{\mathbf{P}})$ is the complete projective embedding of $\mathbf{R} = \mathbf{P}(A_1, \mathbf{a}_1) \cup \mathbf{P}(-A_1, -\mathbf{a}_1)$. Furthermore, by Proposition 3 we have $\mathbf{R} \subseteq \mathbf{Q}$.

(v) Otherwise, we have $0 < k < N+1$ and $m > 1$. Now, let E' be the matrix which is obtained by replacing all but the last occurrence of -1 in E by 0, i. e., $E' = \mathrm{diag}(1, \ldots, 1, 0, \ldots, 0, -1)$. Any solution of $\binom{\mathbf{y}}{\mu}^T E' \binom{\mathbf{y}}{\mu} \leq 0$ is also a solution of (4), since for all $y_1, \ldots, y_N$ it holds $y_1^2 + \cdots + y_k^2 \geq y_1^2 + \cdots + y_k^2 - y_{k+l}^2 - \cdots - y_N^2$. Hence, $\tilde{\mathbf{H}}' = \Big\{ \binom{\mathbf{y}}{\mu} \,\Big|\, \binom{\mathbf{y}}{\mu}^T E' \binom{\mathbf{y}}{\mu} \leq 0 \Big\}$ is the complete projective embedding of the cylinder $\mathbf{S}^k \times \mathbb{E}^{N-k}$ and $\tilde{\mathbf{H}}' \subseteq \tilde{\mathbf{H}}$. Now, we proceed like in the previous item, where we use E' and $\tilde{\mathbf{H}}'$ instead of E and $\tilde{\mathbf{H}}$.

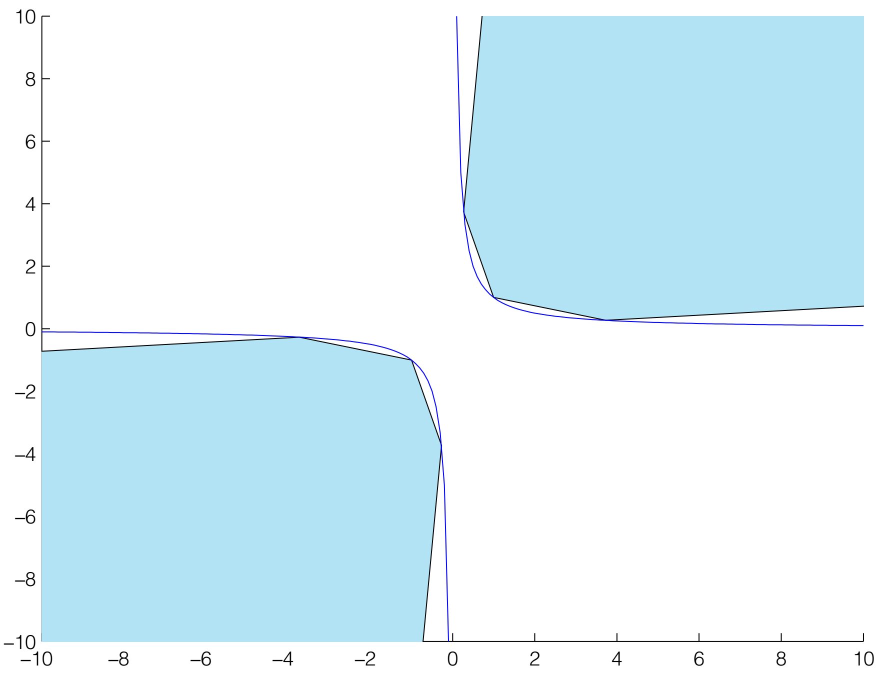


Figure 6: Hexagon Inscribed in Hyperbola $xy \geq 1$

7.4 Generating Polytopes With Circumsphere

In this section we discuss the problem of generating polytopes with circumsphere of arbitrary dimensions.

Obviously, the convex hull of several sampling points $\mathbf{v}_1, \ldots, \mathbf{v}_n$ on the boundary of the N-dimensional hyperball $\mathbf{S}$ results in an inscribed $\mathcal{V}$-polytope $\mathbf{P}$. Hence, to obtain the corresponding $\mathcal{H}$-representation, we had to perform a costly facet-enumeration and lose control on the number of facets of the resulting $\mathcal{H}$-representation. Instead, we are interested in methods which allow us to generate inscribed $\mathcal{H}$-polytopes directly. We abandon the idea of taking samples and provide a more regular way to generate inscribed $\mathcal{H}$-polyhedra. An interesting class of polytopes with circumsphere is the class of uniform polytopes [11, 10] which includes the class of regular polytopes. A complete enumeration of uniform polyhedra is only known in low dimensions. In two dimensions the uniform polytopes are the infinitely many regular polygons. In three dimensions the uniform polytopes cover the five Platonic solids, the 13 Archimedean solids, and the infinite set of prisms and antiprisms. In all dimensions the class of uniform polytopes contains the regular simplex, the hypercube, and the cross polytope. A N-dimensional simplex has only $N+1$ facets and vertices, which is certainly insufficient to provide a good inner approximation of a sphere. The N-dimensional cross polytope has 2^N facets and $2N$ vertices. Hence, the resulting $\mathcal{H}$-polytopes are only feasible in lower dimensions. The dual of a N-dimensional cross polytope is the N-dimensional hypercube having $2N$ facets and 2^N vertices. Apparently, the hypercube is well suited for computational purposes, but still may have too less facets to provide a good inner approximation of the sphere.

Hence, it is desirable to have a regular method which allows us to generate a richer variety of uniform polytopes. The next proposition provides such a method.

Proposition 9. *Given a polytope $\mathbf{P}_1 = \mathbf{P}(A_1, \mathbf{a}_1) \in \mathbb{K}^{N_1}$ with unit circumsphere $\mathbf{S}^{N_1}$ and a polytope $\mathbf{P}_2 = \mathbf{P}(A_2, \mathbf{a}_2) \in \mathbb{K}^{N_2}$ with unit circumsphere $\mathbf{S}^{N_2}$. Then for any $\alpha > 0$, $\beta > 0$ with $\alpha^2 + \beta^2 = 1$ the weighted cross product*

$$\begin{aligned}\mathbf{P} = \alpha\mathbf{P}_1 \times \beta\mathbf{P}_2 &= \{\left(\begin{smallmatrix}\mathbf{x}\\ \mathbf{y}\end{smallmatrix}\right) \mid \mathbf{x} \in \alpha\mathbf{P}_1,\, \mathbf{y} \in \beta\mathbf{P}_2\} \\ &= \mathbf{P}\Big(\Big(\begin{smallmatrix}A_1 & \mathrm{O}\\ \mathrm{O} & A_2\end{smallmatrix}\Big), \left(\begin{smallmatrix}\alpha\mathbf{a}_1\\ \beta\mathbf{a}_2\end{smallmatrix}\right)\Big)\end{aligned}$$

is a polytope in $\mathbb{K}^{N_1+N_2}$ with unit circumsphere $\mathbf{S}^{N_1+N_2}$. Furthermore, let f_1, f_2 be the number of facets of $\mathbf{P}_1$ and $\mathbf{P}_2$, and let v_1, v_2 be the number of vertices of $\mathbf{P}_1$ and $\mathbf{P}_2$. Then $\mathbf{P}$ has $f_1 + f_2$ facets and $v_1 v_2$ vertices.

Proof. The statement on the number of vertices and facets is rather obvious and belongs to mathematical folklore. We show that $\mathbf{P}$ has the circumsphere $\mathbf{S}^{N_1+N_2}$. That is, for any vertex $\left(\begin{smallmatrix}\alpha\mathbf{x}_i\\ \beta\mathbf{y}_j\end{smallmatrix}\right)$, $i = 1, \ldots, v_1$, $j = 1, \ldots, v_2$ of $\mathbf{P}$ we have $\left|\left(\begin{smallmatrix}\alpha\mathbf{x}_i\\ \beta\mathbf{y}_j\end{smallmatrix}\right)\right| = \left(\begin{smallmatrix}\alpha\mathbf{x}_i\\ \beta\mathbf{y}_j\end{smallmatrix}\right)^T \left(\begin{smallmatrix}\alpha\mathbf{x}_i\\ \beta\mathbf{y}_j\end{smallmatrix}\right) = \alpha^2\mathbf{x}_i^T\mathbf{x}_i + \beta^2\mathbf{y}_j^T\mathbf{y}_j = \alpha^2 + \beta^2 = 1$. □

Clearly, the previous proposition can be generalized to the weighted cross product of finitely many circumscribed polytopes. For example, the N-dimensional hypercube is the N-fold weighted cross product of the line segments $[-1, 1]$ with weight $\alpha = \frac{1}{\sqrt{N}}$.

However, in our experiments we restricted ourselves to only two different types of templates for the inner approximation of a sphere:

- The N-dimensional hypercube given by

$$\mathbf{P}\Big(\begin{pmatrix}\mathrm{I}\\ -\mathrm{I}\end{pmatrix}, \frac{1}{\sqrt{N}}\mathbf{1}\Big) = \big\{\mathbf{x} = (x_1, \ldots, x_N)^T \mid -1 \le x_i \le 1,\, i = 1, \ldots, N\big\}$$

- and the N-dimensional cross-polytope given by $\mathbf{P}(A, \mathbf{1})$, where all rows of the $2^N \times N$-matrix A are different and each row is a combination of the entries 1 and -1.

References

[1] A. Abate, M. Prandini, J. Lygeros, and S. Sastry. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, 44(11):2724–2734, 2008.

[2] M. Althoff and B. H. Krogh. Avoiding geometric intersection operations in reachability analysis of hybrid systems. In *HSCC*, pages 45–54. ACM, 2012.

[3] E. Asarin, T. Dang, and A. Girard. Reachability analysis of nonlinear systems using conservative approximation. In *HSCC*, pages 20–35. Springer, 2003.

[4] E. Asarin, T. Dang, and A. Girard. Hybridization methods for the analysis of nonlinear systems. *Acta Informatica*, 43(7):451–476, 2007.

[5] M. Berger. *Geometry I*, volume 1. Springer, 1987.

[6] B. Borchers. CSDP, a c library for semidefinite programming. *Optim. Met. Softw.*, 10:613–623, 1999.

[7] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge Uni. Press, 2004.

[8] A. Charnes and W. W. Cooper. Programming with linear fractional functionals. *Naval Research logistics quarterly*, 9(3-4):181–186, 1962.

[9] A. Chutinan and B. Krogh. Computational techniques for hybrid system verification. *Automatic Control, IEEE Transactions on*, 48(1):64–75, 2003.

[10] H. S. M. Coxeter. *Regular polytopes.* Methuen, London, 1948.

[11] H. S. M. Coxeter, M. S. Longuet-Higgins, and J. Miller. Uniform polyhedra. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 246(916):401–450, 1954.

[12] W. Damm, H. Dierks, J. Oehlerking, and A. Pnueli. Towards component based design of hybrid systems: Safety and stability. In *Essays in Memory of Amir Pnueli*, pages 96–143, 2010.

[13] W. Damm, W. Hagemann, E. Möhlmann, and A. Rakow. Component based design of hybrid systems: A case study on concurrency and coupling. Technical Report 95, SFB/TR 14 AVACS, 2014.

[14] W. Damm, E. Möhlmann, and A. Rakow. Component based design of hybrid systems: A case study on concurrency and coupling. In *HSCC*, pages 145–150. ACM, 2014.

[15] T. Dang, O. Maler, and R. Testylier. Accurate hybridization of nonlinear systems. In *HSCC*, pages 11–20. ACM, 2010.

[16] P. S. Duggirala and S. Mitra. Lyapunov abstractions for inevitability of hybrid systems. In *HSCC*, pages 115–124. ACM, 2012.

[17] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. SpaceEx: Scalable verification of hybrid systems. In *CAV*, volume 6806 of *LNCS*, pages 379–395. Springer, 2011.

[18] K. Fujisawa, K. Nakata, M. Yamashita, and M. Fukuda. SDPA project: Solving large-scale semidefinite programs. *JORSP*, 50(4):278–298, 2007.

[19] J. Gallier. Notes on convex sets, polytopes, polyhedra, combinatorial topology, Voronoi diagrams and Delaunay triangulations. Technical Report 650, University of Pennsylvania Department of Computer and Information Science, 2009.

[20] J. Gallier. *Geometric methods and applications: for computer science and engineering*, volume 38. Springer, 2011.

[21] A. Girard. Reachability of uncertain linear systems using zonotopes. In *HSCC*, volume 3414 of *LNCS*, pages 291–305. Springer, 2005.

[22] A. Girard and C. Le Guernic. Zonotope/hyperplane intersection for hybrid systems reachability analysis. In *HSCC*, pages 215–228. Springer, 2008.

[23] W. Hagemann. Reachability analysis of hybrid systems using symbolic orthogonal projections. In *CAV*, volume 8559 of *LNCS*, pages 406–422. Springer, 2014.

[24] W. Hagemann and E. Möhlmann. Inscribing $\mathcal{H}$-polyhedra in quadrics using a projective generalization of closed sets. In *CCCG*, 2015. To appear.

[25] W. Hagemann, E. Möhlmann, and A. Rakow. Verifying a PI controller using SoapBox and Stabhyli: Experiences on establishing properties for a steering controller. In *ARCH*, 2014.

[26] A. B. Kurzhanski and P. Varaiya. Ellipsoidal techniques for reachability analysis. In *HSCC*, volume 1790 of *LNCS*, pages 202–214. Springer, 2000.

[27] C. Le Guernic and A. Girard. Reachability analysis of hybrid systems using support functions. In *CAV*, volume 5643 of *LNCS*, pages 540–554. Springer, 2009.

[28] J. Löfberg. YALMIP: A toolbox for modeling and optimization in MATLAB. In *CACSD*, Taipei, Taiwan, 2004.

[29] A. M. Lyapunov. Problème général de la stabilité du movement. In *Ann. Fac. Sci. Toulouse, 9*, pages 203–474. Université Paul Sabatier, 1907.

[30] I. Mitchell and C. Tomlin. Level set methods for computation in hybrid systems. In *HSCC*, pages 310–323, 2000.

[31] E. Möhlmann and O. E. Theel. Stabhyli: A tool for automatic stability verification of non-linear hybrid systems. In *HSCC*, pages 107–112. ACM, 2013.

[32] J. Oehlerking. *Decomposition of Stability Proofs for Hybrid Systems*. PhD thesis, University of Oldenburg, Dept. of Computer Science, Oldenburg, Germany, 2011.

[33] J. Oehlerking, H. Burchardt, and O. E. Theel. Fully automated stability verification for piecewise affine systems. In *HSCC*, pages 741–745, 2007.

[34] J. Oehlerking and O. E. Theel. Decompositional construction of Lyapunov functions for hybrid systems. In *HSCC*, pages 276–290, 2009.

[35] A. Papachristodoulou, J. Anderson, G. Valmorbida, S. Prajna, P. Seiler, and P. A. Parrilo. *SOSTOOLS: Sum-of-Squares Optimization Toolbox for MATLAB*. `http://arxiv.org/abs/1310.4716`, 2013.

[36] A. Podelski and S. Wagner. Region stability proofs for hybrid systems. In *FORMATS*, pages 320–335, 2007.

[37] P. Prabhakar. Foundations for approximation based analysis of stability properties of hybrid systems. In *ACCCC*, pages 1602–1609, 2012.

[38] P. Prabhakar, G. E. Dullerud, and M. Viswanathan. Pre-orders for reasoning about stability. In *HSCC*, pages 197–206, 2012.

[39] P. Prabhakar, J. Liu, and R. M. Murray. Pre-orders for reasoning about stability properties with respect to input of hybrid systems. In *EMSOFT*, pages 1–10, 2013.

[40] P. Prabhakar and M. G. Soto. Abstraction based model-checking of stability of hybrid systems. In *CAV*, pages 280–295, 2013.

[41] S. Prajna and A. Papachristodoulou. Analysis of switched and hybrid systems - beyond piecewise quadratic methods. In *ACC*, volume 4, pages 2779–2784, 2003.

[42] S. Ratschan and Z. She. Providing a basin of attraction to a target region of polynomial systems by computation of Lyapunov-like functions. *SIAM J. Control and Optimization*, 48(7):4377–4394, 2010.

[43] S. Sankaranarayanan, T. Dang, and F. Ivančić. Symbolic model checking of hybrid systems using template polyhedra. In *TACAS*, pages 188–202. Springer, 2008.