Reachability of Hybrid Systems in Space-Time

Goran Frehse
Univ. Grenoble Alpes, VERIMAG, F-38000 Grenoble, France
CNRS, VERIMAG, F-38000 Grenoble, France
goran.frehse@imag.fr

ABSTRACT

In set-based reachability, a cover of the reachable states of a hybrid system is obtained by repeatedly computing one-step successor states. It can be used to show safety or to obtain quantitative information, e.g., for measuring the jitter in an oscillator circuit. In general, one-step successors can only be computed approximately and are difficult to scale in the number of continuous variables. The approximation error requires particular attention since it can accumulate rapidly, leading to a coarse cover, prohibitive state explosion, or preventing termination. In this paper, we propose an approach with precise control over the balance between approximation error and scalability. By lazy evaluation of set representations, the precision can be increased in a targeted manner, e.g., to show that a particular transition is spurious. Each evaluation step scales well in the number of continuous variables. The set representations are particularly suited for clustering and containment checking, which are essential for reducing the state explosion. This provides the building blocks for refining the cover of the reachable set just enough to show a property of interest. The approach is illustrated on several examples.

CCS Concepts

•Computing methodologies → Model verification and validation;

Keywords

Hybrid systems, verification, reachability, tools

1. INTRODUCTION

Hybrid systems describe the change of a set of continuous-valued variables combined with discrete states. In continuous time, the change is governed by a set of ordinary differential equations (ODEs) or, more generally, inclusions. Jumps of the discrete state can modify the ODEs or the values of the variables, and such changes can be state-dependent and nondeterministic. Such systems are difficult to analyze, even numerically simulate, as neighboring states, no matter how close, may exhibit qualitatively different behaviors. Conventional techniques from model-based design, such as simulation of corner cases or stochastic simulation, may fail to detect critical behavior. Reachability analysis exhaustively computes a cover of all behaviors and, if precise enough, can show safety of the system as well as provide quantitative measurements of key variables. Hybrid automata are an extension of finite state machines with continuous variables and ODEs. Complex models are readily constructed by composing automata that interact by sharing variables and synchronizing events. Hybrid and cyber-physical systems in a wide range of application domains have been analyzed through reachability analysis, e.g., automotive control [14], robotics [28], electronic circuits [16], and systems biology [8].

The set of solutions of ODEs is difficult to compute and generally has no closed-form representation. Computing a cover of the ODEs, called *flowpipe approximation*, is necessarily done approximately, and balancing accuracy against computational cost is nontrivial. A scalable algorithm has been presented in [23], but as in all related approaches, increasing the accuracy also increases the number of sets in the cover. A way to cluster sets optimally, without compromising the approximation error, was proposed in [15]. A key element of that approach is adding time as an additional state variable, and operating on sets in this augmented space, which we refer to as *space-time*. In addition to flowpipe approximation, this involves computing jump successors, containment checking, and emptiness checking. In this paper, we present a reachability algorithm based on flowpipe approximation in space-time, which combines previously published work on flowpipe approximation [17], jump successors [18, 13], clustering [15], and emptiness checking [13]. It completes the picture with containment checking and other details, tying those techniques together in a fixed-point algorithm. The approach is implemented in the *STC scenario* on the SpaceEx tool platform. The tool and examples are available for download [11].

Scalable flowpipe approximation algorithms for affine continuous systems are known for several set representations, such as ellipsoids [22]. However, reachability with ellipsoids does not easily extend to hybrid systems because ellipsoids are not closed under important set operations such as Minkowski sum, convex hull, or intersection. Zonotopes are a subclass of central-symmetric polytopes that is closed under Minkowski sum, and for which good approximations of the convex hull can be efficiently obtained. Zonotopes have been used successfully for reachability analysis [21, 19, 2]. Zonotopes are not closed under intersection, which can make the computation of jump successors problematic. In special cases, *continuization* (interpolation between dynamics of neighboring locations) can help to avoid the intersection operation [3]. Reachability algorithms for affine dynamics can be extended to nonlinear dynamics by piecewise approximation, called *phase potrtrait approximation* [20] or *hybridization* [5]. For polynomial dynamics, sets can be rep-

resented in polynomial form [9, 29]. Similarly, representing sets as polynomial images of hyperboxes can lead to efficient successor computations for nonlinear systems [6, 2]. While such nonconvex set representations can be advantageous for flowpipe approximation, they seem to be less suitable for clustering, containment checking, and emptiness checking (detecting overlap with guard sets and forbidden states).

The remainder of the paper is organized as follows. In the next section, we informally define hybrid automata and give a high-level description of a reachability algorithm. In Sect. 3, we show how support functions can be used as a lazy set representation, accommodating approximate computations and directional refinement. In Sect. 4, we present the components of our space-time reachability algorithm and how they fit together. Experimental results to illustrate the approach are shown in Sect. 5.

2. HYBRID AUTOMATA

We consider hybrid systems described by *hybrid automata*, which are state-transition systems with ODEs associated to the states and assignments associated to the transitions. An approximation of the reachable states of a hybrid automaton can be obtained by computing successor states with respect to time elapse and jumps, repeating the process until all successors have already been encountered in a previous step. This procedure does not necessarily terminate, and the problem is in general undecidable. We give a high-level description of the reachability algorithm, whose operators will be described in more detail in Sect. 4.

A *hybrid automaton* $H = (Loc, Inv, Flow, Trans, Init)$ is defined as follows [4]. It has a set of discrete states Loc called *locations*. Each location $l \in Loc$ is associated with a set of differential inclusions $Flow(l)$ that defines the time-driven evolution of the continuous variables. A *state* $s \in Loc \times \mathbb{R}^n$ consists of a location and values for the n continuous variables. A set of *discrete transitions* $Trans$ defines how the state can jump between locations and instantaneously modify the values of continuous variables. A jump can take place when the state is inside the transition's *guard* set, and the target states are given by the transition's *assignment*. The system can remain in a location l while the state is inside the *invariant* set $Inv(l)$. All behavior originates from the set of *initial states* $Init$.

In this paper, we consider $Flow(l)$ to be ODEs of the form

$$\dot{x}(t) = Ax(t) + u(t), \qquad u(t) \in \mathcal{U}, \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is an n-dimensional vector, $A \in \mathbb{R}^{n\times n}$ and $\mathcal{U} \subseteq \mathbb{R}^n$ is a closed and bounded convex set. Transition assignments are of the (deterministic) affine form

$$x' = Rx + w, \tag{2}$$

where $x' \in \mathbb{R}^n$ denotes the values after the transition, $R \in \mathbb{R}^{n\times n}$ and $w \in \mathbb{R}^n$.

We compute the reachable states by recursively computing the image of the initial states with respect to time elapse and discrete transitions until a fixed-point is reached. The standard method to compute the reachable states is to iterate the following *one-step successor* operators for discrete and continuous transitions. Given a set of states S, let $\mathsf{Post}_C(S)$ be the set of states reachable by letting time elapse from any state in S, and let $\mathsf{Post}_D(S)$ be the set of states resulting from a jump from any state in S. Starting from the initial states, $\mathsf{Post}_C(S)$ and $\mathsf{Post}_D(S)$ are computed and recorded in alternation, as in the following sequence:

$$\begin{aligned} R_0 &= \mathsf{Post}_C(Init), \\ R_{i+1} &= R_i \cup \mathsf{Post}_C(\mathsf{Post}_D(R_i)). \end{aligned}$$

If the sequence reaches a fixed-point, i.e., when $R_{i+1} = R_i$, then R_i is the set of reachable states.

A set of states S is typically represented by a set of *symbolic states* $(\ell, \mathcal{X})$, with location ℓ and continuous set $\mathcal{X} \subseteq \mathbb{R}^n$. The operators $\mathsf{Post}_C(S)$ and $\mathsf{Post}_D(S)$ then realized by enumerating over the symbolic states $(\ell, \mathcal{X})$ in S, and computing the set of successors of $\mathcal{X}$. The time elapse successors of $\mathcal{X}$ are called its *flowpipe*. Computing the jump successors may generate more than one symbolic successor state per transition. This can quickly lead to an explosion in the number of symbolic states in R_i. Detecting a fixed-point requires checking containment between continuous sets. Note that containment checking can also be used to remove redundant states in R_i, which lessens the state explosion.

The system is called *safe* if a given set of unsafe states F is not reachable. This is true if $R_i \cap F = \emptyset$, which can be verified pairwise through intersection and emptiness-checking on the symbolic states in R_i and F.

3. LAZY SET REPRESENTATION

Our goal is to compute a sequence of sets that cover the solutions of a set of ordinary differential equations (ODEs). This requires solving the ODEs in some form, and we proceed similarly as in numerical simulation, but computing with sets instead of numbers. Starting from an initial set, each successor set is constructed from the predecessor with geometric set operations that arise from the dynamic equations. These set operations need to be accurate and scalable, and the choice of set representation turns out to be vital. We use support functions, as proposed by Le Guernic and Girard [23], since several operations with typically exponential cost are only of constant or linear complexity when applied to support functions.

3.1 Support Functions

We now introduce our notation, give definitions for polyhedra and support functions, and recall some fundamental properties. A *halfspace* $\mathcal{H} \subseteq \mathbb{R}^n$ is the set of points satisfying a linear constraint, $\mathcal{H} = \{x \mid a^{\mathsf{T}}x \le b\}$, with *normal vector* $a = (a_1 \cdots a_n) \in \mathbb{R}^n$ and $b \in \mathbb{R}$. A *polyhedron* $\mathcal{P} \subseteq \mathbb{R}^n$ is the intersection of a finite number of halfspaces, written as

$$\mathcal{P} = \Big\{\bigwedge\nolimits_{i=1}^{m} a_i^{\mathsf{T}}x \le b_i\Big\},$$

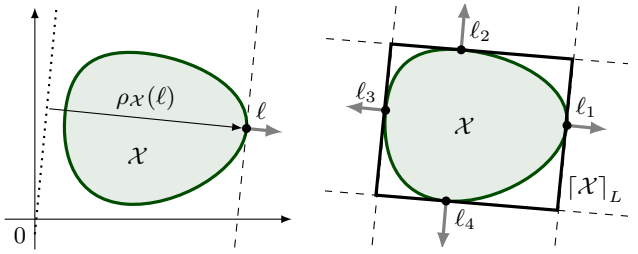
where $a_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$. A *polytope* is a bounded polyhedron. Any convex set can be represented by its support function. The support function of a compact set $\mathcal{X}$ attributes to a direction $\ell \in \mathbb{R}^n$ the scalar value

$$\rho_{\mathcal{X}}(\ell) = \max\{\, \ell^{\mathsf{T}}x \mid x \in \mathcal{X} \,\}.$$

For a given direction ℓ, it defines the position of a halfspace

$$\mathcal{H}_\ell = \{\ell^{\mathsf{T}}x \le \rho_{\mathcal{X}}(\ell)\},$$

which touches and contains $\mathcal{X}$. If ℓ is of unit length, then $\rho_{\mathcal{X}}(\ell)$ is the signed distance of $\mathcal{H}_\ell$ to the origin, see Fig. 1(a) for an illustration. Evaluating the support function for a set



(a) support function (b) outer approximation

Figure 1: Evaluating the support function in a set of directions gives a polyhedral outer approximation

of directions $L \subseteq \mathbb{R}^n$ gives an *outer approximation*

$$\lceil\mathcal{X}\rceil_L = \bigcap_{\ell\in L} \{\ell^\mathsf{T} x \le \rho_{\mathcal{X}}(\ell)\}, \tag{3}$$

i.e., $\mathcal{X} \subseteq \lceil\mathcal{X}\rceil_L$. If $L = \mathbb{R}^n$, then $\mathcal{X} = \lceil\mathcal{X}\rceil_L$, so the support function represents $\mathcal{X}$ exactly. If L is a finite set of directions $L = \{\ell_1, \ldots, \ell_m\}$, then $\lceil\mathcal{X}\rceil_L$ is a polyhedron, as shown in Fig. 1(b). This also referred to as a *template polyhedron* with L being the *template directions*. The difference between using support functions and traditional methods for template polyhedra, e.g., [30], lies in the fact that the outer approximation $\lceil\mathcal{X}\rceil_L$ can be refined at any time, and incrementally, by adding more directions to L. One can interpret evaluating support functions as the lazy, on-demand, construction of a template polyhedron.

3.2 Geometric Operations

The following set operations are required by our reachability algorithm, and are extremely efficient on support functions. Consider non-empty compact convex sets $\mathcal{X}, \mathcal{Y} \subseteq \mathbb{R}^n$. The *linear map* with a matrix $M \in \mathbb{R}^{m\times n}$ is $M\mathcal{X} = \{Mx \mid x \in \mathcal{X}\}$. If $\mathcal{X}$ is a polyhedron and M is singular, then computing the constraint form of $M\mathcal{X}$ requires existential quantification of complexity $\mathcal{O}(\exp(n))$. Using support functions, the linear map simplifies to

$$\rho_{M\mathcal{X}}(\ell) = \rho_{\mathcal{X}}(M^\mathsf{T}\ell), \tag{4}$$

which is $\mathcal{O}(mn)$. The *convex hull* of as set $\mathcal{Z} \subseteq \mathbb{R}^n$ is

$$\mathrm{CH}(\mathcal{Z}) = \left\{ \sum_{i=1}^m \lambda_i v_i \;\middle|\; v_i \in \mathcal{Z}, \lambda_i \in \mathbb{R}^{\ge 0}, \sum_{i=1}^m \lambda_i = 1 \right\}.$$

Using support functions, the convex hull of $\mathcal{X}$ and $\mathcal{Y}$ is

$$\rho_{\mathrm{CH}(\mathcal{X}\cup\mathcal{Y})}(\ell) = \max\{\rho_{\mathcal{X}}(\ell), \rho_{\mathcal{Y}}(\ell)\},$$

which is $\mathcal{O}(1)$. The *Minkowski sum* is $\mathcal{X} \oplus \mathcal{Y} = \{x + y \mid x \in \mathcal{X}, y \in \mathcal{Y}\}$, which translates to the $\mathcal{O}(1)$ operation

$$\rho_{\mathcal{X}\oplus\mathcal{Y}}(\ell) = \rho_{\mathcal{X}}(\ell) + \rho_{\mathcal{Y}}(\ell).$$

In the following example, a sequence of linear mapping and Minkowski sums leads to a convex set that is prohibitively complex for polyhedral operations, but that is easy to approximate with support functions.

Example 3.1. *We consider dynamics of the form (1), i.e., $\dot{x} = Ax + u$, with $u(t) \in \mathcal{U}$ and $x(0) \in \mathcal{X}_0$. At a time instant t, the states reachable with theses dynamics are*

$$\mathcal{X}_t = e^{At}\mathcal{X}_0 \oplus \int_0^t e^{As}\mathcal{U}ds = e^{At}\mathcal{X}_0 \oplus \lim_{\delta\to 0} \bigoplus_{k=0}^{\lfloor t/\delta\rfloor} e^{A\delta k}\delta\mathcal{U}.$$

The matrix $e^{A\delta k}$ is different for each k, so the Minkowski sum leads to infinitely many vertices and constraints as $\delta \to 0$. Even if $\mathcal{X}_0$ and $\mathcal{U}$ are polyhedra, $\mathcal{X}_t$ can be a smooth set [31]. An outer approximation of $\mathcal{X}_t$ is obtained by bloating $\delta\mathcal{U}$ by a sufficient amount. Let the bloating factor be

$$\epsilon_\delta = \sum_{i=1}^\infty \frac{\|A\|^i\delta^{i+1}}{(i+1)!} = (e^{\|A\|\delta} - 1 - \|A\|\delta)/\|A\|,$$

$\mathcal{B}$ the unit ball of the norm, and t a multiple of δ. The set

$$\Omega_t^\delta = e^{At}\mathcal{X}_0 \oplus \bigoplus_{k=0}^{t/\delta-1} e^{A\delta k}\big(\delta\mathcal{U} \oplus \epsilon_\delta\|\mathcal{U}\|\mathcal{B}\big),$$

is an outer approximation of $\mathcal{X}_t$ within distance $2\epsilon_\delta$ [23]. Using polyhedra in constraint form, Ω_t^δ is prohibitively expensive to compute, since the number of constraints may grow exponentially. Using support functions, we get

$$\rho_{\Omega_t^\delta}(\ell) = \rho_{\mathcal{X}_0}(e^{At^\mathsf{T}}\ell) + \sum_{k=0}^{t/\delta-1} \delta\rho_{\mathcal{U}}(e^{A\delta k^\mathsf{T}}\ell) + \epsilon_\delta\|\mathcal{U}\|\|e^{A\delta k^\mathsf{T}}\ell\|. \tag{5}$$

The directions $e^{A\delta k^\mathsf{T}}\ell$ can be computed with the sequence $\ell_0 = \ell$, $\ell_{k+1} = e^{A\delta^\mathsf{T}}\ell_k$, up to $k = t/\delta$. Note that $e^{At^\mathsf{T}}\ell$ is the last element in the sequence. Let $\mathcal{L}(\mathcal{X})$ be the cost of computing the support function of $\mathcal{X}$, in our case solving a linear program. Computing $\rho_{\Omega_t^\delta}(\ell)$ is $\mathcal{O}\left(\mathcal{L}(\mathcal{X}_0) + n^2\frac{t}{\delta} + \mathcal{L}(\mathcal{U})\frac{t}{\delta}\right)$. If the goal is to compute an approximation of $\rho_{\Omega_t}(\ell)$ for a sequence of time instants $t = 0, \delta, \ldots, (N-1)\delta$, the summands can be reused. The computation of the entire sequence is of complexity $\mathcal{O}\left(N\mathcal{L}(\mathcal{X}_0) + Nn^2 + N\mathcal{L}(\mathcal{U})\right)$ [23].

Evidently, support functions do not enable us to avoid the curse of dimensionality in general: An n-dimensional approximation with a distance of ε to the real set requires $\mathcal{O}\left(1/\varepsilon^{n-1}\right)$ evaluations of the support function [24]. But many problems do not require precise knowledge of the whole set; with the support function the approximation can be targeted to where it matters. A simple example shall illustrate this point.

Example 3.2. *Consider the halfspace $\mathcal{H} = \{a^\mathsf{T}x \ge b\}$. Then $\mathcal{X} \cap \mathcal{H} \ne \emptyset$ if and only if $\rho_{\mathcal{X}}(a) \ge b$. Furthermore, $\mathcal{X} \subseteq \mathcal{H}$ if and only if $-\rho_{\mathcal{X}}(-a) \ge b$. Each problem can be decided with a single evaluation of the support function.*

The problem of whether two convex sets intersect is known as the *separation problem*, and it can be addressed by computing the support function in a sequence of directions. Let

$$h(\ell) = \rho_{\mathcal{X}\oplus(-\mathcal{Y})}(\ell) = \rho_{\mathcal{X}}(\ell) + \rho_{\mathcal{Y}}(-\ell),$$

which is a convex function. Then $\mathcal{X} \cap \mathcal{H} = \emptyset$ if and only if $\min_{\ell\in\mathbb{R}^n} h(\ell) < 0$. Any converging convex minimization algorithm applied to $h(\cdot)$ will produce a sequence of directions that converges to a separating direction if it exists. Separation algorithms adapted to the context of approximate reachability have been proposed in [13].

Two more operations of our reachability algorithm, intersection and containment, turn out to be difficult for support

functions. The intersection $\mathcal{X} \cap \mathcal{Y}$ requires solving a convex optimization problem:

$$\rho_{\mathcal{X}\cap\mathcal{Y}}(\ell) = \inf_{\nu\in\mathbb{R}^n} \rho_{\mathcal{X}}(\ell - \nu) + \rho_{\mathcal{Y}}(\nu).$$

Containment checking with support functions generally involves an uncountable number of directions: $\mathcal{X} \subseteq \mathcal{Y}$ if and only if $\rho_{\mathcal{X}}(\ell) \leq \rho_{\mathcal{Y}}(\ell)$ for all $\ell \in \mathbb{R}^n$. We circumvent these problems by switching to polyhedra as set representation at corresponding points in the algorithm; for details see Sect. 4. Intersection of polyhedra is cheap since it amounts to taking the union of their constraints. Containment checking is simple if the right hand side is a polyhedron. Let $\mathcal{P} = \left\{\bigwedge_{i=1}^m a_i^\mathsf{T} x \leq b_i\right\}$, then $\mathcal{X} \subseteq \mathcal{P}$ if and only if $\rho_{\mathcal{X}}(a_i) \leq b_i$ for $i = 1, \ldots, m$.

3.3 Lower Bounds and Inner Approximation

Support functions are generally known for their capacity to cheaply produce tight outer approximations. Having evaluated the support function in some directions gives a lower bound on the support function in all other directions [15]. A lower bound on the support function can be more powerful than conventional under-approximations, since it may decide properties like intersection even before any point in the set is known to actually be in the set.

EXAMPLE 3.3. *Assume we evaluated the support function of $\mathcal{X} \subseteq \mathbb{R}^2$ in three directions and obtained an outer approximation in the form of a triangle with nonempty interior. Any of the facets of the triangle would give the same outer approximation, so $\mathcal{X}$ could be any of the facets. If we could deduce any point in $\mathcal{X}$ based on the outer approximation, it would therefore need to be in all three facets. But their intersection is empty if the interior is nonempty, which is a contradiction. Nonetheless, the lower bound on the support function given in [15] can show that $\mathcal{X}$ overlaps with any halfspace that contains one of the facets.*

When evaluated for sufficiently distributed directions, the lower bound also defines an inner approximation, i.e., a set of points that are guaranteed to be inside the set [13]. In Sect. 4.2, we will exploit lower bounds on the support functions for obtaining convex cover of the nonconvex flowpipe.

3.4 Approximate Evaluation

In general, support functions can only be computed with finite precision, e.g., due to rounding errors, and computing them with low precision may be desirable to reduce the computation costs. To formalize this notion, we consider a function $\mathsf{support}$ that, given a direction ℓ and an accuracy $\varepsilon > 0$, produces both an upper and a lower bound on the support function:

$$\mathsf{support}(\mathcal{X}, \ell, \varepsilon) - \varepsilon \leq \rho_{\mathcal{X}}(\ell) \leq \mathsf{support}(\mathcal{X}, \ell, \varepsilon).$$

EXAMPLE 3.4. *Consider the computation of the support function of the reachable states in Ex. 3.1. The exact support function is the limit of an infinite series, and the approximation given in (5) is an upper bound. The approximation is within distance $2\epsilon_\delta\|\mathcal{U}\|$ of the real solution, so the error on the support function is bounded by $2\epsilon_\delta\|\mathcal{U}\|\|\ell\|$. This gives the lower and upper bound on the support function*

$$\rho_{\Omega_t^\delta}(\ell) - 2\epsilon_\delta\|\mathcal{U}\|\|\ell\| \leq \rho_{\mathcal{X}_t}(\ell) \leq \rho_{\Omega_t^\delta}(\ell).$$

The error above is a result of the time step δ. To meet the required error bound, δ can be adapted, e.g., through bisection. A parameter of the ODE solving method is thus replaced with a parameter with geometric significance in the state space, which can be exploited, e.g., in testing containment and intersection.

The set operations and approximations from the previous sections readily extend to approximate evaluations, leading to a conservative overapproximation

$$\lceil\mathcal{X}\rceil_L^+ = \bigcap_{\ell_k\in L} \left\{\ell_k^\mathsf{T} x \leq \mathsf{support}(\mathcal{X}, \ell_k, \varepsilon)\right\}.$$

However, this introduces an additional degree of freedom when the approximation needs to be refined: Should we evaluate the support function in more directions, or increase the precision? The lower bound on the support function can help make this call, as illustrated by the following example.

EXAMPLE 3.5. *Consider the halfspace $\mathcal{H} = \{a^\mathsf{T} x \geq b\}$ and its intersection with the reachable set $\mathcal{X}_t$ for a given value of t. Assume we evaluate $s = \mathsf{support}(\mathcal{X}_t, a, \varepsilon)$ for some arbitrarily chosen accuracy ε. If $s < b$, then $\mathcal{X}_t$ and $\mathcal{H}$ are disjoint. If $s - \varepsilon \geq b$, then $\mathcal{X}_t$ intersects with $\mathcal{H}$. Otherwise, we must refine our approximation of $\mathcal{X}_t$. Evaluating $s_k = \mathsf{support}(\mathcal{X}_t, a, \varepsilon_k)$ with $\varepsilon_k = 2^{-k}$ will eventually lead to a value $s_k < b$ or $s_k \geq b$ except in the pathological case where $\rho_{\mathcal{X}_t}(a) = b$. If instead of reducing ε we had chosen additional directions ℓ_k and evaluated $\mathsf{support}(\mathcal{X}_t, \ell_k, \varepsilon)$, the outer approximation $\lceil\mathcal{X}_t\rceil_L^+$ would increase in accuracy but could remain intersecting with $\mathcal{H}$ for any choice of L.*

4. SPACE-TIME REACHABILITY

We first present our reachability algorithm and then discuss its components in separate subsections: flowpipe approximation, convexification, computing jump successors, containment checking, and emptiness checking. Let L be a given set of template directions and ε a given accuracy. In our algorithm, we represent the reachable states with a *passed list* P of symbolic states, which initially empty, and keep a *waiting list* W, which initially contains the initial states *Init*. We start by computing the time successors. We pop a symbolic state $(q, \mathcal{X}_0)$ from W, where $\mathcal{X}_0$ is a nonempty, compact and convex set represented by its support function (in our implementation, a function object). The time successors of $\mathcal{X}_0$ are computed as follows:

1. Determine a time horizon T for the flowpipe by finding the smallest t such that the flowpipe no longer intersects with the invariant (timed flowpipe separation from Sect. 4.5).
2. For each template direction $\ell_i \in L$, compute a bound on the support function over the time domain $[0, T]$ (flowpipe approximation from Sect. 4.1).
3. The flowpipe approximation $\bar{\Omega}_{0,T}$, as defined in (6), is added to the passed list P.

Note that at this point convexification is not yet necessary. For each of the outgoing transitions τ in location q, we compute the jump successors of $\bar{\Omega}_{0,T}$ as follows:

4. Determine a cover $I_1, I_2, \ldots$ of the time domain in which the flowpipe intersects the guard (timed flowpipe separation from Sect. 4.5)

5. For each interval I_k, remove from I_k the subintervals for which $\bar{\Omega}_t$ is contained in one of the previously computed flowpipe approximations on the passed or the waiting list (containment from Sect. 4.4). Discard intervals that have become empty.
6. For each interval I_k, compute a convex cover $\bar{\Omega}_1, \bar{\Omega}_2, \ldots$ of $\bar{\Omega}_{I_k}$ (convexification from Sect. 4.2).
7. For each $\bar{\Omega}_j$ in the convex cover of $\bar{\Omega}_{I_k}$, compute the intersection with $\mathcal{G}^*$ and discard if empty. Otherwise, instantiate the set $\mathcal{X}' = \mathsf{post}_\tau(\bar{\Omega}_j)$ as a support function object and add the symbolic state $(q', \mathcal{X}')$ to the waiting list W (jump successors from Sect. 4.3).

The time successor and jump successor computations in steps 1.-7. are repeated until the waiting list is empty. If the process terminates, the symbolic states on the passed list contain the set of reachable states of the system.

Note that convexification and containment checking are only carried out on segments of the flowpipe that intersect with a guard. Furthermore, containment checking is carried out before the convexification step, so that approximation errors from the convexification step do not impact the containment checking.

4.1 Flowpipe Approximation

In a given location of the hybrid automaton, we refer to the states reachable from an initial set $\mathcal{X}_0$ by time elapse as the *flowpipe* of $\mathcal{X}_0$. We assume that $\mathcal{X}_0$ is convex. Given an initial set $\mathcal{X}_0$, the *reachable states at time* t is the set of values of the solutions of (1) with initial condition $x(0) \in \mathcal{X}_0$. We denote this set with

$$\mathcal{X}_t = e^{At}\mathcal{X}_0 \oplus \int_0^t e^{As}\mathcal{U} ds.$$

For affine dynamics, $\mathcal{X}_t$ is convex for any given t, so $\mathcal{X}_t$ can be represented by its support function. The flowpipe segment over the time interval $[t_b, t_e]$ is the set $\mathcal{X}_{t_b,t_e} = \bigcup_{t_b \le t \le t_e} \mathcal{X}_t$.

We extend the results from Sect. 3 to flowpipes by applying them pointwise over time: since the flowpipe is a convex set at each point in time, the entire flowpipe can be described by a support function that is parameterized over time. Put formally, let the support function over time be $s_\ell(t) = \rho_{\mathcal{X}_t}(\ell)$. As a straightforward consequence of (3), the functions $s_\ell(t)$ describe the flowpipe exactly, i.e.,

$$\mathcal{X}_{t_b,t_e} = \bigcup_{t_b \le t \le t_e} \bigcap_{\ell \in \mathbb{R}^n} \{\ell^\mathsf{T} x \le s_\ell(t)\}.$$

We now describe how approximations of $s_\ell(t)$ can be used to derive an approximation of $\mathcal{X}_{t_b,t_e}$. Given an interval $[t_b, t_e]$, a direction ℓ, and an accuracy bound $\varepsilon > 0$, we construct a piecewise linear function $s^+_{\ell,\varepsilon} : [t_b, t_e] \to \mathbb{R}$ with

$$s^+_{\ell,\varepsilon}(t) - \varepsilon \le \rho_{\mathcal{X}_t}(\ell) \le s^+_{\ell,\varepsilon}(t) \quad \text{for all} \quad t \in [t_b, t_e].$$

A method to effectively compute $s^+_{\ell,\varepsilon}(t)$ is described in [15]. Briefly summarized, the support function is computed at discrete time points, similar to Ex. 3.1. The continuous bound is then obtained from a linear interpolation between the discrete points, augmenting (bloating) it enough to be conservative between the sampling points. The details of the construction are omitted since they are somewhat technical and the remainder of this paper applies to any $s^+_{\ell,\varepsilon}(t)$ as long as it is piecewise linear.

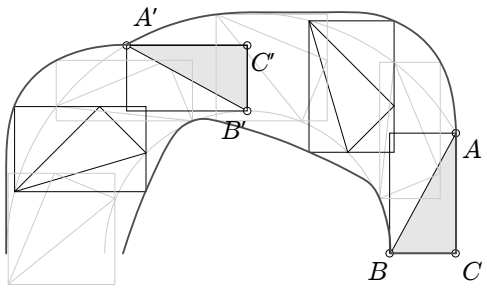


Figure 2: A flowpipe approximation constructed with the axis directions as templates, starting from the initial set $\mathcal{X}_0 = ABC$. At time t', the approximation is the bounding box of the reachable set $\mathcal{X}_{t'} = A'B'C'$

Assume we have computed $s^+_{\ell_i,\varepsilon}(t)$ for a set of directions $L = \{\ell_1, \ldots, \ell_m\}$. These functions define a flowpipe approximation as follows. For all t, the outer approximation

$$\Omega_t = \lceil \mathcal{X}_t \rceil^+_L = \bigcap_{\ell_i \in L} \{\ell_i^\mathsf{T} x \le s^+_{\ell_i,\varepsilon}(t)\} \tag{6}$$

satisfies $\mathcal{X}_t \subseteq \Omega_t$. Taking the union over the time interval $[t_b, t_e]$, an outer approximation of the flowpipe segment is

$$\Omega_{t_b,t_e} = \bigcup_{t_b \le t \le t_e} \Omega_t, \text{ with } \mathcal{X}_{t_b,t_e} \subseteq \Omega_{t_b,t_e}.$$

This flowpipe approximation is defined as the union of infinitely many convex polyhedra, see Fig. 2. But for our reachability algorithm the flowpipe approximation needs to produce finitely many convex sets. To transform this infinite union into a finite one, we consider the problem in space-time, which will be described in the following section.

So far, we have ignored the invariant of the location. Since the solution of the ODEs under constraints is a difficult problem, we follow a frequently used heuristic and intersect the flowpipe approximation a posteriori with the invariant – this will be discussed as part of the jump successor computation in Sect. 4.3. In addition, we cut off the flowpipe approximation when Ω_t lies completely outside of the invariant. This is a timed flowpipe separation problem, which will be discussed in Sect. 4.5. A more precise approximation of the flowpipe under invariant constraints is described in [23], and is based on intersecting the flowpipe approximation with states that are backwards-reachable from the invariant.

4.2 Convexification

We now turn to the problem of transforming a given flowpipe approximation Ω_{t_b,t_e} into a finite number of convex sets, ideally as few as possible. We call this step *convexification*. So far we have considered operations in the *continuous state space* $\mathbb{R}^n$. In the following, we operate in *space-time* $\mathbb{R}^{n+1}$, with t as an additional variable. This will allow us to show that Ω_{t_b,t_e} is a finite union of polyhedra, and furthermore to overapproximate and simplify Ω_{t_b,t_e} within given error bounds. For the sake of clarity, we will denote sets in space-time with a bar, as in $\bar{\mathcal{X}}$. In space-time, we associate the sets $\mathcal{X}_t$ and Ω_t with the time instant t at which they are defined. The flowpipe segment and its approximation

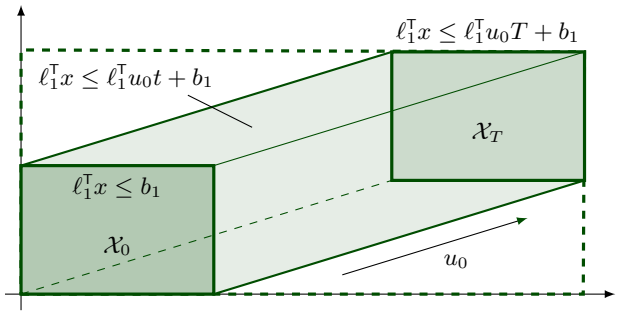


Figure 3: Projected from space-time onto the state-space, the flowpipe approximation has facet normals that are not part of the template directions. This reduces the approximation error compared to a template approximation in the state-space (dashed)

become

$$\bar{\mathcal{X}}_{t_b,t_e} = \bigcup_{t_b \le t \le t_e} \mathcal{X}_t \times t \quad \text{and} \quad \bar{\Omega}_{t_b,t_e} = \bigcup_{t_b \le t \le t_e} \Omega_t \times t.$$

Now consider a subinterval $[t'_b, t'_e]$ of $[t_b, t_e]$ such that the bounds on the support functions are all linear, i.e., for $i = 1, \ldots, m$ and $t'_b \le t \le t'_e$ we have $s^+_{\ell_i,\varepsilon}(t) = \alpha_i t + \beta_i$. Then the flowpipe approximation is

$$\bar{\Omega}_{t'_b,t'_e} = \bigcup_{t'_b \le t \le t'_e} \bigcap_{\ell_i \in L} \left\{ \ell_i^\mathsf{T} x \le \alpha_i t + \beta_i \right\} \times t,$$

which can be shown to be equivalent to

$$\bar{\Omega}_{t'_b,t'_e} = \bigcap_{\ell_i \in L} \left\{ \ell_i^\mathsf{T} x - \alpha_i t \le \beta_i \right\} \cap \{ t'_b \le t \le t'_e \}. \qquad (7)$$

This is a convex polyhedron in space-time that approximates the flowpipe segment, i.e., $\bar{\mathcal{X}}_{t'_b,t'_e} \subseteq \bar{\Omega}_{t'_b,t'_e}$. It is remarkable that this approximation of the flowpipe — valid over an entire interval of time — requires just $m + 2$ constraints over $n + 1$ variables. Its complexity is similar to that of an outer approximation of the initial states, which would give m constraints over n variables.

The space-time set $\bar{\Omega}_{t'_b,t'_e}$ can be projected into the state-space by eliminating t. Fourier-Motzkin elimination leads to $\mathcal{O}\left(m^2\right)$ constraints of the form

$$(\alpha_i \ell_j - \alpha_j \ell_i)^\mathsf{T} x \le \beta_j \alpha_i - \beta_i \alpha_j,$$

defined by all combinations of $\alpha_i > 0$ and $\alpha_j < 0$. This means we obtain facet normals that are not necessarily part of the template directions. The space-time constraints in (7) can be seen as a special case of the template generalizations in [7]. The non-template facets created by working in space-time can significantly decrease the approximation error compared to state-space templates, and even produce the exact reachable set for certain dynamics and choice of template directions.

Example 4.1. *Consider a system with constant dynamics $\dot{x} = u_0$ (a linear hybrid automaton), and let the template directions be the facet normals of the initial states, i.e., $\mathcal{X}_0 = \bigcap_{\ell_i \in L} \left\{ \ell_i^\mathsf{T} x \le b_i \right\}$. For these dynamics, the flowpipe is*

$$\mathcal{X}_t = \mathcal{X}_0 \oplus t\mathcal{U} = \mathcal{X}_0 \oplus t u_0 = \bigcap_{\ell_i \in L} \left\{ \ell_i^\mathsf{T} x - \ell_i^\mathsf{T} u_0 t \le b_i \right\}.$$

For the time domain $[0, T]$, the flowpipe approximation algorithm in [15] computes the bounds on the support functions as the linear interpolation between the support at $t = 0$ and $t = T$, plus appropriate bloating. Because $A = 0$, the bloating factor is zero and the exact bounds are returned: $s^+_{\ell_i,\varepsilon}(t) = (1 - t/T)\rho_{\mathcal{X}_0}(\ell_i) + t/T \rho_{\mathcal{X}_T}(\ell_i) = (1 - t/T)\rho_{\mathcal{X}_0}(\ell_i) + t/T \rho_{\mathcal{X}_0}(\ell_i) + t/T \rho_{Tu_0}(\ell_i) = \rho_{\mathcal{X}_0}(\ell_i) + t\rho_{u_0}(\ell_i) = b_i + t\ell_i^\mathsf{T} u_0$. Since these bounds are concave, the space-time approximation (7) consists of a single polyhedron

$$\bar{\Omega}_{0,T} = \bigcap_{\ell_i \in L} \left\{ \ell_i^\mathsf{T} x - \ell_i^\mathsf{T} u_0 t \le b_i \right\} \cap \{ 0 \le t \le T \},$$

whose projection onto the state-space is identical to $\mathcal{X}_{0,T}$, as illustrated in Fig. 3.

Quantifier elimination on the space-time constraints would significantly increase the complexity. Using support functions, projection onto the state space comes practically for free, since with (4) we have

$$\rho_{\Omega_{t_b,t_e}}(\ell) = \rho_{\bar{\Omega}_{t_b,t_e}}(\ell \times 0).$$

Assuming each of the m piecewise linear functions $s^+_{\ell_i,\varepsilon}(t)$ has K pieces, the time interval $[t_b, t_e]$ can be cut into no more than Km pieces, inside which all functions are linear. Consequently, $\bar{\Omega}_{t_b,t_e}$ can be described as the union of Km convex polyhedra, each of which defined as in (7). To avoid the state explosion problem it is essential to obtain as few sets as possible. In the following, we describe a method to divide $\bar{\Omega}_{t_b,t_e}$ into the smallest number N of convex sets for a given bound on the total approximation error.

We now show that $\bar{\Omega}_{t'_b,t'_e}$ is a convex polyhedron over any interval $[t'_b, t'_e]$ where the bounds on the support function are concave. Assume that all $s^+_{\ell_i,\varepsilon}(t)$ are concave in $[t'_b, t'_e]$ with K pieces, the j-th piece of $s^+_{\ell_i,\varepsilon}(t)$ being $\alpha_{i,j} t + \beta_{i,j}$. Because the functions are concave, $s^+_{\ell_i,\varepsilon}(t) \le \alpha_{i,j} t + \beta_{i,j}$ for all $t \in [t'_b, t'_e]$ and for all j we have with (7) that

$$\bar{\mathcal{X}}_{t'_b,t'_e} \subseteq \bigcap_{\ell_i \in L} \left\{ \ell_i^\mathsf{T} x - \alpha_{i,j} t \le \beta_{i,j} \right\} \cap \{ t'_b \le t \le t'_e \}.$$

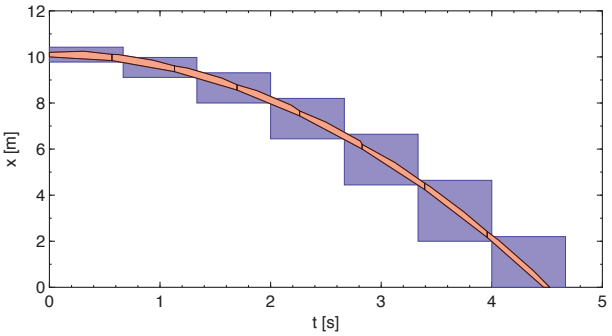
Intersecting the right hand side over all values of j gives the flowpipe approximation

$$\bar{\Omega}_{t'_b,t'_e} = \bigcap_{\ell_i \in L} \bigcap_{1 \le j \le K} \left\{ \ell_i^\mathsf{T} x - \alpha_{i,j} t \le \beta_{i,j} \right\} \cap \{ t'_b \le t \le t'_e \}. \quad (8)$$

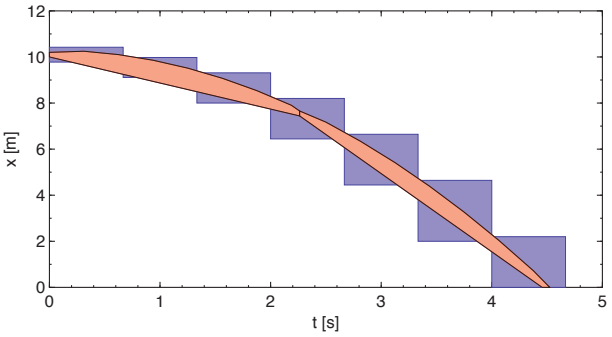
We can therefore convexify $\bar{\Omega}_{t_b,t_e}$ by identifying the largest subintervals where all $s^+_{\ell_i,\varepsilon}(t)$ are concave, and obtain a convex polyhedron (8) for each of those subintervals.

Example 4.2. *Figure 4 shows different flowpipe approximations of the ODE $\ddot{x} = -10$, with initial condition $9.9 \le x \le 10.1$, $\dot{x} = 0$, axis directions as template directions and directional error bound $\varepsilon = 0.1$. In Fig. 4(a), the flowpipe approximation is constructed by identifying the subintervals in which all $s^+_{\ell_i,\varepsilon}(t)$ are concave. Each subinterval defines a convex polyhedron, giving a total of 8 polyhedra. For comparison, the figure also shows the 7 boxes produces by the state-space template approximation from [17], using the same parameters.*

The number of polyhedra in $\bar{\Omega}_{t_b,t_e}$ can be further reduced by taking the convex hull over the entire time domain or parts of it. The convex hull of states corresponds to the



(a) state-space templates from [17] (blue) and space-time approximation (red) with error $\varepsilon = 0.1$ gives 8 convex sets



(b) optimal convexification from [15] for a total error of $\varepsilon = 1$ reduces the number of convex sets to 2

Figure 4: Flowpipe approximations with axis directions as template directions, using different convexification schemes

concave hull of the support functions. The terminology is reversed since a function is called convex if its epigraph (the points above) is a convex set, while the support functions relate to states via their hypograph (the points below). The concave hull of $s^+_{\ell_i,\varepsilon}(t)$ on a time interval $[t'_b, t'_e]$ can be computed using the Graham scan in $\mathcal{O}(K)$, where K is the number of linear pieces. Importantly, the convexification error is measurable in this approach. Let $c_i(t)$ be the concave hull of $s^+_{\ell_i,\varepsilon}(t)$. Recalling that $s^+_{\ell_i,\varepsilon}(t)$ was constructed with an accuracy of ε, we have that

$$s^+_{\ell_i,\varepsilon}(t) - \varepsilon \leq \rho_{\mathcal{X}_t}(\ell_i) \leq c_i(t),$$

which means that the approximation error in direction ℓ_i is smaller than $\max_{t_b \leq t \leq t_e} c_i(t) - s^+_{\ell_i,\varepsilon}(t) + \varepsilon$. The convexification can be taken one step further. As described in [15], it is possible to construct for a given *total* approximation error a relaxation of $\bar{\Omega}_{t_b,t_e}$ with the minimum number of convex pieces.

Example 4.3. *The flowpipe approximation in Fig. 4(b) was constructed by first computing the same support function bounds $s^+_{\ell_i,\varepsilon}(t)$ as in Ex. 4.2, and then applying the optimal convexification approach from [15]. For a given total error (flowpipe approximation plus convexification) of $\varepsilon = 1$, the relaxation, the support function bounds can be approximated with just 2 concave pieces in all template directions, giving an flowpipe approximation with 2 polyhedra.*

The template directions together with the approximation error therefore determine the number of convex sets with which one can cover the flowpipe. Note this number is not necessarily minimal in the state space: A flowpipe that is convex in the state space may be nonconvex in space-time. But automatic convexification for a given error bound can be tremendously helpful in that it takes care of the additional degree of freedom, i.e., the number of convex sets in the cover: As with convex sets, the approximation accuracy is determined by the choice of template directions and the directional error.

After the convexification step, each concave piece of $s^+_{\ell_i,\varepsilon}(t)$ can be further simplified, i.e., overapproximated with a piecewise linear function with fewer pieces. Since each linear piece of $s^+_{\ell_i,\varepsilon}(t)$ contributes a linear constraint to the flowpipe approximation (8), reducing the number of pieces also reduces the number of constraints in the convex polyhedra and therefore the computational effort required downstream.

4.3 Jump Approximation

The successor states of a transition are computed as follows. Let $\mathcal{G}$ be the guard set of the transition, $\mathcal{I}^-$ the invariant of the source location, $\mathcal{I}^+$ the invariant of the target location, and let the transition assignment be deterministic and affine as in (2). We consider $\mathcal{G}$, $\mathcal{I}^-$, $\mathcal{I}^+$ to be polyhedra and assume that the set of template directions L contains the normal vectors of the constraints of these polyhedra. Let the target invariant be

$$\mathcal{I}^+ = \Big\{ x \;\Big|\; \bigwedge\nolimits_{i=1}^{m} \bar{a}_i^\mathsf{T} x \leq \bar{b}_i \Big\}.$$

The image of a set $\mathcal{X}$ with respect to a transition τ is

$$\mathsf{post}_\tau(\mathcal{X}) = \Big(R\big(\mathcal{X} \cap \mathcal{G} \cap \mathcal{I}^-\big) \oplus w \Big) \cap \mathcal{I}^+.$$

Let G^* be the intersection of the guard, the source invariant, and the back-transformed target invariant,

$$\mathcal{G}^* = \mathcal{G} \cap I^- \cap \Big\{ x \;\Big|\; \bigwedge\nolimits_{i=1}^{m} \bar{a}_i^\mathsf{T} R x \leq \bar{b}_i - w^\mathsf{T} \bar{a}_i \Big\}. \tag{9}$$

Using $\mathcal{G}^*$, the image operator can be simplified so that it involves a single intersection operation [18]:

$$\mathsf{post}_\tau(\mathcal{X}) = R\big(\mathcal{X} \cap \mathcal{G}^*\big) \oplus w. \tag{10}$$

In our reachability algorithm, the jump approximation (10) is applied to the outer approximation of a flowpipe segment. More precisely, we apply it to each convex piece $\bar{\Omega}_{t'_b,t'_e}$ of the approximation, which is defined in space-time. Interpreting the constraints of $\mathcal{G}^*$ as space-time constraints (leaving t unconstrained), the intersection is easily carried out without eliminating t. The projection to the state space and the affine map are straightforward using support functions:

$$\rho_{\mathsf{post}_\tau(\bar{\Omega}_{t'_b,t'_e})}(\ell) = \rho_{\bar{\Omega}_{t'_b,t'_e} \cap \mathcal{G}^*}(M^\mathsf{T}\ell \times 0) + \ell^\mathsf{T} w.$$

4.4 Containment checking

In our reachability algorithm, containment is checked between flowpipe approximations. We apply the comparison

$$\Omega_{t_b,t_e} \subseteq \Omega'_{t'_b,t'_e}$$

to nonconvex Ω (before convexification) and convex Ω' (after convexification). Recall that both are defined over sets of template directions, which are not necessarily the same. For the containment, we only need the directions of $\bar{\Omega}'$, and any directions that are missing in Ω are added to it, computing

support function bounds on-demand if necessary. In general, checking containment between a convex set and a union of convex sets is expensive to compute. A classic heuristic is to only use pairwise comparisons between convex sets. In the case of affine dynamics, it seems that containment could fail indefinitely, if the points of overlap vary.

We obtain an efficient check by exploiting the template properties in space-time. We extend the containment of convex sets, see Sect. 3.2, pointwise over time and detect the time intervals where containment holds. For each template direction ℓ_i of Ω', we construct a set-valued map $C_i : [t_b, t_e] \to [t'_b, t'_e]^2$, where $C_i(t) = [t', t'']$ signifies that in direction ℓ_i, Ω_t is contained $\Omega_{t',t''}$. Formally, $C_i(t) = [t', t'']$ implies for all $z \in [t', t'']$, $s^+_{\ell_i,\varepsilon}(t) \leq {s'}^+_{\ell_i,\varepsilon}(z)$. Because Ω' is convex, its support function bound is a unimodal function, so the matching intervals can usually be found quickly (the points to the left and the right of the maximum are ordered in value). Finally, we combine the results for all directions to the set-valued map $C(t) = \bigcap C_i(t)$. Then $\Omega_t \subseteq \Omega'_z$ for all $z \in C(t)$. If $C(t) \neq \emptyset$ for all $t \in [t_b, t_e]$, then $\Omega_{t_b,t_e} \subseteq \Omega'_{t'_b,t'_e}$. Otherwise, we trim Ω_{t_b,t_e} to the subdomains that are not contained, possibly splitting it into several flowpipe approximations, each of which is treated separately. Note that the $C_i(t)$ and $C(t)$ can be represented with piecewise linear functions.

The containment check is degraded by the approximation error. Consider that overapproximations of the same flowpipe may differ even for the same error bound. Recomputing the left hand side with higher precision may lead to containment, but how much more precision is useful? This can be decided by repeating the containment check using a lower bound on the support function, here $s^+_{\ell_i,\varepsilon}(t) - \varepsilon \leq {s'}^+_{\ell_i,\varepsilon}(z)$. For time points where the lower bound is not contained, refinement with higher precision is useless.

4.5 Emptiness checking

The computation of jump successors in Sect. 4.3 involves only one intersection operation. We can therefore eliminate spurious transitions by deciding whether the flowpipe intersects with $\mathcal{G}^*$. Futhermore, the reachability of an unsafe set of states $\mathcal{F}$ can be cast as intersection of the flowpipe with $\mathcal{F}$, so it reduces to the same problem.

Deciding whether the flowpipe intersects with a convex set $\mathcal{G}^*$ is called *flowpipe separation*, and it can be addressed by synthesizing suitable template directions as proposed in [13]. Since this intersection is the only operation in the reachability algorithm that can lead to an empty set of states, all emptiness checking in the reachability algorithm of Sect. 2 reduces to flowpipe separation.

The flowpipe separation problem is known in several forms. It is equivalent to showing safety of linear time-invariant system, which has been addressed for deterministic dynamics ($\mathcal{U}$ is a singleton) in [10]. It is also equivalent to showing the existence of a controller: does there exist a sequence of control inputs such that the system reaches the guard set? The constructive solution of such control inputs is known as the *falsification* or *counter-example generation* and is generally more difficult than verification of safety properties; for recent approaches see [1, 32].

The intersection of our flowpipe approximation with $\mathcal{G}^*$ may contain states that are not actually reachable. Judiciously shrinking the time interval of the flowpipe segment can help to reduce this error. Instead of $\bar{\Omega}_{t_b,t_e} \cap \mathcal{G}^*$, we can use $\bar{\Omega}_{t'_b,t'_e} \cap \mathcal{G}^*$ if we show that the flowpipe does not intersect with $\mathcal{G}$ for $t_b \leq t < t'_b$ and $t'_e < t \leq t_e$. The *timed flowpipe separation problem* for a given set $\mathcal{G}^*$ consists of covering the time intervals for which the flowpipe overlaps with $\mathcal{G}^*$, i.e., $\mathcal{X}_t \cap \mathcal{G}^* \neq \emptyset$. The jump successors can be computed flowpipe segments of those intervals instead of the entire time horizon. Similarly, the time horizon of the flowpipe can be reduced by timed separation of the flowpipe from the invariant $\mathcal{I}$: Let t^* be the smallest t such that $\mathcal{X}_t \cap \mathcal{I} = \emptyset$. Then $\mathcal{X}_{t_b,t_e} \cap \mathcal{I} \subseteq \bar{\Omega}_{t_b,t^*} \cap \mathcal{I}$. Using the approach in [13], this involves computing support function bounds $s^+_{\ell_i,\varepsilon_i}(t)$ for a sequence of directions ℓ_i, in which the error bound ε_i is progressively reduced until a given threshold is reached. The following examples shall illustrate the principle.

EXAMPLE 4.4. *Let $\mathcal{I}$ be the halfspace $\mathcal{I} = \{a^\mathsf{T} x \leq b\}$. According to Ex. 3.2, $\mathcal{X}_t \cap \mathcal{I} = \emptyset$ if and only if $-\rho_{\mathcal{X}_t}(-a) > b$. We start with an arbitrary time horizon T_0 and large initial accuracy ε_0, and determine a time horizon T with the following sequence starting from $k = 0$. In iteration k, we compute the support function bound $s^+_{-a,\varepsilon_k}(t)$ over the time domain $[0, T_k]$. If for all $0 \leq t \leq T_k$, $s^+_{-a,\varepsilon_k}(t) \leq b$, the flowpipe does not fully leave the invariant and we may choose to stop with $T = T_k$ (obtaining a non-exhaustive flowpipe approximation), or choose to increase the horizon to $T_{k+1} = 2T_k$. Otherwise, we continue the sequence with*

$$T_{k+1} = \min\{\, t \mid s^+_{-a,\varepsilon_k}(t) > b \,\},$$

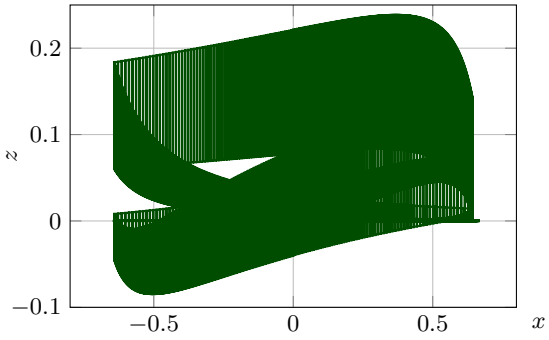
$\varepsilon_{k+1} = \varepsilon_k/10$, and $k \leftarrow k + 1$. If ε_k is smaller than a given threshold $\varepsilon_{\min}$, we stop and use the time horizon $T = T_k$ for the flowpipe approximation. If $\mathcal{I}$ is a polyhedron, we detect the time horizon for each constraint separately and choose the smallest one.

5. EXPERIMENTAL RESULTS

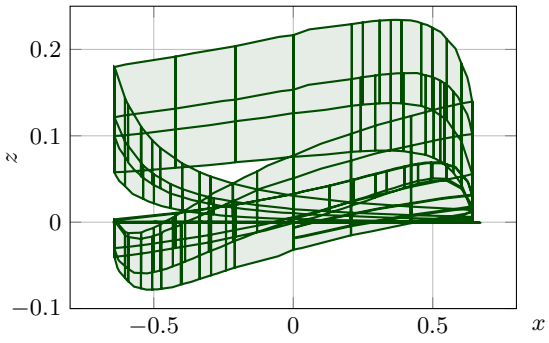
The space-time reachability algorithm is implemented as the *STC scenario* in the SpaceEx tool platform [17] and available for download [11]. We now give experimental results for several examples, some of which have been presented in [26]. The experiments were run using a virtual machine on a standard laptop.

5.1 Filtered Oscillator

This example from [17] consists of a switched oscillator system with signal x in series with M first-order filters that produce the output z. The system has $M+2$ continuous variables. Figure 5 shows the reachable states computed by the state-space template approach in [17], called LGG scenario, with that of the space-time approach in the STC scenario for $M = 32$. Template directions were chosen to be box directions (positive and negative axes) as well as the guard normals. The bound on the total error was given as $\varepsilon = 0.01$ for the STC, and for the LGG we chose an error tolerance that results in a similar discretization per template direction (in LGG, the error tolerance does not imply a hard bound on the error). LGG uses template clustering, in which the flowpipe segment intersecting the guard is approximated with a single template polyhedron. The optimal clustering of STC is of higher precision compared to LGG. Table 1 shows the performance of a full fixed-point computation for varying M. It indicates the # of continuous variables, the # of template



(a) state-space reachability



(b) space-time reachability

Figure 5: Projection of the reachable states of a filtered oscillator with 34 continuous variables

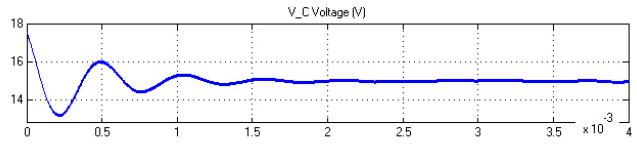
Table 1: Performance results for the filtered oscillator benchmark, varying the number of variables

		LGG			*STC*		
Var.	Dir.	T. [s]	#Post	T/#P	T. [s]	#Post	T/#P
6	12	0.1	6	0.02	0.1	6	0.03
18	36	1.0	10	0.10	1.8	21	0.09
34	68	4.3	15	0.29	5.4	27	0.20
66	132	27.2	24	1.13	45.6	53	0.86
130	260	388.0	92	4.22	182.2	43	4.24

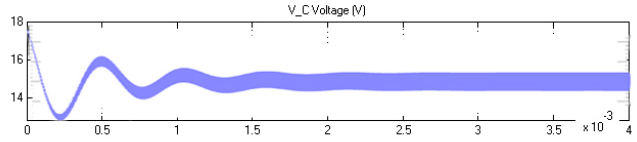
directions, the runtime, the # of post operations on symbolic states, and the time per post. The table shows that the time per post operation is comparable between LGG and STC. Because STC takes the convexification error into account, it may choose a higher precision for certain directions and therefore results in an overall better accuracy. The higher precision combined with better containment checking leads to the STC algorithm terminating after fewer iterations, and therefore to a lower runtime. LGG constructs 6453 hyperboxes (since box directions were used), while STC produces 210 polyhedra (not necessarily boxes). The STC algorithm simplifies the support function after convexification, as discussed at the end of Sect. 4.2. Without simplification, it takes about twice as long.

5.2 DC-DC Converter

This example is a model of a *DC-to-DC switched-mode power converter* described in [27], with continuous dynamics specified by linear ODEs. A DC-to-DC converter transforms a DC source voltage from one voltage level to another by switching at a high frequency between low and high voltage levels. The 4 continuous variables are the current and voltage in the circuit, a timer for the switching frequency, and a global clock. The system is subjected to significant additive noise.



(a) simulation with Simulink



(b) reachable set

Figure 6: Simulation and reachable set for the voltage of the DC-DC converter

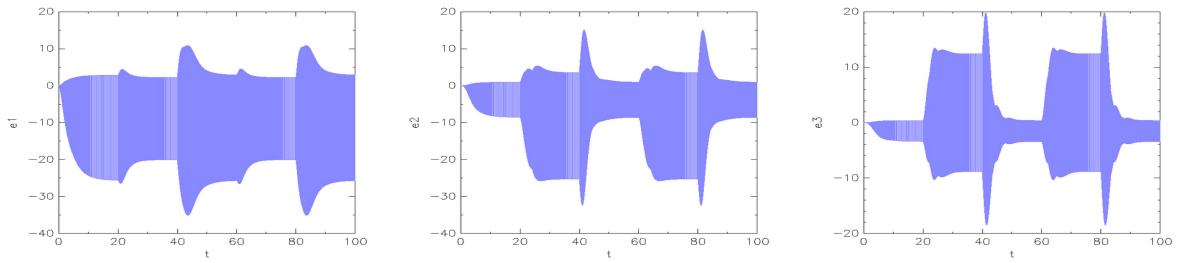
Figure 6(a) shows a simulation trace, computed with Matlab/Simulink, of the voltage level over a time horizon of $0.004\,s$. The additive noise is modeled in Simulink with a Gaussian random signal. In SpaceEx, we start from a single initial state and take the noise to be nondeterministic within a given interval. Figure 6(b) shows the reachable states obtained with error bound 0.01 and octagonal template directions (coefficients ± 1 in at most two of the variables). This benchmark is challenging for the STC algorithm since it switches frequently, every $25\,\mu s$, compared to the continuous dynamics, whose time constants are on the order of milliseconds. The computation involves 1000 one-step successors and takes about $900\,s$. In this example, the approximation error incurred during jumps tends to quickly accumulate. The LGG algorithm diverges for this system within 50 steps, i.e., within 1/20th of the time horizon.

5.3 Networked Platoon

This benchmark consists of a platoon of three controlled vehicles with a manually driven leader vehicle, which can break and accelerate within a given range [25]. The vehicles exchange information via a communication network that may be subjected to total loss of communication after $c_1\,s$, with communication being restored after $c_2\,s$. The ith vehicle is modeled by the deviation e_i between the distance to its predecessor and a fixed reference, and its velocity and acceleration relative to its predecessor. The resulting model has ten continuous variables. The goal is to determine the minimum allowable safe gaps among the vehicles. We consider the case with possible failure and parameters $c_1 = c_2 = 20\,s$. The reachable set, shown in Fig. 7, was computed in 182 s with the STC algorithm, using error bound $\varepsilon = 0.1$ and box directions.

6. ACKNOWLEDGMENTS

This work was partly supported by the European Commission under grant 643921 (UnCoVerCPS) and by the Institut Carnot–Logiciels et Systémes Intelligents.



(a) $e_1 \geq -35.442m$ (b) $e_2 \geq -32.759m$ (c) $e_3 \geq -19.078m$

Figure 7: Reachable states of the distance deviations e_1, e_2, and e_3 in the platoon benchmark, over time

7. REFERENCES

[1] H. Abbas and G. Fainekos. Linear hybrid system falsification through local search. In *ATVA'11*. Springer, 2011.

[2] M. Althoff. Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In *HSCC'13*, pages 173–182. ACM, 2013.

[3] M. Althoff and B. H. Krogh. Avoiding geometric intersection operations in reachability analysis of hybrid systems. In *Hybrid Systems: Computation and Control (HSCC'12)*, pages 45–54. ACM, 2012.

[4] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

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

[6] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Taylor model flowpipe construction for non-linear hybrid systems. In *RTSS*, pages 183–192. IEEE Computer Society, 2012.

[7] M. A. Colón and S. Sankaranarayanan. Generalizing the template polyhedral domain. In *Programming Languages and Systems*. Springer, 2011.

[8] T. Dang, T. Dreossi, and C. Piazza. Parameter synthesis using parallelotopic enclosure and applications to epidemic models. In *Int. Ws. Hybrid Systems and Biology HSB'14*, LNBI. Springer, 2014.

[9] T. Dang and R. Testylier. Reachability analysis for polynomial dynamical systems using the bernstein expansion. *Reliable Computing*, 17(2):128–152, 2012.

[10] P. S. Duggirala and A. Tiwari. Safety verification for linear systems. In *EMSOFT'13*. IEEE, 2013.

[11] G. Frehse. SpaceEx – State Space Explorer. Univ. Grenoble Alpes – Verimag, http://spaceex.imag.fr, 2015.

[12] G. Frehse and M. Althoff, editors. *1st Workshop on Applied Verification for Continuous and Hybrid Systems (ARCH)*. http://cps-vo.org/group/ARCH/benchmarks, 2014.

[13] G. Frehse, S. Bogomolov, M. Greitschus, T. Strump, and A. Podelski. Eliminating spurious transitions in reachability with support functions. In *Hybrid Systems: Computation and Control (HSCC'15)*, pages 149–158. ACM, 2015.

[14] G. Frehse, A. Hamann, S. Quinton, and M. Wöhrle. Formal Analysis of Timing Effects on Closed-loop Properties of Control Software. In *35th IEEE Real-Time Systems Symposium 2014 (RTSS)*, Rome, Italy, Dec. 2014.

[15] G. Frehse, R. Kateja, and C. Le Guernic. Flowpipe approximation and clustering in space-time. In *HSCC'13*, pages 203–212. ACM, 2013.

[16] G. Frehse, B. H. Krogh, and R. A. Rutenbar. Verifying analog oscillator circuits using forward/backward abstraction refinement. In *Design, Automation and Test in Europe (DATE'06)*, pages 257–262, 2006.

[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*, pages 379–395, 2011.

[18] G. Frehse and R. Ray. Flowpipe-guard intersection for reachability computations with support functions. In *IFAC ADHS*, pages 94–101, 2012.

[19] A. Girard. Reachability of uncertain linear systems using zonotopes. In M. Morari and L. Thiele, editors, *HSCC*, volume 3414 of *LNCS*, pages 291–305. Springer, 2005.

[20] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. *IEEE Trans. Automatic Control*, 43:540–554, 1998.

[21] W. Kühn. Rigorously computed orbits of dynamical systems without the wrapping effect. *Computing*, 61:47–67, September 1998.

[22] A. B. Kurzhanski and P. Varaiya. *Dynamics and Control of Trajectory Tubes*. Springer, 2014.

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

[24] A. V. Lotov, V. A. Bushenkov, and G. K. Kamenev. *Interactive Decision Maps*, volume 89 of *Applied Optimization*. Kluwer, 2004.

[25] I. B. Makhlouf and S. Kowalewski. Networked cooperative platoon of vehicles for testing methods and verification tools. In Frehse and Althoff [12].

[26] S. Minopoli and G. Frehse. Running spaceex on the ARCH14 benchmarks. In *ARCH'15*, 2015.

[27] L. V. Nguyen and T. T. Johnson. Dc-to-dc switched-mode power converters. In Frehse and Althoff [12].

[28] A. Pereira and M. Althoff. Safety control of robots under computed torque control using reachable sets. In *IEEE Int. Conf. Robotics and Automation*, 2015.

[29] P. Prabhakar and M. Viswanathan. A dynamic algorithm for approximate flow computations. In *HSCC'11*, 2011.

[30] S. Sankaranarayanan, T. Dang, and F. Ivančić. Symbolic model checking of hybrid systems using template polyhedra. In *TACAS'08*, pages 188–202. Springer, 2008.

[31] P. Varaiya. Reach set computation using optimal control. In *Proc. KIT Workshop*, pages 377–383, 1997.

[32] A. Zutshi, J. V. Deshmukh, S. Sankaranarayanan, and J. Kapinski. Multiple shooting, cegar-based falsification for hybrid systems. In *EMSOFT'14*, page 5. ACM, 2014.