Set Propagation Techniques for Reachability Analysis*

Matthias Althoff,[1] Goran Frehse,[2] and Antoine Girard[3]

[1]Department of Informatics, Technical University of Munich
Garching, Germany, 85748
email: althoff@tum.de

[2]Computer Science and Systems Engineering Laboratory (U2IS), ENSTA Paris
Palaiseau, France
email: goran.frehse@ensta-paris.fr

[3]Université Paris-Saclay, CNRS, CentraleSupélec
Laboratoire des signaux et systèmes
91190, Gif-sur-Yvette, France
email: antoine.girard@l2s.centralesupelec.fr

Abstract

Reachability analysis consists in computing the set of states that are reachable by a dynamical system from all initial states and for all admissible inputs and parameters. It is a fundamental problem motivated by many applications in formal verification, controller synthesis, and estimation, to name only a few. This paper focuses on a class of methods for computing a guaranteed over-approximation of the reachable set of continuous and hybrid systems, relying predominantly on set propagation: starting from the set of initial states, these techniques iteratively propagate a sequence of sets according to the system dynamics. After a review on set representation and computation, the paper presents the state of the art on set propagation techniques for reachability analysis of linear, nonlinear, and hybrid systems. The paper ends with a discussion on successful applications of reachability analysis to real world-problems.

Contents

*Posted with permission from the Annual Review of Control, Robotics, and Autonomous Systems, Volume 4

1 INTRODUCTION

Historically, analysis of dynamical systems has been done analytically through the use of mathematical notions such as transfer functions, Lyapunov functions, etc. As the complexity of engineered systems has grown considerably over the past decades, there has been a need to complement these analytical tools by computer-based techniques, which are able to deal for instance with high-dimensional state spaces or with hybrid (discrete/continuous) dynamics. The current industrial practice mostly relies on simulation and testing, which makes it possible to explore the behavior of

a system under various scenarios. While this approach is useful to detect undesirable unforeseen behaviors that need to be corrected, it suffers from the limitation that it is not possible to explore exhaustively all possible scenarios and that therefore it does not provide any formal guarantee that the system is correct.

In the early 1980s, research on model checking of discrete state dynamical systems [1, 2] was initiated with the objective of proving automatically that a system satisfies some requirements. The domain has had a great impact with many industrial successes, and is today extensively used in the hardware industry. Many algorithms used in model checking rely heavily on reachability analysis, which consists in computing the set of states that are reachable by the system, from a given set of initial states under all possible inputs. The extension of model checking and of reachability analysis to continuous [3] and hybrid systems [4] gained traction in the 1990s with the first tools being able to deal with nontrivial continuous dynamics, such as Checkmate [5] or d/dt [6], being released in the early 2000s. Since then, there has been continuous progresses in that research area with the development of more accurate and more scalable algorithms. The goal of this paper is to present the state of the art on reachability analysis of continuous and hybrid systems with an emphasis on a family of techniques based on propagation of sets.

1.1 Reachability Analysis of Continuous Systems

We first introduce the problem of reachability analysis in the context of continuous systems. The case of hybrid systems will be treated specifically in Section 5. In this paper, we use calligraphic capital letters to denote sets and bold letters to denote functions of time. Let us consider a continuous dynamical system modeled by a differential equation of the form:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{w}(t)),\ \mathbf{x}(t) \in \mathbb{R}^n,\ \mathbf{w}(t) \in \mathcal{W} \tag{1}$$

where $\mathbf{x}(t)$ denotes the state of the system and $\mathbf{w}(t)$ is an external input, considered in the following as a disturbance. The input values are assumed to belong to a given compact set $\mathcal{W} \subseteq \mathbb{R}^p$. We make the standing assumption that for a given initial state $x_0 \in \mathbb{R}^n$, for a given measurable input signal $\mathbf{w} : \mathbb{R}_0^+ \to \mathcal{W}$, the system (1) admits a unique trajectory defined on $\mathbb{R}_0^+$, denoted in the following by $\xi(., x_0, \mathbf{w})$.

The *reachable set* of (1) at time $t \in \mathbb{R}_0^+$, from a set of initial states $\mathcal{X}_0 \subseteq \mathbb{R}^n$ is

$$\mathsf{Reach}_t(\mathcal{X}_0) = \{\xi(t, x_0, \mathbf{w}) \mid x_0 \in \mathcal{X}_0,\ \mathbf{w}(s) \in \mathcal{W}, \forall s \in [0, t]\}. \tag{2}$$

The reachable set (or *reachable tube*) of (1) over a time interval $[0, T] \subseteq \mathbb{R}_0^+$ is then defined as

$$\mathsf{Reach}_{[0,T]}(\mathcal{X}_0) = \bigcup_{t \in [0,T]} \mathsf{Reach}_s(\mathcal{X}_0). \tag{3}$$

In words, the reachable set consist of all states that can be reached at time t or on the time interval $[0, T]$, from any initial state in $\mathcal{X}_0$, for any admissible external input. Reachability analysis consists in computing the reachable set of a dynamical system.

1.2 Difficulty and Usefulness

The question whether a state can be reached by a dynamical system is known to be tricky. In fact, for many types of systems one can show that there can be no algorithm that decides the

question in a finite number of steps – this kind of problem is called undecidable. The decidability of reachability depends on the kind of dynamics and on the constraints on inputs and initial states. For instance, reachability is decidable for a discrete-time linear time-invariant system if the set of inputs is unconstrained. However, if the inputs are constrained to a finite union of affine subspaces the problem becomes undecidable [7]. In the case of hybrid automata, reachability is undecidable even for classes with very simple dynamics, such as systems with piecewise constant derivatives [8]. In some cases, the problem becomes decidable if one restricts the analysis to a finite time horizon [9]. For this reason, one usually resorts to approximation methods. However, in order to be able to reason formally on the system using these approximations, one needs these approximation to present some guarantees. For instance, we can request that the computed approximation contains, or is contained by, the true reachable set. In that case, we talk about over-approximations and under-approximations, respectively. The focus of the present paper is on the computation of over-approximations, which are more often used in practice, but several works on under-approximations exist, see e.g. [10] and the references therein. Computing approximations of reachable sets can be useful for many purposes:

- **Formal verification**: If an over-approximative reachable set does not intersect an unsafe set, one can prove that it cannot be entered by any trajectory. If more complicated properties such as liveness or fairness are formulated, e.g., in temporal logics, one can sometimes synthesize a corresponding monitor automaton. By using parallel composition, the monitor and the system to be verified result in a hybrid automaton for which one has to check whether a bad state of the monitor can be reached. Thus, the verification problem can be translated into a reachability problem.

- **Computation of invariant sets and regions of attractions**: If a state starts in an invariant set, it will remain in it indefinitely [11]. For instance, invariant sets are useful to ensure stability of model predictive control. While one can obtain invariant sets from Lyapunov functions, in many cases one obtains better results when using reachable sets [12]. The region of attraction of a steady state contains all states that asymptotically converge to it. Similarly as for invariance sets, one often obtains better results when using reachability analysis compared to using Lyapunov functions [13].

- **Robust control**: Reachability analysis can be incorporated in controllers to ensure that a goal region is reached while unsafe regions are avoided and input and state constraints are respected. Especially for model predictive control, robust variants using reachability analysis have emerged [14, 15, 16].

- **Set-based observers and fault detection**: In safety-critical systems, it does not suffice to estimate the internal state using a standard approach, such as a Kalman filter. One rather requires a set of states in which the true state is guaranteed to be found. Many set-based observers propagate the set of states using reachability analysis and then constrain these sets using new sensor information [17]. Sets of estimated states are also used to reduce the false alarm rate in fault detection [18, 19].

- **Set-based prediction**: Prediction algorithms are often used by autonomous systems to avoid conflicts with surrounding entities. To ensure safety, one has to predict all possible behaviors of these entities using reachable sets, which serve as time-varying unsafe sets for verification purposes [20, 21].

- **Controller synthesis**: Reachability analysis is often used as a building block for synthesizing controllers from formal specifications. One application is to compute discrete abstractions of continuous and hybrid systems so that synthesis methods for discrete systems can be applied [22]. To avoid the state-explosion problem, one can also directly synthesize controllers in the continuous space using reachable sets [23].
- **Conformance checking**: For the above methods, it is often crucial that all behaviors of a real system are captured by its corresponding model, ensuring that obtained results also hold in reality. This requires non-deterministic models with potentially uncertain inputs, initial states, and parameters. A special form of conformance checking—called reachset conformance checking—ensures that all recorded behaviors of a real system are captured by the abstract non-deterministic model [24, 25].

1.3 Scope of the Paper

There exist several types of approaches to reachability analysis. Some approaches rely on optimal control theory to show that the reachable set of a system corresponds to the zero sublevel set of the solution of a Hamilton-Jacobi partial differential equation [26, 27]. Approximations of the reachable set can then be obtained by solving this equation numerically. While this approach allows to cope with complex nonlinear dynamics with inputs, their computational cost increases sharply with the state dimensions, since partial differential equations are usually solved through discretizing the state space. Barrier certificates can also be used to provide an over-approximation of the reachable set using the zero sublevel set of a function [28]. Intuitively, a function is a barrier certificate if no trajectory of the system can evolve from negative to positive values of the function. It follows that if the zero sublevel set of a barrier certificate contains the initial states of the system, it also contains the reachable set. Barrier certificates can be a powerful tool for showing safety if the reachable states can be separated from the unsafe states by a suitable barrier function. Intuitively speaking, this depends on both the distance to the unsafe states and the geometric complexity of the reachable set. Since a single barrier certificate is typically insufficient to accurately represent a reachable set, a sequence of piecewise barrier functions for different time intervals is presented in [29]. Another class of methods rely on simulation and trajectory sensitivity analysis to cover the reachable set using a finite number of neighborhoods of individual trajectories [30, 31, 32, 33, 34]. Trajectory-based techniques have been successfully applied to complex nonlinear systems. However, in cases of event-based switching (hybrid systems) the trajectory branches into a tree whose size can grow rapidly with time. Decrease in complexity of this tree, by detecting redundant states, can be easier with set-propagation techniques. Another perspective on reachability analysis is to construct a logic formula that encodes whether a state is reachable [35, 36]. Expressing reachability as the satisfiability of a set of constraints can be advantageous in particular over a bounded time horizon. However, it requires bounding the reachable states with a finite number of constraints (each one geometrically simple), which is difficult if the reachable states are geometrically complex.

In this paper, we will not elaborate further on these approaches and we will focus on a class of methods that rely on set propagation. Starting from the set of initial states, these approaches compute iteratively a sequence of sets which correspond to propagations of the initial set according to the system dynamics. Set-propagation techniques can be seen as an extension of numerically solving the ordinary differential equations of the system, where the solution is expressed in terms of sets rather than numbers. The error is generally a function of the time step and of the size of

the initial states. In principle, both can be made arbitrarily small, possibly by splitting the initial states, so that the approximation can be made arbitrarily precise. The set-propagation techniques presented in this paper are by design conservative, i.e., they cover all possible solutions of the system. Set-propagation techniques can benefit from techniques that reduce the complexity, such as detecting previously explored states through containment checks, accelerating cycles through widening, and merging sets through hull operations. These properties have made set-propagation techniques popular in both academic research and industrial case studies.

The paper is organized as follows. In Section 2, we present several classes of sets that are suitable for computer representation and have favorable properties for reachability analysis. Then, Sections 3, 4, 5 will present set propagation techniques for over-approximation of the reachable sets for linear, nonlinear and hybrid systems. In Section 6, we review a number of applications where such techniques have shown useful.

Several software tools and packages are available for computing reachable states, most of them are specialized to a particular class of systems. Since a comprehensive listing would go beyond the scope of this paper, we refer the reader to the proceedings of the yearly ARCH verification competition, which provides an up-to-date overview [37].

2 SET REPRESENTATIONS

To efficiently and accurately compute over-approximations of the reachable sets, one important question is the representation of sets. We provide below a taxonomy of sets (see Figure 1). At the coarser level, we distinguish between convex and non-convex sets. In this section, we discuss how these sets are represented on a computer and which types of operations can be effectively computed. We use $C_{(\cdot,i)}$ to denote the ith column vector of a matrix C.

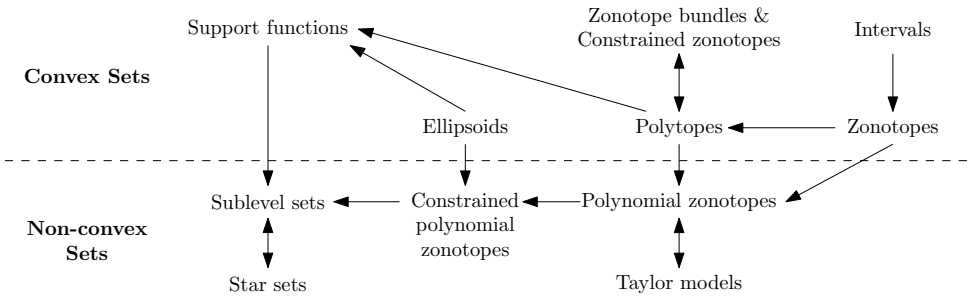

Figure 1: Visualization of the relations between the different set representations, where A $\rightarrow$ B denotes that B contains A.

2.1 Convex Sets

Convex sets are attractive for their geometric simplicity and computational efficiency of many subclasses. Moreover, they are particularly suitable for reachability analysis of linear systems, since convexity of reachable sets at any instant is preserved under linear dynamics. In that case, one usually chooses to work with a subclass of compact convex sets, which can be easily represented on a computer, and for which elementary set operations can be effectively computed. The most

common set operations that are needed in that respect are linear transformations, Minkowski sum, and convex hull.

2.1.1 Ellipsoids, Polytopes and Zonotopes

Definition 1 (Ellipsoid)**.** *Given a center $c \in \mathbb{R}^n$ and a positive definite symmetric matrix $Q \in \mathbb{R}^{n \times n}$, an ellipsoid is*

$$\mathcal{E} = \left\{ x \in \mathbb{R}^n \,\middle|\, (x-c)^\top Q (x-c) \leq 1 \right\}.$$

Definition 2 (Polytope)**.** *Given a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $d \in \mathbb{R}^m$, a $\mathcal{H}$-polyhedron is*

$$\mathcal{P} = \left\{ x \in \mathbb{R}^n \,\middle|\, Ax \leq b \right\}.$$

A $\mathcal{H}$-polytope is a bounded polyhedron. Given a finite set of vertices $\{v_1, \dots, v_r\} \subseteq \mathbb{R}^n$, a $\mathcal{V}$-polytope is $\mathcal{P} = \text{conv}(\{v_1, \dots, v_r\})$.

Definition 3 (Zonotope)**.** *Given a center $c \in \mathbb{R}^n$ and a generator matrix $G \in \mathbb{R}^{n \times p}$, a zonotope is*

$$\mathcal{Z} := \left\{ c + \sum_{i=1}^{p} \alpha_i G_{(\cdot,i)} \,\middle|\, \alpha_i \in [-1, 1] \right\}. \tag{4}$$

Ellipsoids can be represented by $n \times (n+3)/2$ real numbers. A $\mathcal{H}$-polytope can be represented by $m \times (n+1)$ real numbers. A $\mathcal{V}$-polytope can be represented by $r \times n$ real numbers. It is always possible to go from a $\mathcal{H}$-polytope to a $\mathcal{V}$-polytope, and conversely. The representation of a $\mathcal{H}$-polytope is usually more compact than its equivalent representation as a $\mathcal{V}$-polytope. A zonotope can be represented by $n \times (p+1)$ real numbers. In addition, it is always possible to go from a zonotope to an equivalent $\mathcal{H}$-polytope or $\mathcal{V}$-polytope, though the zonotope representation is usually much more compact.

2.1.2 Zonotope Bundles and Constrained Zonotopes

A major disadvantage of polytopes is that the Minkowski sum is computationally expensive. Since Minkowski sum is an essential operation in reachability analysis, alternative set representations based on zonotopes have been developed that can also represent polytopes, but are computationally cheaper for certain operations.

Definition 4 (Zonotope Bundle (see Def. 4 in [38]))**.** *Given a finite set of zonotopes $\mathcal{Z}_\cap$, a zonotope bundle is $\mathcal{Z}_\cap = \bigcap_{i=1}^{s} \mathcal{Z}_i$, i.e. the intersection of zonotopes $\mathcal{Z}_i$. Note that the intersection is not computed, but the zonotopes $\mathcal{Z}_i$ are stored in a list, which we write as $\mathcal{Z}_\cap = \{\mathcal{Z}_1, \dots, \mathcal{Z}_s\}^\cap$.*

Due to the lazy representation of polytopes as a set of intersecting zonotopes, Minkowski sum cannot be computed exactly, but can be tightly over-approximated in the sense that the result is always better than facet lifting [38, Prop. 2-4]. Also, convex hulls have to be over-approximated. An alternative to zonotope bundles are constrained zonotopes, which are computationally more demanding, but operations on them are exact for Minkowski sum and convex hull [18].

Definition 5 (Constrained Zonotope (see Def. 3 in [18])). *Given a center vector $c \in \mathbb{R}^n$, a generator matrix $G \in \mathbb{R}^{n \times p}$, a constraint matrix $A \in \mathbb{R}^{m \times p}$, and a constraint vector $b \in \mathbb{R}^m$, a constrained zonotope $\mathcal{CZ} \subset \mathbb{R}^n$ is*

$$\mathcal{CZ} := \left\{ c + \sum_{i=1}^{p} \alpha_i G_{(\cdot,i)} \;\middle|\; \sum_{i=1}^{p} \alpha_i A_{(\cdot,i)} = b, \quad \alpha_i \in [-1, 1] \right\}. \tag{5}$$

For a concise notation we use the shorthand $\mathcal{CZ} = \langle c, G, A, b \rangle_{CZ}$.

2.1.3 Support Functions

The support function of a compact convex set is determined by the position of the supporting hyperplane in a given direction.

Definition 6 (Support function). *Given a compact convex set $\mathcal{S} \subseteq \mathbb{R}^n$, the support function is defined for all $\ell \in \mathbb{R}^n$ as $\rho_{\mathcal{S}}(\ell) = \max_{x \in \mathcal{S}} \ell^\top x$*

Tight polyhedral over-approximations of $\mathcal{S}$ can be obtained by sampling its support function in a finite number of directions $\mathcal{L} \subseteq \mathbb{R}^n$:

$$\mathcal{S} \subseteq \bigcap_{\ell \in \mathcal{L}} \left\{ x \in \mathbb{R}^n \,\middle|\, \ell^\top x \leq \rho_{\mathcal{S}}(\ell) \right\}.$$

Moreover, $\mathcal{S}$ is uniquely determined by its support function since the above inclusion becomes an equality when $\mathcal{L} = \mathbb{R}^n$. The evaluation of the support function in a given direction requires to solve a convex optimization problem. Closed form solutions exist for ellipsoids and zonotopes, while for polytopes one needs to solve a linear program.

2.2 Non-convex Sets

Convexity of reachable sets for points in time is preserved for linear systems. However, reachable sets are in general no longer convex for nonlinear systems [39] so that non-convex set representations provide tighter over-approximations, see e.g. [40, 41].

2.2.1 Polynomial Zonotopes and Constrained Polynomial Zonotopes

We first present the more general class of constrained polynomial zonotopes.

Definition 7 (Constrained Polynomial Zonotope (see Def. 4) in [42]). *Given a starting point $c \in \mathbb{R}^n$, a generator matrix $G \in \mathbb{R}^{n \times h}$, an exponent matrix $E \in \mathbb{Z}_{\geq 0}^{p \times h}$, factors α_k, a constraint matrix $A \in \mathbb{R}^{m \times q}$, a constraint vector $b \in \mathbb{R}^m$, and a constraint exponent matrix $R \in \mathbb{Z}_{\geq 0}^{p \times q}$, a constrained polynomial zonotope is defined as*

$$\mathcal{CPZ} := \left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} \;\middle|\; \sum_{i=1}^{q} \left(\prod_{k=1}^{p} \alpha_k^{R_{(k,i)}} \right) A_{(\cdot,i)} = b, \; \alpha_k \in [-1, 1] \right\}. \tag{6}$$

Polynomial zonotopes are a special case of constrained polynomial zonotopes without constraints, except that $\alpha_k \in [-1, 1]$. Even though polynomial zonotopes are a special case, they can represent polytopes since 1) they are closed under convex hull (see Tab. 1), 2) a point is obviously a special case of a polynomial zonotope, and 3) the convex hull of points is a polytope. Polynomial zonotopes have first been introduced in [41] and later extended to the above-presented sparse representation [43].

2.2.2 Taylor Models

Taylor models are essentially n-dimensional Taylor polynomials enlarged by an n-dimensional interval $[x] := [\underline{x}, \overline{x}]$, $\forall i : \underline{x}_i \leq \overline{x}_i$, $\underline{x}, \overline{x} \in \mathbb{R}^n$.

Definition 8 (Taylor polynomial (see Sec. 3 in [44]))**.** *Let us first introduce the multi-index set*

$$\mathcal{L}^q = \left\{ (l_1, l_2, \ldots, l_n) \;\middle|\; l_i \in \mathbb{N}, \sum\nolimits_{i=1}^{n} l_i \leq q \right\}.$$

We define $P^q(x, x_0, p)$ as the q-th order Taylor polynomial of $f(x, p) : \mathbb{R}^n \times \mathbb{R}^{\tilde{p}} \to \mathbb{R}^m$ parameterized by $p \in \mathcal{P} \subset \mathbb{R}^{\tilde{p}}$ around x_0 ($x, x_0 \in \mathbb{R}^n$):

$$P^q(x, x_0, p) = \sum_{l \in \mathcal{L}^q} \frac{(x_1 - x_{0,1})^{l_1} \ldots (x_n - x_{0,n})^{l_n}}{l_1! \ldots l_n!} \left(\frac{\partial^{l_1 + \ldots + l_n} f(x, p)}{\partial x_1^{l_1} \ldots \partial x_n^{l_n}} \right) \Bigg|_{x = x0}. \tag{7}$$

Next, we define Taylor models and show how they can enclose arbitrary multi-dimensional functions.

Definition 9 (Taylor model (see Def. 1 in [45]))**.** *Let $f(x, p) : \mathbb{R}^n \times \mathbb{R}^{\tilde{p}} \to \mathbb{R}^m$ be a function that is $(q + 1)$ times continuously differentiable in an open set containing the n-dimensional interval $[x]$. Using the q-th order Taylor polynomial $P^q(x, x_0, p)$ of $f(x, p)$ around $x_0 \in [x]$, we choose an m-dimensional interval $[I]$ such that*

$$\forall x \in [x], p \in \mathcal{P} : \quad f(x, p) \in P^q(x, x_0, p) \oplus [I]. \tag{8}$$

The tuple $T = (P^q(x, x_0, p), [I], [x], \mathcal{P})$ fully specifies a q^{th} order Taylor model of $f(x, p)$ around x_0. In Sec. 4.1, Taylor models $T = (P^q(t, t_0, x_0), [I], [t], \mathcal{X}_0)$ are obtained with respect to t and parameterized by the initial state x_0 to enclose a reachable set starting in $x_0 \in \mathcal{X}_0$ for a given time interval $[t]$. Taylor models are per se not sets, but are used in Sec. 4.1 to obtain reachable sets through $\{P^q(t, t_0, x_0) | t \in [t], x_0 \in \mathcal{X}_0\} \oplus [I]$. When replacing α_k in Def. 7 by $x_k \in [-1, 1]$, one can easily see that Taylor models and polynomial zonotopes are equally expressive. However, Taylor models are only closed under linear map, convex hull, and quadratic maps if the interval $[I]$ is reduced to a point. Also, many operations, such as Minkowski sum, require renaming variables, which is not required when using polynomial zonotopes.

2.2.3 Sublevel Sets and Star Sets

Sublevel sets are a general concept used for various problems:

Definition 10 (Sublevel Set)**.** *Given a real valued function $\mu(x) : \mathbb{R}^n \to \mathbb{R}$, a sublevel set is defined as*

$$\mathcal{LS} := \{ x \mid \mu(x) \leq 0 \}, \tag{9}$$

where any constraint $\tilde{\mu}(x) \leq c$, $c \in \mathbb{R}$ can be transformed to the above form.

Sublevel sets are very general since $\mu(x)$ can be arbitrarily chosen. However, due to their general structure, it is hard to reduce their representation size. Star sets have a little bit more structure:

Definition 11 (Generalized Star Set (see Def. 1 in [46])). *Given the center $x_0 \in \mathbb{R}^n$, q vectors $v_1, v_2, \ldots, v_q$ (Def. 1 in [46] restricted q, but later publications removed this restriction) forming the basis, and a predicate $P : \mathbb{R}^n \to \top, \bot$, a generalized star set is defined as*

$$\mathcal{SS} := \left\{ x \;\middle|\; x = x_0 + \sum_{i=1}^{q} \alpha_i v_i, P(\alpha) = \top \right\}. \tag{10}$$

Since the predicate can be arbitrarily chosen, closed-form solutions for many set operations do not exist.

2.3 Comparison

In Table 1, we summarize the different set representations and their property regarding some geometrical transformations. For closed-form expression of operations, complexity results are provided with respect to the system dimension n. The computational complexity is obtained assuming that the resulting sets are not reduced to their minimum representation, e.g., redundant vertices of $\mathcal{V}$-polytopes and redundant halfspaces of $\mathcal{H}$-polytopes are not removed. Also, we only consider the number of required binary operations (i.e. operations where two operands are required). The computational effort of unary operations like concatenations of lists can be safely neglected. Please note that we do not assume special numerical tricks that have been developed for large matrices and consider the "schoolbook method". The linear map is defined by $M \in \mathbb{R}^{m \times n}$ while the quadratic map is defined as $\{[x^T Q_1 x] \times \ldots \times [x^T Q_n x] | x \in \mathbb{S}\}$, $Q \in \mathbb{R}^{n \times n \times n}$.

From Table 1, one can see that for reachability analysis of linear systems that relies heavily on linear maps and Minkowski sum, convex representations such as zonotopes or support functions possess interesting computational features. However, when the dynamics present strong nonlinearities, the accuracy provided by convex representations may not be sufficient anymore and non-convex alternatives such as (constrained) polynomial zonotopes can be considered.

3 LINEAR SYSTEMS

In this section, we focus on reachability analysis of continuous dynamical systems with linear dynamics:

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{w}(t),\ \mathbf{x}(t) \in \mathbb{R}^n,\ \mathbf{w}(t) \in \mathcal{W} \subseteq \mathbb{R}^m \tag{11}$$

It is common to assume that the set of inputs $\mathcal{W}$ and of initial states $\mathcal{X}_0 \subseteq \mathbb{R}^n$ are compact convex sets. Then, for all $t \in \mathbb{R}_0^+$, the reachable set $\mathsf{Reach}_t(\mathcal{X}_0)$ is also a compact convex subset of $\mathbb{R}^n$. For this reason, it is generally sufficient, for linear systems such as (11) to over-approximate the set $\mathsf{Reach}_{[0,T]}(\mathcal{X}_0)$ by a union of convex sets. One usually uses one of the class of convex sets presented in

[1] Number of halfspaces only bounded by $\binom{2^n}{n}$ [56].
[2] Intersection has to be performed by facet and vertex enumeration.
[3] Requires coordinate transformation as for $\mathcal{H}$-polytopes.
[4] $\mu([x_1, x_2]) = \max(\mu_1(x_1), \mu_2(x_2))$ ($\mu(x)$ as in Def. 10)
[5] $\mu(x) = \max(\mu_1(x), \mu_2(x))$ ($\mu(x)$ as in Def. 10)
[6] $\mu(x) = \min(\mu_1(x), \mu_2(x))$ ($\mu(x)$ as in Def. 10)

Table 1: Comparison of set representations for unreduced results (−: closed under set operation, but no closed-form expression; x: not closed under set operation; †: only if M is square and full rank).

Class	**Linear map**	**Minkowski sum**	**Cartesian product**	**Convex hull**	**Quadratic map**	**Intersection**	**Union**
Intervals	x	$\mathcal{O}(n)$ [47, (2.67)]	$\mathcal{O}(1)$ [47, (2.59)]	x	x	$\mathcal{O}(n)$ [47, (2.63)]	x
Ellipsoids	$\mathcal{O}(\max(mn^2, m^2n))$ [48, Sec. 2.2.1]	x	x	x	x	x	x
$\mathcal{H}$-polytopes (spanned by n generators)	$\mathcal{O}(n^3)$ † [49, Tab. 1]	$\mathcal{O}(2^n)$ [49, Tab. 1]	$\mathcal{O}(1)$ [50, Sec. 2]	super-polynomial [1] [51, Thm. 2]	x	$\mathcal{O}(1)$ Def. 2	x
$\mathcal{V}$-polytopes (spanned by n generators)	$\mathcal{O}(mn2^n)$ [49, Tab. 1]	$\mathcal{O}(n2^{2n})$ [49, Tab. 1]	$\mathcal{O}(1)$ Def. 2	$\mathcal{O}(1)$ [50, Sec. 2]	x	super-polynomial [2] [52, Ch. 6.1]	x
Zonotopes (n generators)	$\mathcal{O}(mn^2)$ [53, Tab. I]	$\mathcal{O}(n)$ [53, Tab. I]	$\mathcal{O}(1)$ Def. 3	x	x	x	x
Zonotope Bundles (n generators)	$\mathcal{O}(n^3)$ † [38, Prop. 1]	-	$\mathcal{O}(1)$ Def. 4	-	x	$\mathcal{O}(1)$ [38, Def. 4]	x
Constrained Zonotopes (n generators)	$\mathcal{O}(mn^2)$ [18, (11)]	$\mathcal{O}(n)$ [18, (12)]	$\mathcal{O}(1)$ Def. 5	-	x	$\mathcal{O}(n)$ [18, (13)]	x
Support Functions (n directions)	$\mathcal{O}(mn^2)$ [53, Tab. II]	$\mathcal{O}(n)$ [53, Tab. II]	$\mathcal{O}(n)$ [49, Sec. 2.1]	$\mathcal{O}(n)$ [53, Tab. II]	x	-	x
Polynomial Zonotopes/Taylor Models	$\mathcal{O}(mn^2)$ [43, Sec. 2.4]	$\mathcal{O}(n)$ [42, Prop. 7]	$\mathcal{O}(1)$ [42, Prop. 8]	$\mathcal{O}(n^2)$ [42, Prop. 9]	$\mathcal{O}(n^4)$ [43, Sec. 2.4]	x	x
Constrained Polynomial Zonotopes	$\mathcal{O}(mn^2)$ [42, Prop. 6]	$\mathcal{O}(n)$ [42, Prop. 7]	$\mathcal{O}(1)$ [42, Prop. 8]	$\mathcal{O}(n^2)$ [42, Prop. 9]	$\mathcal{O}(n^4)$ [42, Prop. 10]	$\mathcal{O}(n)$ [42, Prop. 11]	$\mathcal{O}(n)$ [42, Thm. 1]
Sublevel Sets	$\mathcal{O}(n^3)$ † [3] Def. 10	-	$\mathcal{O}(1)$ [4] Def. 10	-	-	$\mathcal{O}(1)$ [5] Def. 10	$\mathcal{O}(1)$ [6] Def. 10
Star Sets (n generators)	$\mathcal{O}(mn^2)$ [54, Prop. 3.2]	$\mathcal{O}(n)$ [55, Sec. 3.1]	$\mathcal{O}(1)$ [55, Def. 3]	-	-	-	-

Section 2.1: ellipsoids [57, 58], polytopes [59, 60], zonotopes [61, 62] or support functions [63, 64]. In the following we focus on time-invariant systems but similar techniques exist for linear time-varying or uncertain systems [65]

3.1 Time Discretization

Most of the approaches for computing over-approximations of the reachable set of (11) are based on time discretization. Let $N \in \mathbb{N}$, and $\tau = T/N$, from the semi-group property of differential equation (11), it follows that:

$$\mathsf{Reach}_{[0,T]}(\mathcal{X}_0) = \bigcup_{k=0}^{N-1} \mathsf{Reach}_{k\tau}(\mathsf{Reach}_{[0,\tau]}(\mathcal{X}_0)).$$

The main idea is to compute a sequence of compact convex sets $(\mathcal{S}_k)_{k=0}^{N-1}$ such that for all k, $\mathcal{S}_k$ contains $\mathsf{Reach}_{k\tau}(\mathsf{Reach}_{[0,\tau]}(\mathcal{X}_0))$. Several approximation schemes exist, we briefly describe the one from [63]. To initialize the sequence, one can use

$$\mathcal{S}_0 = \text{conv}\left(\mathcal{X}_0 \cup (e^{A\tau}\mathcal{X}_0 \oplus \tau\mathcal{W} \oplus \alpha(\tau, A, B, \mathcal{X}_0, \mathcal{W})\mathcal{B})\right)$$

where $\alpha(\tau, A, B, \mathcal{X}_0, \mathcal{W}) = \mathcal{O}(\tau)$ is a positive scalar and $\mathcal{B}$ is the unit ball for a norm of $\mathbb{R}^n$. Then, the remaining elements of the sequence can be obtained using the recurrence equation for $k = 1, \ldots, N-1$:

$$\mathcal{S}_k = e^{A\tau}\mathcal{S}_{k-1} \oplus \tau\mathcal{W} \oplus \beta(\tau, A, B, \mathcal{W})\mathcal{B} \tag{12}$$

where $\beta(\tau, A, B, \mathcal{W}) = \mathcal{O}(\tau^2)$ is a positive scalar. The analytic expressions of α and β can be found in [63]. Then, we obtain the following result where d_H denotes the Hausdorff distance between sets:

$$\mathsf{Reach}_{[0,T]}(\mathcal{X}_0) \subseteq \mathcal{R}_N \text{ and } d_H(\mathsf{Reach}_{[0,T]}(\mathcal{X}_0), \mathcal{R}_N) = \mathcal{O}(\tau), \text{ where } \mathcal{R}_N = \bigcup_{k=0}^{N-1} \mathcal{S}_k. \tag{13}$$

One should remark that computing the sequence $(\mathcal{S}_k)_{k=0}^{N-1}$ and thus the over-approximation $\mathcal{R}_N$ of the reachable set involves computing one convex hull, N linear transformations and N Minkowski sums whose computation has been discussed in the previous section for ellipsoids, polytopes, zonotopes, and support functions. However, one should note that the complexity of the representation of $\mathcal{S}_k$ increases as k grows. For instance, if one uses a zonotope to represent $\mathcal{S}_k$, the size of the generator matrix increases linearly with k. Then, the algorithm to compute $\mathcal{R}_N$ has memory and time complexity that is quadratic with respect to N.

A way to impose linear memory and time complexity is to add a reduction step at each iteration (see e.g. [61]). In that case, the recurrence equation (12) becomes for $k = 1, \ldots, N-1$:

$$\mathcal{S}_k = \mathsf{reduce}\left(e^{A\tau}S_{k-1} \oplus \tau\mathcal{W} \oplus \beta(\tau, A, B, \mathcal{W})\mathcal{B}\right) \tag{14}$$

where reduce is a function which over-approximates a convex set (e.g. a zonotope) by a convex set of given complexity (e.g. a zonotope with a generator matrix of given size). With this approach, the complexity becomes linear with respect to N but at the expense of additional conservatism. In particular, the approximation estimate in the Hausdorff distance provided in (13) is not valid any more. Moreover, while with recurrence equation (13), the approximation error can be reduced by increasing N, it is generally not the case any more with recurrence equation (14). This phenomenon is referred to as the *wrapping effect* [66].

3.2 Computing without the Wrapping Effect

For linear time-invariant systems such as (11), it is actually possible to reschedule the computations of the sequence $(\mathcal{S}_k)_{k=0}^{N-1}$ given by (12) to reduce the complexity without additional conservatism, as presented in [67]. For that purpose, let us introduce auxiliary sequences $(\mathcal{X}_k)_{k=0}^{N-1}$, $(\mathcal{W}_k)_{k=0}^{N-1}$, $(\mathcal{Y}_k)_{k=0}^{N-1}$ given by

$$\mathcal{Z}_0 = \text{conv}\left(\mathcal{X}_0 \cup (e^{A\tau}\mathcal{X}_0 \oplus \tau\mathcal{W} \oplus \alpha(\tau, A, B, \mathcal{X}_0, \mathcal{W})\mathcal{B})\right),\ \mathcal{V}_0 = \tau\mathcal{W} \oplus \beta(\tau, A, B, \mathcal{W})\mathcal{B},\ \mathcal{Y}_0 = \{0\},$$

and the recurrence equation for $k = 1, \ldots, N-1$:

$$\mathcal{Z}_k = e^{A\tau}\mathcal{Z}_{k-1},\ \mathcal{V}_k = e^{A\tau}\mathcal{V}_{k-1},\ \mathcal{Y}_k = \mathcal{Y}_{k-1} \oplus \mathcal{V}_{k-1}.$$

Then, one can verify that for $k = 0, \ldots, N-1$, $\mathcal{S}_k = \mathcal{Z}_k \oplus \mathcal{Y}_k$. The main advantage of this approach is that contrarily to (12), the complexity of the sets to which the linear maps are applied does not increase with k. By computing these sequences with zonotopes [67] or with support functions [63], it is actually possible to compute the over-approximation $\mathcal{R}_N$ of the reachable set with memory and time complexity that is linear with respect to N.

3.3 High-dimensional Systems

In order to scale reachability analysis for linear systems for systems beyond 1000 state variables, the methods presented above typically do not suffice. Mainly two methods for scaling to larger systems have been investigated: reduction methods and methods based on decomposing the system dynamics. The first work using order reduction techniques for reachability analysis is probably [68]. The same authors later used Krylov subspace reduction methods [69], which preserve the entire reachable set in contrast to [68]. While non-Krylov order reduction techniques are still explored for reachability analysis [70], Krylov methods have been most successful [55, 71]. Recently, Krylov methods for reachability analysis have been extended such that arbitrarily varying inputs can be considered [72].

In decomposition techniques, the system is divided up into subsystems. For each subsystem, the reach tube is computed by treating variables from the other subsystems as bounded inputs [73]. Finally, the reach tubes of the subsystems are combined, e.g., embedded back into full-dimensional space and intersected. For linear systems, one can use a coordinate transform to bring the system to a block-triangular form (Schur form) and the transformation matrix can be chosen to minimize the coupling terms [74]. However, the back-transformation into the full-dimensional space can amplify approximation errors. A decomposition without fully decoupling can be achieved by computing successor states in the subsystems and using the resulting state sets at each time step [49]. Taken to the extreme, decomposing the system into 1-dimensional subsystems, this is equivalent to computing the reachable states via interval arithmetic. For nonlinear systems, related projection techniques have been applied to level sets [75] and using two-dimensional polygonal projections [76]. To improve the accuracy, one can treat the computed reach tubes of subsystems as time-varying interval-shaped inputs in order to recompute the reach tubes with higher accuracy [77].

4 NONLINEAR SYSTEMS

The analysis of nonlinear systems is much more complicated since many valuable properties are no longer valid, such as the superposition principle and that the homogeneous solution can be computed by a linear map. We consider general nonlinear continuous systems with Lipschitz continuity. Using the same notation as for linear systems, the evolution of the state x is defined by the following differential equation:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{w}(t)), \qquad \mathbf{x}(0) \in \mathcal{X}_0 \subset \mathbb{R}^n, \quad \mathbf{w}(t) \in \mathcal{W} \subset \mathbb{R}^m. \tag{15}$$

As for linear systems, all subsequent approaches compute reachable sets for consecutive time intervals. Due to the difficulty of computing reachable sets of nonlinear systems, most approaches resort to abstraction techniques—either in the solution space or the state space; other techniques exploit specific system properties such as monotonicity. Finally, we briefly discuss the extension to differential-algebraic systems.

4.1 Abstraction in Solution Space

Abstractions in solution space approximate solutions of (15) by a Taylor polynomial (see Def. 8) with respect to time. Taylor polynomials approximating initial value problems of ordinary differential equations can be obtained by using the Picard iteration or by a truncated Lie series [40]. While the Taylor polynomial is most accurate for initial value problems with the initial state chosen as the expansion point, the solution is often well approximated for other initial states in its vicinity. To enclose the solution for a set of possible initial states and inputs, intervals are added to the Taylor series, resulting in Taylor models as presented in Def. 9. Taylor models have been originally developed by Makino and Berz [78, 79, 45]. An earlier development equivalent to Taylor models can be found in [80]. A common technique to obtain the interval $[I]$ for reachability problems is to guess $[I]$ so that evaluating the Picard operator for $(P^q(x, x_0), [I])$ yields a contractive Taylor model $(P^q(x, x_0), [I]_{post})$ with $[I]_{post} \subseteq [I]$ [40, Sec. III]. If the guess is not successful, $[I]$ is iteratively increased. These techniques have been further developed in [40] where Taylor models are not only computed from the initial point in time, but also from later points in time by obtaining the set of states of intermediate time points using standard set representations, such as polytopes or zonotopes [40].

4.2 Abstraction in State Space

Another abstraction technique is to abstract the system dynamics rather than its solution. In order to write subsequent abstractions in a compact way, we introduce $\mathbf{z} = [\mathbf{x}^T, \mathbf{w}^T]^T \in \mathbb{R}^o$ $(o = n + m)$ and the Nabla symbol $\nabla = \sum_{i=1}^{o} e^{(i)} \frac{\partial}{\partial \mathbf{z}_i}$, where $e^{(i)}$ are orthogonal unit vectors. The nonlinear system in (15) can be abstracted by a Taylor expansion of order κ at point z^* with Lagrange remainder $\mathcal{L}$ (see [78]):

$$\begin{aligned} &\dot{\mathbf{x}}_i(t) \in P^{\kappa}(\mathbf{z}(t), z^*) \oplus \mathcal{L}_i(t), \\ &\mathcal{L}_i(t) = \left\{ \frac{\left((\mathbf{z}(t) - z^*)^T \nabla\right)^{\kappa+1} f_i(\tilde{\mathbf{z}}(t))}{(\kappa+1)!} \middle| \tilde{\mathbf{z}}(t) = z^* + \alpha(\mathbf{z}(t) - z^*),\ \alpha \in [0,1] \right\}. \end{aligned} \tag{16}$$

In many cases, only first-order terms are considered so that the system is conservatively linearized. In a few cases, it is even possible to obtain a polynomial system without any abstraction error by a change-of-bases transformation [81, 82].

4.2.1 Time-Invariant Regions

The abstraction in (16) is only tight in the vicinity of the expansion point z^*. To expand the scope of tight abstractions, several works have split the state space into regions with different expansion points, resulting in hybrid systems as described in Sec. 5. In [83], nonlinear dynamics is abstracted by a constant value plus the abstraction error using polyhedral regions. A more precise abstraction is performed in [84, 85], where the original dynamics in linearized and the state space is decomposed into polyhedral regions; the abstraction error is determined either by the Lipschitz constant or by a bounded second derivative.

One disadvantage of static regions is that the number of required regions scales exponentially with the system dimension. This can be alleviated by only computing reachable regions on-the-fly [86]. Another disadvantage is that static regions require two expensive operations: intersections with guards and unifications of sets to avoid computing too many instances of reachable sets; this is addressed in Sec. 5.

4.2.2 Time-Variant Regions

To avoid computing intersections and unifications when using time-invariant regions, one can construct time-variant regions moving along with the reachable set of the current time interval. The simplest form of abstraction is to only consider the linear part of (16), see, e.g., [87, 88]. Several methods to obtain the abstraction error exist. One possibility is to evaluate the Lagrange remainder in (16) over-approximatively using interval arithmetic [87]. Another example is to obtain the abstraction error from linear interpolation at the vertices of simplicies [89, 88], requiring simplicies as a set representation. One can also obtain an abstraction error from a scalar error dynamics [90]; however, a scalar error dynamics over-approximating a multi-dimensional dynamics introduces substantial conservatism.

The disadvantage of linear abstractions is that their abstraction error becomes too large when the reachable set is not sufficiently contracting. Obviously, using a higher-order Taylor expansion reduces the abstraction error at the cost of having to compute the reachable set for polynomial differential equations. Since no analytical solution is known for polynomial differential equations, over-approximative conversions to non-deterministic polynomial difference equations have been proposed [91, 41]. Since the resulting sets are no longer convex, polynomial zonotopes are used in [41], while the earlier approach in [91] uses convex simplices. Another difference is that the nonlinear part of the polynomial is solved as a linear function of time in [91]. Thus, a solution space abstraction is integrated in a state-space abstraction, while [41] only relies on a state-space abstraction.

4.3 Computing Bounds

Instead of computing the whole set of reachable states, one can compute the bound of the reachable set. One of the earliest technique using this method is face lifting, where facets are pushed outwards to obtain reachable sets [92]. This is typically not beneficial when using the techniques from Sec. 4.1 and Sec. 4.2 since one only converts a n-dimensional problem to a $n-1$ dimensional problem at

the expense of propagating many regions of the surface, such as facets of a polytope. However, this idea makes it possible to use different methods summarized below.

4.3.1 Optimization-based Techniques

By parameterizing the bound of a reachable set, one can solve an optimization problem minimizing a performance criterium (e.g., volume) under the constraint that all solutions have to be enclosed. This has been performed for general nonlinear systems in [59] for bounds modeled as halfspaces. Also in [93], halfspaces are used, but a more tailored approach for polynomial differential equations is presented, where the optimization problem is abstracted to linear programming using Bernstein polynomials.

4.3.2 Bounds from Monotonicity

When the system dynamics is monotone, it is particularly easy to obtain the bounds of the reachable set.

Definition 12 (Monotone dynamics; see [94, Def. II.1])**.** *The system dynamics is monotone with respect to the initial state x_0 and input trajectories $\mathbf{w}$ when the following property holds for the solution $\xi(t, x_0, \mathbf{w})$:*

$$\forall i, j, t \geq 0 : x_{0,i} \leq \overline{x}_{0,i}, \mathbf{w}_j(t) \leq \overline{\mathbf{w}}_j(t) \implies \forall i, t \geq 0 : \xi_i(t; x_0, \mathbf{w}) \leq \xi_i(t; \overline{x}_0, \overline{\mathbf{w}}).$$

A constructive method to prove monotonicity is presented in [94, Prop. III.2]. From the definition follows directly that the upper bound of each state can be computed by $\xi_i(t; \overline{x}_0, \overline{\mathbf{w}})$ while analogously the lower bound can be computed by $\xi_i(t; \underline{x}_0, \underline{\mathbf{w}})$—however, exact solutions of those do not exist. For this reason, one can provide upper bounds of $\xi_i(t; \overline{x}_0, \overline{\mathbf{w}})$ and lower bounds of $\xi_i(t; \underline{x}_0, \underline{\mathbf{w}})$, using validated integration methods, such as interval Taylor methods [95]. These can be seen as a form of reachability computation when only a single initial state is provided. An extension for piecewise monotone systems is presented in [96] and for mixed monotone systems in [97]. If a system is not piecewise monotone or mixed monotone, one can generate monotone dynamics whose upper and lower bound enclose the reachable set [96, 98]; results can be further tightened by known constraints on states as shown in [98].

4.4 Nonlinear Differential Algebraic Systems

Besides nonlinear systems, there are also extensions for nonlinear differential algebraic equations. Most of the current literature on reachability analysis for nonlinear differential algebraic systems focuses on index-1 systems. While some work exists for methods solving Hamilton-Jacobi partial differential equations [99, 100], which are not particularly scalable, only few works use set propagation techniques. The work in [101] presented an approach using polyhedral set representations, which requires computationally expensive projections of the reachable set onto the constraint manifold. A scalable approach that does not require intersections is based on abstracting differential algebraic systems to linear differential inclusions [102].

5 HYBRID SYSTEMS

Hybrid systems are systems with both discrete states and continuous state variables. Various formalisms have been proposed to describe hybrid systems. Typically, the evolution of the continuous state variables is described through differential equations. The changes between discrete states can be described, e.g., by a finite state automaton [103]. Alternatively, changes in dynamics can be encoded through difference equations involving integer variables [104] or the continuous variables themselves [105]. The continuous dynamics in a hybrid system depend on its discrete state. Since the timing of transitions between the discrete states is a priori unknown, this is called event-based switching. The interaction between events and differential equations may lead to very complex behavior, even if the dynamics of the ODEs themselves are simple or even constant. In the following subsections, we briefly describe the formalism of hybrid automata and a high-level reachability algorithm. Then we present the major subclasses of hybrid automata, which are distinguished by their continuous dynamics, and discuss their particularities in more detail.

5.1 Hybrid Automata

Hybrid automata are the result of combining finite state automata with differential equations [4, 106]. The discrete states are represented by an automaton (finite state machine) and are referred to as *locations* $\mathsf{Loc} = \{\ell_1, \ldots, \ell_m\}$. A state (ℓ, x) of the hybrid automaton consists of a location ℓ and a *continuous state* $x \in \mathbb{R}^n$. The *state space* is $\mathsf{Loc} \times \mathbb{R}^n$. The states must at all time lie within a given subset of the state space called *invariant* or *staying condition* $\mathsf{Inv} \subseteq \mathsf{Loc} \times \mathbb{R}^n$.

We first describe the changes between discrete states, which can instantaneously modify the continuous variables. Changes between locations are described by *transitions* $\mathsf{Edg} \subseteq \mathsf{Loc} \times \mathsf{Lab} \times \mathsf{Loc}$, where each transition is associated with a label from a finite set Lab. The labels can be used to synchronize transitions when several automata are connected in parallel; for lack of space we refer the reader to [4]. For a given transition, a state x can jump instantaneously to any other state x' for which (x, x') satisfies the *jump relation* of the transition $e \in \mathsf{Edg}$, denoted as $\mathsf{Jump}(e) \subseteq \mathbb{R}^n \times \mathbb{R}^n$. For convenience, the jump relation is often described by a *guard* set $\mathcal{G}_e$ and a set-valued *reset* map R_e: any state $x \in \mathcal{G}_e$ can take the transition and can jump to any $x' \in R_e(x)$.

The continuous variables evolve with time according to a *flow relation* Flow, where for each location ℓ, $\mathsf{Flow}(\ell) \subseteq \mathbb{R}^n \times \mathbb{R}^n$. A state x can evolve with derivative $\dot{x}$ if $(\dot{x}, x) \in \mathsf{Flow}(\ell)$. Often, the flow relation is described by an ODE $\dot{x} = f(x)$ or by a differential inclusion. A solution $\xi(t)$ to the above ODE describes a (local) trajectory of the continuous state as long as its remains inside the invariant, i.e., for all $0 \le \tau \le t$, $\xi(\tau) \in \mathsf{Inv}(\ell)$.

Run Semantics: All behaviors of the system originate in a given set of *initial states* Init. Starting from a state $(\ell_0, x_0) \in \mathsf{Init}$, a *run* of the hybrid automaton is an alternating sequence of trajectories and jumps. Denoting at the i-th step the trajectory with $\xi_i(t)$ and the time on this trajectory with δ_i, a run of length N is the sequence

$$(\ell_0, x_0) \xrightarrow{\delta_0, \xi_0} (\ell_0, \xi_0(\delta_0)) \xrightarrow{\alpha_0} (\ell_1, x_1) \xrightarrow{\delta_1, \xi_1} (\ell_1, \xi_1(\delta_1)) \ldots \xrightarrow{\delta_N, \xi_N} (\ell_N, \xi_N(\delta_N)),$$

with $(\ell_0, x_0) \in \mathsf{Init}$, $\alpha_i \in \mathsf{Lab}$, that satisfies for $i = 0, 1, \ldots, N$:

1. *Trajectories:* $(\dot{\xi}_i(t), \xi_i(t)) \in \mathsf{Flow}(\ell)$ and $\xi_i(t) \in \mathsf{Inv}(\ell_i)$ for all $t \in [0, \delta_i]$, $\delta_i \ge 0$.
2. *Jumps:* $(\xi_i(\delta_i), x_{i+1}) \in \mathsf{Jump}(e_i)$, $e_i = (\ell_i, \alpha_i, \ell_{i+1}) \in \mathsf{Edg}$, and $x_{i+1} \in \mathsf{Inv}(\ell_{i+1})$.

A state (ℓ, x) is called *reachable* if it is a state of a run of any length. The above definition of the behaviors of a hybrid automaton is inherently nondeterministic. The automaton may change the location if it enters the guard set of a transition, but it may also continue in the same location provided that it keeps satisfying the invariant. This is called *may* semantics. An alternative is called *must* semantics, in which a transition must be taken as soon as possible, e.g., as soon as the system enters a guard set. Note that most simulation tools apply must semantics, while most reachability tools use may semantics. Some models can be translated from must to may semantics [107].

5.2 Reachability for Hybrid Automata

The set of reachable states of a hybrid automaton can be computed by executing the runs on sets of states [106, 4]. We show a simple but widely used algorithm that applies it to *symbolic states* $(\ell, \mathcal{P})$, which is a set of states that share the same location ℓ, with $\mathcal{P} \subseteq \mathbb{R}^n$ being the set of continuous states. Let W be a waiting list for the symbolic states to still be explored and let R be a list of symbolic states that are reachable. Let $\mathsf{Post}_C(\ell, \mathcal{P})$ be the one-step successors by time elapse and $\mathsf{Post}_D(\ell, \alpha, \ell', \mathcal{P})$ be the one-step successors of a jump with transition (ℓ, α, ℓ'):

$$\begin{aligned} \mathsf{Post}_C(\ell, \mathcal{P}) &= \big\{\xi(\delta) \;\big|\; \exists x \in \mathcal{P} : (\ell, x) \xrightarrow{\delta, \xi} (\ell, \xi(\delta))\big\}, \\ \mathsf{Post}_D(\ell, \alpha, \ell', \mathcal{P}) &= \big\{x' \;\big|\; \exists x \in \mathcal{P} : (\ell, x) \xrightarrow{\alpha} (\ell', x')\big\}. \end{aligned}$$

To check whether a successor state has already been explored, we need an operator visited$(R, \ell, \mathcal{P})$ that returns true if all the states in $(\ell, \mathcal{P})$ are already contained in the passed list R (more details below). The reachability algorithm proceeds as follows:

1. Let $\mathcal{Q}(\ell) := \big\{x \;\big|\; (\ell, x) \in \mathsf{Init}\big\}$. Compute $W := \big\{\mathsf{Post}_C(\ell, \mathcal{Q}(\ell)) \;\big|\; \ell \in \mathsf{Loc}\big\}$ and let $R := W$.
2. Pop $(\ell, \mathcal{P})$ from W.
3. For each $\alpha \in \mathsf{Lab}$, $\ell' \in \mathsf{Loc}$, compute $\mathcal{P}' = \mathsf{Post}_D(\ell, \alpha, \ell', \mathcal{P})$. If $\mathcal{P}' \neq \emptyset$ and visited$(R, \ell', \mathcal{P}')$ is false, compute $\mathcal{P}'' = \mathsf{Post}_C(\ell', \mathcal{P}')$ and add $(\ell', \mathcal{P}'')$ to the waiting list.
4. If $W = \emptyset$, terminate and return R. Otherwise, go to step 2.

The containment check using visited$(R, \ell', \mathcal{P}')$ avoids adding the same states over and over. An exact computation of this containment relationship is costly, since it requires checking whether P is in the union of all symbolic states in R that involve location ℓ. A simple heuristic is to to apply a pairwise check: Let visited$(R, \ell', \mathcal{P}')$ return true if there is some $(\ell', \mathcal{P})$ in R such that $\mathcal{P}' \subseteq \mathcal{P}$ and return false otherwise. This may of course create a situation where the algorithm iterates forever while an exact containment check would terminate, but in practice this seems to be a minor problem. The main bottleneck is the computational complexity of the one-step successor operations, Post_C and Post_D. In the next sections we will discuss how they can be computed efficiently, depending on the dynamics.

5.3 Piecewise Constant Dynamics

In a hybrid automaton with *piecewise constant dynamics* (PCD), the relationships between continuous variables are given by linear constraints and the derivatives are independent of the continuous

state [106] (for brevity, we give a slightly simplified definition). Despite having relatively simple flow relations, PCD can exhibit complex, even chaotic, behavior. For instance, they can model discrete-time LTI systems by setting all derivatives to zero and placing the LTI state update in a transition. PCD automata stand out since one-step successor computations can be computed exactly (assuming infinite precision arithmetic), which is not the case for most other classes in the literature.

In a PCD, all continuous sets and relations ($\mathsf{Inv}(\ell)$, $\mathsf{Init}(\ell)$, $\mathsf{Jump}(e)$) are polyhedra, which can be given by strict or non-strict linear inequalities. $\mathsf{Flow}(\ell)$ is of the form $\dot{x} \in \mathcal{P}$, where $\mathcal{P}$ is a polyhedron. If we restrict the invariants to closed constraints, we can construct an equivalent automaton for which all sets are convex polyhedra. Writing out Post_D as a predicate, we obtain

$$\mathsf{Post}_D(\ell, \alpha, \ell', \mathcal{P}) \quad = \big\{x' \;\big|\; \exists x \in \mathcal{P} : (x, x') \in \mathsf{Jump}(\ell, \alpha, \ell') \wedge x' \in \mathsf{Inv}(\ell')\big\}.$$

Since $\mathsf{Inv}(\ell)$ is convex, the time elapse operator Post_C can be written as

$$\mathsf{Post}_C(\ell, \mathcal{P}) \quad = \mathcal{P} \cup \big\{x' \;\big|\; \exists x \in \mathcal{P}, \dot{x} \in \mathsf{Flow}(\ell), t > 0 : x' = x + t\dot{x} \wedge x' \in \mathsf{Inv}(\ell)\big\}.$$

Looking at Post_D and Post_C from the point of view of geometric operations, the quantifier elimination can be seen as a projection (an irreversible linear map) and conjunction as intersection. Both can be implemented using the operations discussed in Sect. 2.1. For efficiency, a *dual representation* for polyhedra combines both $\mathcal{H}$-polyhedra and $\mathcal{V}$-polyhedra ($\mathcal{V}$-polytopes extended with rays to represent unbounded sets) [108]. Further improvements in representation and targeted algorithms for both Post_D and Post_C can lead to significant speed-ups [109].

Termination: Computing the reachable states using the above operators frequently does not terminate, even in cases where the computed set of states converge to a fixed point. To induce termination, one can resort to overapproximations in the hope that the enlarged set is either already a fixed point or converges more rapidly towards one [110]. In *widening* [111], one looks at subsequent iterations and removes constraints that are relaxed from one iteration to the next. Alternatively, one can modify constraints by quantizing their coefficients, or remove them based on other criteria [112].

5.4 Piecewise Affine Dynamics

In a hybrid automaton with *piecewise affine dynamics* (PWA), the flow relation $\mathsf{Flow}(\ell)$ defines affine dynamics as in (11) [7] :

$$\dot{\mathbf{x}} = A_\ell \mathbf{x} + B_\ell \mathbf{w}_\ell,\ \mathbf{w}_\ell \in \mathcal{W}_\ell \subseteq \mathbb{R}^{m_\ell}. \tag{17}$$

The jump relation for transition e is given by a guard set $\mathcal{G}_e$ and an affine reset map with nondeterministic inputs :

$$\mathbf{x}' = R_e \mathbf{x} + S_e \mathbf{w}_e,\ \mathbf{x} \in \mathcal{G}_e,\ \mathbf{w}_e \in \mathcal{W}_e \subseteq \mathbb{R}^{m_e}.$$

Note that the constraints $\mathbf{w}_\ell \in \mathcal{W}_\ell$ and $\mathbf{w}_e \in \mathcal{W}_e$ can be encoded as part of the invariant by including $\mathbf{w}_\ell$, $\mathbf{w}_e$ in the variables of the hybrid automaton [113]. Invariants and initial states are given by linear constraints, as in PCD automata. The jump operator is

$$\mathsf{Post}_D(\ell, \alpha, \ell', \mathcal{P}) \quad = \big(R_{\ell,\alpha,\ell'}(\mathcal{P} \cap \mathcal{G}) \oplus S_{\ell,\alpha,\ell'} \mathcal{W}_{\ell,\alpha,\ell'}\big) \cap \mathsf{Inv}(\ell').$$

[7] Affine dynamics $\dot{\mathbf{x}} = A\mathbf{x} + \hat{B}\hat{\mathbf{w}} + \hat{\mathbf{c}}$, $\hat{\mathbf{w}} \in \hat{\mathcal{W}}$ can be brought to the form of (17) by taking $B = I, \mathcal{W} = \hat{B}\hat{\mathcal{W}} \oplus \hat{c}$.

The time elapse operator can be approximated with a sequence of sets as described in Sect. 3, but with the added difficulty that the invariant $\mathsf{Inv}(\ell)$ must be taken into account [114]. For now we assume a finite time horizon and later present the extension to infinite time in Sect. 5.6. Intersecting the invariant at each step of the reach tube approximation (12), we get the sequence $\mathcal{S}_k$ for $k = 0, \ldots, N-1$ as

$$\begin{aligned} \mathcal{S}_0 &= \operatorname{conv}\left(\mathcal{X}_0 \cup (e^{A\tau}\mathcal{X}_0 \oplus \tau\mathcal{W} \oplus \alpha(\tau, A, B, \mathcal{X}_0, \mathcal{W})\mathcal{B})\right) \cap \mathsf{Inv}(\ell), \\ \mathcal{S}_k &= \left(e^{A\tau}\mathcal{S}_{k-1} \oplus \tau\mathcal{W} \oplus \beta(\tau, A, B, \mathcal{W})\mathcal{B}\right) \cap \mathsf{Inv}(\ell). \end{aligned} \tag{18}$$

The time elapse operator is $\mathsf{Post}_C(\ell, P) = \bigcup_{k=0}^{N-1} \mathcal{S}_k$. Since the intersection in (18) is incompatible with several of the scalability improvements in Sect. 3, one often resorts to computing $\mathcal{S}_k$ using (12) and afterwards intersecting with the invariant. For set representations that are not closed under intersection, the time elapse operator under invariant constraints introduces an approximation error that can not be decreased by taking smaller time steps. Instead, the set of initial states can be split into smaller pieces, but this leads to state explosion.

State Explosion and Clustering: Thanks to the methods in Sect. 3, computing successor states with (12) scales fairly well. However, in the fixed point computation from Sect. 5.2, this approach can lead to severe state explosion. Recall that the input to Post_C is a single set, while its output is a possibly long sequence of convex sets. Post_D maps each one of these usually to another convex set, usually filtering out a number of them that do not overlap with the guard set. Repeating this process produces an exponentially increasing number of sets, unless Post_D happens to filter out all but one set in the sequence.

A common remedy for state explosion is to force Post_D to return a single set, e.g., by taking the convex hull or a bounding box of its image. This is referred to as *clustering*. Clustering can reduce the state explosion problem, but it introduces a wrapping effect that may quickly lead to an exploding approximation error. As a remedy, optimal clustering has been proposed in [115]. The approximated reach sets are clustered such that the overapproximation error can be measured. Finding the minimal number and exact time intervals over which to cluster sets are reduced to a graph coloring problem [115].

5.5 Nonlinear Dynamics

For nonlinear dynamics, the time elapse operator is computed using one of the methods described in Sect. 4. The central problems outlined for piecewise affine systems remain: intersecting with the invariant may induce overapproximation and clustering is required to fight state explosion, adding to the wrapping effect. Complex dynamics can be abstracted by simpler dynamics over a given region of the state space, as described in Sect. 4.2.1. The abstraction error depends on the diameter of the region and can be made arbitrarily small by dividing the state space into small enough pieces [83]. This technique is readily applied to hybrid systems, by representing each piece as a discrete state and connecting adjacent pieces by transitions. The technique is therefore referred to as *hybridization*.

5.6 Unbounded vs Bounded Time

There is a subtle but fundamental difference between the reachability computations in the sections 3 and 4 and this section on hybrid systems: The set propagation techniques for continuous systems

cover a finite time horizon, while our definition of the time elapse operator Post_C, as well as the definition of the reachable states for hybrid automata, range over an infinite time horizon. Even if the hybrid system only spends a finite time in each location, it may not be easy to estimate an upper bound that is not overly conservative. One solution is to extend the time elapse operator to infinite time through widening or abstract acceleration as in [116].

We briefly sketch out another solution, which exploits the fixed point computation of the reachability algorithm from Sect. 5.2 to lift the operation from bounded to infinite time. The system is augmented with a clock variable and transitions that reset the clock every $\delta > 0$ time units. The time elapse operator can be limited to a time horizon of δ in the augmented system, since the system can not remain longer in any location without taking a transition. The reachable states of the augmented system are identical to those of the original system if one projects away the clock. This augmentation can be applied to purely continuous systems by considering them as a hybrid automata with a single location. It follows that infinite-time reachability can be reduced to bounded-time reachability with an unbounded number of jumps.

6 APPLICATIONS

Reachability analysis has been applied across many application areas. A small selection of applications is provided in Tab. 2 covering aerospace, analg/mixed-signal circuits, automotive, power systems, robotics, and system biology. The table also identifies the main purposes, such as verification, stability analysis, planning, prediction, and conformance checking. We also list whether reachability is predominantly used in an offline setting (i.e., reachability analysis is performed before deployment) or in an online setting (i.e., reachability analysis is continuously performed during operation considering updated environment models). Each application area makes use of all purposes of reachability analysis even though the list is not exhaustive.

Nevertheless, one can see some trends in each application area. Systems developed or analyzed in analog/mixed-signal circuits and system biology are less autonomous in the sense that they do not explore an environment that is unknown during design time and thus rather use reachability analysis in an offline setting. A difference between analog/mixed-signal circuits and system biology is that reachability is mainly used for verification in analog/mixed-signal circuits since modeling is fairly well understood, whereas in system biology, reachability analysis is primarily used for finding conformant models since underlying mechanisms are not yet well understood.

The aerospace, automotive, and robotics sectors are similar in the sense that the safety-critical applications are those where decisions are made autonomously in constantly changing environments. Thus, these sectors require online methods , mainly for predicting the environment in a set-based fashion and for verifying whether these decisions are safe. The search space for these safe decisions can be efficiently pruned using reachability analysis.

The power sector is very diverse, since it is the world's most complex interconnected man-made system. While some decisions are made on a day to day basis, such as planning of energy production schedules, some control actions have to be made within milliseconds to ensure transient stability of parts of the grid. Thus, one finds a mix of online and offline techniques, as well as a mix of purposes and considered system types—even systems described by differential algebraic equations.

In particular, reachability analysis has been successfully applied to online verification; a selection is shown in Fig. 2. Illustration A in Fig. 2 shows a modular robot that can be reconfigured. After each reconfiguration, the robot reprograms its online verification based on reachability analysis itself [134]. It is the presumably the first robot that provably avoids collisions when humans interact with

Table 2: Examples of applications of reachability analysis across several application domains.

Paper	Main purpose	System type	Offline vs. online
		Aerospace	
[117]	planning	nonlinear	online (simulation)
[118]	prediction	nonlinear	online (simulation)
		Analog/Mixed-signal circuits	
[101]	verification	differential algebraic equations, hybrid	offline
[119]	verification	nonlinear, hybrid	offline
[120]	verification, stability	nonlinear, hybrid	offline
[121]	verification, stability	linear, hybrid	offline
[86]	verification	nonlinear	offline
		Automotive	
[122]	verification	linear, hybrid	offline
[123]	verification	nonlinear	online (simulation)
[124]	verification	nonlinear	online (simulation)
[125]	verification	nonlinear	online (real system)
[20]	prediction	nonlinear	online (simulation)
[126]	planning	linear	online (simulation)
		Power systems	
[127]	prediction	linear	offline
[128]	verification	nonlinear	offline
[129]	verification	differential algebraic equations	online (simulation)
[130]	conformance checking, verification	nonlinear	online (real system)
[131]	stability	nonlinear	offline
		Robotics	
[132]	planning	nonlinear	online (real system)
[133]	conformance checking, planning	linear	online (simulation)
[134]	verification	linear	online (real system)
		System biology	
[135]	conformance checking	nonlinear	offline
[136]	conformance checking	nonlinear	offline
[137]	verification	linear	offline
[138]	conformance checking	nonlinear	offline

it. Illustration B in Fig. 2 displays a scene during a test drive of a BMW vehicle that uses set-based motion planning and online verification to guarantee safety [20]. The last example shown in Fig. 2 is an office robot that can be certified using reachability analysis. It can be shown that the robot is provably safe, but also that the robot reaches its goal around 1.4-3.5 time faster than the navigation according to the ISO 13855 and 13482 standards [133].

7 CONCLUSIONS

After more than two decades of research, reachability analysis for continuous and hybrid systems has become an effective tool for model-based design of complex engineered systems. During that period, there has been tremendous improvement on scalability and accuracy of algorithms for approximating the reachable set of continuous and hybrid systems, making it possible to use these techniques on real-world problems. Several challenging research problems remain to be addressed in the field, such as handling large initial sets for nonlinear systems and many guard intersections in hybrid systems. Both aspects are especially relevant when verifying systems involving neural

Figure 2: Examples of applications in which our methods are already used. (A) Provably safe human-robot interaction. (B) Guaranteeing safety in autonomous driving. (C) Office robot whose safety is certifiable due to our approach; image provided by courtesy of Bosch Research.

networks. Also, new methods are required to be able to perform online verification (aka real-time verification) to verify autonomous systems in (partially) unknown environments.

ACKNOWLEDGMENTS

This work was partially supported by the Inria IPL ModeliScale large scale initiative (2017-2021) and by the European Commission project justITSELF under grant number 817629.

References

[1] Clarke Jr EM, Grumberg O, Kroening D, Peled D, Veith H. 2018. Model checking. MIT press

[2] Baier C, Katoen JP. 2008. Principles of model checking. MIT press

[3] Bertsekas DP, Rhodes IB. 1971. On the minimax reachability of target sets and target tubes. *Automatica* 7:233–247

[4] Alur R, Courcoubetis C, Halbwachs N, Henzinger TA, Ho PH, et al. 1995. The algorithmic analysis of hybrid systems. *Theoretical computer science* 138:3–34

[5] Silva BI, Richeson K, Krogh B, Chutinan A. 2000. Modeling and verifying hybrid dynamic systems using CheckMate. In *International Conference on Automation of Mixed Processes*

[6] Asarin E, Dang T, Maler O. 2002. The d/dt tool for verification of hybrid systems. In *Computer Aided Verification*

[7] Fijalkow N, Ouaknine J, Pouly A, Sousa-Pinto J, Worrell J. 2019. On the decidability of reachability in linear time-invariant systems. In *Hybrid Systems: Computation and Control*

[8] Henzinger TA, Kopke PW, Puri A, Varaiya P. 1998. What's decidable about hybrid automata? *Journal of Computer and System Sciences* 57:94–124

[9] Brihaye T, Doyen L, Geeraerts G, Ouaknine J, Raskin JF, Worrell J. 2011. On reachability for hybrid automata over bounded time. In *International Colloquium on Automata, Languages, and Programming*

[10] Rwth XC, Sankaranarayanan S, Abrahám E. 2014. Under-approximate flowpipes for non-linear continuous systems. In *Formal Methods in Computer-Aided Design*

[11] Blanchini F. 1999. Set invariance in control. *Automatica* 35:1747 – 1767

[12] Rungger M, Tabuada P. 2017. Computing robust controlled invariant sets of linear systems. *IEEE Transactions on Automatic Control* 62:3665–3670

[13] El-Guindy A, Han D, Althoff M. 2017. Estimating the region of attraction via forward reachable sets. In *Proc. of the American Control Conference*

[14] Bravo JM, Alamo T, Camacho EF. 2006. Robust MPC of constrained discrete-time nonlinear systems based on approximated reachable sets. *Automatica* 42:1745–1751

[15] Schürmann B, Kochdumper N, Althoff M. 2018. Reachset model predictive control for disturbed nonlinear systems. In *Proc. of the 57th IEEE Conference on Decision and Control*

[16] Gruber F, Althoff M. 2019. Scalable robust model predictive control for linear sampled-data systems. In *Proc. of the 58th IEEE Conference on Decision and Control*

[17] Combastel C. 2005. A state bounding observer for uncertain non-linear continuous-time systems based on zonotopes. In *Proc. of the 44th IEEE Conference on Decision and Control, and the European Control Conference*

[18] Scott JK, Raimondo DM, Marseglia GR, Braatz RD. 2016. Constrained zonotopes: A new tool for set-based estimation and fault detection. *Automatica* 69:126–136

[19] Su J, Chen W. 2019. Model-based fault diagnosis system verification using reachability analysis. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 49:742–751

[20] Althoff M, Magdici S. 2016. Set-based prediction of traffic participants on arbitrary road networks. *IEEE Transactions on Intelligent Vehicles* 1:187–202

[21] Pereira A, Althoff M. 2018. Overapproximative human arm occupancy prediction for collision avoidance. *IEEE Transactions on Automation Science and Engineering* 15:818–831

[22] Zamani M, Pola G, Mazo M, Tabuada P. 2012. Symbolic models for nonlinear control systems without stability assumptions. *IEEE Transactions on Automatic Control* 57:1804–1809

[23] Schürmann B, Althoff M. 2017. Guaranteeing constraints of disturbed nonlinear systems using set-based optimal control in generator space. In *Proc. of the 20th World Congress of the International Federation of Automatic Control*

[24] Roehm H, Oehlerking J, Woehrle M, Althoff M. 2016. Reachset conformance testing of hybrid automata. In *Proc. of Hybrid Systems: Computation and Control*

[25] Roehm H, Oehlerking J, Woehrle M, Althoff M. 2019. Model conformance for cyber-physical systems: A survey. *ACM Transactions on Cyber-Physical Systems* 3:1–26

[26] Mitchell IM, Bayen AM, Tomlin CJ. 2005. A time-dependent Hamilton–Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on Automatic Control* 50:947–957

[27] Chen M, Tomlin CJ. 2018. Hamilton–Jacobi reachability: Some recent theoretical advances and applications in unmanned airspace management. *Annual Review of Control, Robotics, and Autonomous Systems* 1:333–358

[28] Prajna S, Jadbabaie A. 2004. Safety verification of hybrid systems using barrier certificates. In *Hybrid Systems: Computation and Control*

[29] Kong H, Bartocci E, Henzinger TA. 2018. Reachable set over-approximationfor nonlinear systems using piecewisebarrier tubes. In *Proc. of Computer Aided Verification*

[30] Girard A, Pappas GJ. 2006. Verification using simulation. In *Hybrid Systems: Computation and Control*

[31] Donzé A, Maler O. 2007. Systematic simulation using sensitivity analysis. In *Hybrid Systems: Computation and Control*

[32] Ramdani N, Meslem N, Candau Y. 2008. Reachability analysis of uncertain nonlinear systems using guaranteed set integration. In *Proc. of the 17th IFAC World Congress*

[33] Duggirala PS, Mitra S, Viswanathan M. 2013. Verification of annotated models from executions. In *International Conference on Embedded Software*

[34] Fan C, Kapinski J, Jin X, Mitra S. 2016. Locally optimal reach set over-approximation for nonlinear systems. In *Proc. of the International Conference on Embedded Software*

[35] Ratschan S, She Z. 2007. Safety verification of hybrid systems by constraint propagation based abstraction refinement. *ACM Transactions in Embedded Computing Systems* 6:1–23

[36] Gao S, Kong S, Clarke E. 2013. Satisfiability modulo ODEs. In *Proc. of Formal Methods in Computer-Aided Design*

[37] Frehse G, Althoff M, eds. 2019. Arch19. 6th international workshop on applied verification of continuous and hybrid systems, vol. 61 of *EPiC Series in Computing*. EasyChair

[38] Althoff M, Krogh BH. 2011. Zonotope bundles for the efficient computation of reachable sets. In *Proc. of the 50th IEEE Conference on Decision and Control*

[39] Reißig G. 2007. Convexity of reachable setsof nonlinear ordinary differential equations. *Automation and Remote Control* 68:1527–1543

[40] Chen X, Sankaranarayanan S, Ábrahám E. 2012. Taylor model flowpipe construction for non-linear hybrid systems. In *Proc. of the 33rd IEEE Real-Time Systems Symposium*

[41] Althoff M. 2013. Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In *Hybrid Systems: Computation and Control*

[42] Kochdumper N, Althoff M. 2020. Constrained polynomial zonotopes. ArXiv:2005.08849

[43] Kochdumper N, Althoff M. 2019. Sparse polynomial zonotopes: A novel set representation for reachability analysis. ArXiv:1901.01780

[44] Neidinger RD. 2004. Directions for computing truncated multivariate Taylor series. *Mathematics of Computation* 74:321–340

[45] Makino K, Berz M. 2003. Taylor models and other validated functional inclusion methods. *International Journal of Pure and Applied Mathematics* 4:379–456

[46] Duggirala PS, Viswanathan M. 2016. Parsimonious, simulation based verification of linear systems. In *Proc. of International Conference on Computer Aided Verification*

[47] Jaulin L, Kieffer M, Didrit O. 2006. Applied interval analysis. Springer

[48] Kurzhanskiy AA, Varaiya P. 2006. Ellipsoidal toolbox. Tech. Rep. UCB/EECS-2006-46, EECS Department, University of California, Berkeley

[49] Bogomolov S, Forets M, Frehse G, Viry F, Podelski A, Schilling C. 2018. Reach set approximation through decomposition with low-dimensional sets and high-dimensional matrices. In *Proc. of the 21st International Conference on Hybrid Systems: Computation and Control*

[50] Avis D, Bremner D, Seidel R. 1997. How good are convex hull algorithms? *Computational Geometry: Theory and Applications* 7:265–301

[51] Fukuda K, Liebling TM, Lütolf C. 2001. Extended convex hull. *Computational Geometry* 20:13 – 23

[52] Bremner DD. 1997. On the complexity of vertex and facet enumeration for convex polytopes

[53] Althoff M, Frehse G. 2016. Combining zonotopes and support functions for efficient reachability analysis of linear systems. In *Proc. of the 55th IEEE Conference on Decision and Control*

[54] Tran HD, Cai F, Diego ML, Musau P, Johnson TT, Koutsoukos X. 2019. Safety verification of cyber-physical systems with reinforcement learning control. *ACM Transactions on Embedded Computing Systems* 18

[55] Bak S, Duggirala PS. 2017. Simulation-equivalent reachability of large linear systems with inputs. In *Proc. of International Conference on Computer Aided Verification*

[56] McMullen E. 1970. The maximum numbers of faces of a convex polytope. *Mathematika* 17:179–184

[57] Botchkarev O, Tripakis S. 2000. Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In *Hybrid Systems: Computation and Control*

[58] Kurzhanski AB, Varaiya P. 2000. Ellipsoidal techniques for reachability analysis. In *Hybrid Systems: Computation and Control*

[59] Chutinan A, Krogh BH. 2003. Computational techniques for hybrid system verification. *IEEE Transactions on Automatic Control* 48:64–75

[60] Asarin E, Bournez O, Dang T, Maler O. 2000. Approximate reachability analysis of piecewise-linear dynamical systems. In *Hybrid Systems: Computation and Control*, LNCS 1790. Springer

[61] Girard A. 2005. Reachability of uncertain linear systems using zonotopes. In *Hybrid Systems: Computation and Control*

[62] Althoff M, Stursberg O, Buss M. 2010. Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. *Nonlinear Analysis: Hybrid Systems* 4:233–249

[63] Le Guernic C, Girard A. 2010. Reachability analysis of linear systems using support functions. *Nonlinear Analysis: Hybrid Systems* 4:250–262

[64] Frehse G, Le Guernic C, Donzé A, Cotton S, Ray R, et al. 2011. SpaceEx: Scalable verification of hybrid systems. In *Computer Aided Verification*

[65] Althoff M, Le Guernic C, Krogh BH. 2011. Reachable set computation for uncertain time-varying linear systems. In *Hybrid Systems: Computation and Control*

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

[67] Girard A, Le Guernic C, Maler O. 2006. Efficient computation of reachable sets of linear time-invariant systems with inputs. In *Hybrid Systems: Computation and Control*

[68] Han Z, Krogh B. 2004. Reachability analysis of hybrid control systems using reduced-order models. In *Proc. of the American Control Conference*

[69] Han Z, Krogh BH. 2006. Reachability analysis of large-scale affine systems using low-dimensional polytopes. In *Hybrid Systems: Computation and Control*, LNCS 3927. Springer

[70] Tran HD, Nguyen LV, Xiang W, Johnson TT. 2017. Order-reduction abstractions for safety verification of high-dimensional linear systems. *Discrete Event Dynamic Systems* 27:443–461

[71] Bak S, Tran HD, Johnson TT. 2019. Numerical verification of affine systems with up to a billion dimensions. In *Proc. of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*

[72] Althoff M. 2020. Reachability analysis of large linear systems with uncertain inputs in the krylov subspace. *IEEE Transactions on Automatic Control* 65:477–492

[73] Asarin E, Dang T. 2004. Abstraction by projection and application to multi-affine systems. In *Hybrid Systems: Computation and Control*

[74] Kaynama S, Oishi M. 2011. Complexity reduction through a schur-based decomposition for reachability analysis of linear time-invariant systems. *International Journal of Control* 84:165–179

[75] Mitchell IM, Tomlin C. 2003. Overapproximating reachable sets by Hamilton-Jacobi projections. *Journal of Scientific Computing* 19:323–346

[76] Greenstreet MR, Mitchell I. 1999. Reachability analysis using polygonal projections. In *Hybrid Systems: Computation and Control*

[77] Chen X, Sankaranarayanan S. 2016. Decomposed reachability analysis for nonlinear systems. In *IEEE Real-Time Systems Symposium*

[78] Berz M, Hoffstätter G. 1998. Computation and application of Taylor polynomials with interval remainder bounds. *Reliable Computing* 4:83–97

[79] Makino K, Berz M. 1996. Remainder differential algebras and their applications. In *Computational Differentiation: Techniques, Applications, and Tools*. SIAM, 63–74

[80] Eckmann JP, Malaspinas A, Kamphorst SO. 1991. A software tool for analysis in function spaces. Springer, 147–167

[81] Sankaranarayanan S. 2011. Automatic abstraction of non-linear systems using change of bases transformations. In *Hybrid Systems: Computation and Control*

[82] Sankaranarayanan S. 2016. Change-of-bases abstractions for non-linear hybrid systems. *Nonlinear Analysis: Hybrid Systems* 19:107–133

[83] Henzinger TA, Ho PH, Wong-Toi H. 1998. Algorithmic analysis of nonlinear hybrid systems. *IEEE Transactions on Automatic Control* 43:540–554

[84] Asarin E, Dang T, Girard A. 2003. Reachability analysis of nonlinear systems using conservative approximation. In *Hybrid Systems: Computation and Control*, LNCS 2623. Springer

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

[86] Lee HSL, Althoff M, Hoelldampf S, Olbrich M, Barke E. 2015. Automated generation of hybrid system models for reachability analysis of nonlinear analog circuits. In *Proc. of the 20th Asia and South Pacific Design Automation Conference*

[87] Althoff M, Stursberg O, Buss M. 2008. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In *Proc. of the 47th IEEE Conference on Decision and Control*

[88] Dang T, Maler O, Testylier R. 2010. Accurate hybridization of nonlinear systems. In *Hybrid Systems: Computation and Control*

[89] Waldron S. 1998. The error in linear interpolation at the vertices of a simplex. *SIAM Journal on Numerical Analysis* 35:1191–1200

[90] Han Z, Krogh BH. 2006. Reachability analysis of nonlinear systems using trajectory piecewise linearized models. In *Proc. of the American Control Conference*

[91] Dang T. 2006. Approximate reachability computation for polynomial systems. In *Hybrid Systems: Computation and Control*

[92] Dang T, Maler O. 1998. Reachability analysis via face lifting. In *Hybrid Systems: Computation and Control*

[93] Dang T, Salinas D. 2009. Image computation for polynomial dynamical systems using the Bernstein expansion. In *Computer Aided Verification*, LNCS 5643. Springer

[94] Angeli D, Sontag ED. 2003. Monotone control systems. *IEEE Transactions on Automatic Control* 48:1684–1698

[95] Ramdani N, Meslem N, Candau Y. 2010. Computing reachable sets for uncertain nonlinear monotone systems. *Nonlinear Analysis: Hybrid Systems* 4:263–278

[96] Ramdani N, Meslem N, Candau Y. 2008. Reachability of uncertain nonlinear systems using a nonlinear hybridization. In *Hybrid Systems: Computation and Control*, LNCS 4981. Springer

[97] Coogan S, Arcak M. 2015. Efficient finite abstraction of mixed monotone systems. In *Proc. of the 18th International Conference on Hybrid Systems: Computation and Control*

[98] Scott JK, Barton PI. 2013. Bounds on the reachable sets of nonlinear control systems. *Automatica* 49:93–100

[99] Cross EA, Mitchell IM. 2008. Level set methods for computing reachable sets of systems with differential algebraic equation dynamics. In *Proc of the American Control Conference*

[100] Mitchell IM, Susuki Y. 2008. Level set methods for computing reachable sets of hybrid systems with differential algebraic equation dynamics. In *Hybrid Systems: Computation and Control*, LNCS 4981. Springer

[101] Dang T, Donze A, Maler O. 2004. Verification of analog and mixed-signal circuits using hybrid systems techniques. In *Formal Methods for Computer Aided Design*, LNCS 3312. Springer

[102] Althoff M, Krogh BH. 2014. Reachability analysis of nonlinear differential-algebraic systems. *IEEE Transactions on Automatic Control* 59:371–383

[103] Alur R, Courcoubetis C, Henzinger TA, Ho PH. 1993. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, vol. 736 of *LNCS*. Springer, 209–229

[104] Bemporad A, Morari M. 1999. Control of systems integrating logic, dynamics, and constraints. *Automatica* 35:407–427

[105] Goebel R, Sanfelice RG, Teel AR. 2009. Hybrid dynamical systems. *IEEE control systems magazine* 29:28–93

[106] Henzinger T. 1996. The theory of hybrid automata. In *IEEE Annual Symposium on Logic in Computer Science*

[107] Minopoli S, Frehse G. 2016 (to appear). From simulation models to hybrid automata using urgency and relaxation. In *HSCC'16*

[108] Halbwachs N, Proy YE, Raymond P. 1994. Verification of linear hybrid systems by means of convex approximations. In *International Static Analysis Symposium*

[109] Becchi A, Zaffanella E. 2019. Revisiting polyhedral analysis for hybrid systems. In *Static Analysis*

[110] Fränzle M. 1999. Analysis of hybrid systems: An ounce of realism can save an infinity of states. In *International Workshop on Computer Science Logic*

[111] Cousot P, Cousot R. 1977. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Symposium on Principles of Programming Languages*

[112] Frehse G. 2008. PHAVer: algorithmic verification of hybrid systems past HyTech. *International Journal on Software Tools for Technology Transfer* 10:263–279

[113] Donzé A, Frehse G. 2013. Modular, hierarchical models of control systems in spaceex. In *European Control Conference*

[114] Le Guernic C, Girard A. 2009. Reachability analysis of hybrid systems using support functions. In *Computer Aided Verification*

[115] Frehse G, Kateja R, Le Guernic C. 2013. Flowpipe approximation and clustering in space-time. In *Hybrid systems: computation and control*

[116] Cattaruzza D, Abate A, Schrammel P, Kroening D. 2015. Unbounded-time analysis of guarded lti systems with inputs by abstract acceleration. In *International Static Analysis Symposium*

[117] Althoff D, Althoff M, Scherer S. 2015. Online safety verification of trajectories for unmanned flight with offline computed robust invariant sets. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*

[118] Hobbs K, Heidlauf P, Collins A, Bak S. 2018. Space debris collision detection using reachability. In *Proc. of the 5th International Workshop on Applied Verification of Continuous and Hybrid Systems*

[119] Gupta S, Krogh BH, Rutenbar RA. 2004. Towards formal verification of analog designs. In *International Conference on Computer-Aided Design*. IEEE Computer Society / ACM

[120] Frehse G, Krogh BH, Rutenbar RA. 2006. Verifying analog oscillator circuits using forward/backward abstraction refinement. In *Proc. of the Conference on Design, Automation and Test in Europe*

[121] Althoff M, Rajhans A, Krogh BH, Yaldiz S, Li X, Pileggi L. 2013. Formal verification of phase-locked loops using reachability analysis and continuization. *Communications of the ACM* 56:97–104

[122] Stursberg O, Fehnker A, Han Z, Krogh BH. 2004. Verification of a cruise control system using counterexample-guided search. *Control Engineering Practice* 12:1269–1278

[123] Althoff M, Althoff D, Wollherr D, Buss M. 2010. Safety verification of autonomous vehicles for coordinated evasive maneuvers. In *Proc. of the IEEE Intelligent Vehicles Symposium*

[124] Heß D, Althoff M, Sattel T. 2014. Formal verification of maneuver automata for parameterized motion primitives. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*

[125] Althoff M, Dolan JM. 2014. Online verification of automated road vehicles using reachability analysis. *IEEE Transactions on Robotics* 30:903–918

[126] Söntges S, Althoff M. 2018. Computing the drivable area of autonomous road vehicles in dynamic road scenes. *IEEE Transactions on Intelligent Transportation Systems* 19:1855–1866

[127] Chen YC, Domínguez-García AD. 2012. A method to study the effect of renewable resource variability on power system dynamics. *IEEE Transactions on Power Systems* 27:1978–1989

[128] Villegas Pico HN, Aliprantis DC. 2014. Voltage ride-through capability verification of wind turbines with fully-rated converters using reachability analysis. *IEEE Transactions on Energy Conversion* 29:392–405

[129] Althoff M. 2014. Formal and compositional analysis of power systems using reachable sets. *IEEE Transactions on Power Systems* 29:2270–2280

[130] El-Guindy A, Han D, Althoff M. 2016. Formal analysis of drum-boiler units to maximize the load-following capabilities of power plants. *IEEE Transactions on Power Systems* 31:4691–4702

[131] Li Y, Zhang P, Luh PB. 2018. Formal analysis of networked microgrids dynamics. *IEEE Transactions on Power Systems* 33:3418–3427

[132] Lengagne S, Ramdani N, Fraisse P. 2011. Planning and fast replanning safe motions for humanoid robots. *IEEE Transactions on Robotics* 27:1095–1106

[133] Liu SB, Roehm H, Heinzemann C, Lütkebohle I, Oehlerking J, Althoff M. 2017. Provably safe motion of mobile robots in human environments. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*

[134] Althoff M, Giusti A, Liu SB, Pereira A. 2019. Effortless creation of safe robots from modules through self-programming and self-verification. *Science Robotics* 4:1–14

[135] Batt G, Belta C, Weiss R. 2007. Model checking genetic regulatory networkswith parameter uncertainty. In *Hybrid Systems: Computation and Control*

[136] Dang T, Guernic CL, Maler O. 2009. Computing reachable states for nonlinear biological models. In *Computational Methods in Systems Biology*, LNCS 5688. Springer

[137] Kaynama S, Maidens JN, Oishi M, Mitchell IM, Dumont GA. 2012. Computing the viability kernel using maximal reachable sets. In *Proc. of Hybrid Systems: Computation and Control*

[138] Dang T, Dreossi T, Fanchon E, Maler O, Piazza C, Rocca A. 2019. Automated reasoning for systems biology and medicine, chap. Set-Based Analysis for Biological Modeling. Springer, 157–189