# Utilizing Dependencies to Obtain Subsets of Reachable Sets

Niklas Kochdumper
niklas.kochdumper@tum.de
Technische Universität München

Bastian Schürmann
bastian.schuermann@tum.de
Technische Universität München

Matthias Althoff
althoff@tum.de
Technische Universität München

## ABSTRACT

Reachability analysis, in general, is a fundamental method that supports formally-correct synthesis, robust model predictive control, set-based observers, fault detection, invariant computation, and conformance checking, to name but a few. In many of these applications, one requires to compute a reachable set starting within a previously computed reachable set. While it was previously required to re-compute the entire reachable set, we demonstrate that one can leverage the dependencies of states within the previously computed set. As a result, we almost instantly obtain an over-approximative subset of a previously computed reachable set by evaluating analytical maps. The advantages of our novel method are demonstrated for falsification of systems, optimization over reachable sets, and synthesizing safe maneuver automata. In all of these applications, the computation time is reduced significantly.

## CCS CONCEPTS

• **General and reference** → **Verification**; • **Mathematics of computing** → **Ordinary differential equations**

.

## KEYWORDS

dependency preservation, reachability analysis, nonlinear dynamics, polynomial zonotopes.

**ACM Reference Format:**
Niklas Kochdumper, Bastian Schürmann, and Matthias Althoff. 2020. Utilizing Dependencies to Obtain Subsets of Reachable Sets. In *23rd ACM International Conference on Hybrid Systems: Computation and Control (HSCC '20), April 22–24, 2020, Sydney, NSW, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3365365.3382192

## 1 INTRODUCTION

In this paper, we present a novel method to directly extract a reachable set within a pre-computed one as depicted in Fig. 1. To achieve this, we conservatively abstract the original dynamics by a polynomial right-hand side and represent sets by polynomial zonotopes [1]. Since polynomial zonotopes preserve the relation between the reachable states and the states in the initial set, we can extract the reachable set $\widehat{\mathcal{R}}$ for any initial set $\widehat{\mathcal{X}}_0 \subseteq \mathcal{X}_0$ directly from the reachable set $\mathcal{R}$ by evaluating an analytical equation (see Fig. 1),

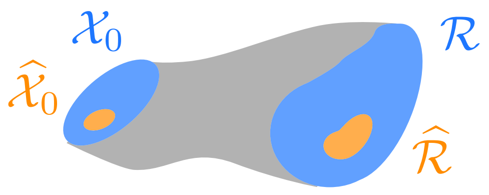

**Figure 1: Given a reachable set $\mathcal{R}$ for a set of initial states $\mathcal{X}_0$ and a subset of initial states $\widehat{\mathcal{X}}_0$, we can obtain $\widehat{\mathcal{R}}$ without any reachability analysis.**

which is computationally more efficient than computing $\widehat{\mathcal{R}}$ with a reachability algorithm. As we demonstrate in Sec. 4, this method offers great advantages for applications where reachable sets have to be computed for many different subsets $\widehat{\mathcal{X}}_0 \subseteq \mathcal{X}_0$, like *safety falsification*, *optimization over reachable sets*, and *motion-primitive based control*.

### 1.1 State of the Art

Reachability algorithms for linear systems and hybrid systems with linear continuous dynamics are mostly based on the propagation of reachable sets. These algorithms use a large variety of convex set representations, like polytopes [18], zonotopes [19], ellipsoids [28], support functions [20], and star sets [17]. Further, other approaches use simulations to compute reachable set [7, 17]. The examples of tools for reachability analysis of linear systems are C2E2 [16], CORA [2], HyDRA [37], Hylaa [6], Julia Reach [10], SpaceEx [18], and XSpeed [35].

The reachability algorithms for nonlinear systems can be categorized into four groups: invariant generation, optimization-based approaches, abstraction in solution space, and abstraction in state space. Since any invariant set which includes the initial set is also a reachable set, approaches for invariant generation can be used for reachability analysis [27, 30, 32]. The optimization-based approaches reformulate reachability analysis as an optimization problem [14, 33]. Thus, the approach in [14] optimizes the outward translation of polytope halfspaces to obtain a flowpipe, whereas [33] expresses reachability analysis with Hamilton-Jacobi equations. Other approaches abstract the solution space directly: The work in [17] uses validated simulations for the construction of bounded flowpipes, and [34] approximates the solution of the ODE with Bernstein polynomials; Taylor models computed from iterations such as the Picard iteration were initially proposed in [23, 31], and later extended to include uncertain inputs [11]. Approaches based on an abstraction of the state space compute simplified differential

*HSCC '20, April 22–24, 2020, Sydney, NSW, Australia*
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-7018-9/20/04…$15.00
https://doi.org/10.1145/3365365.3382192

equations to which a compensating uncertainty is added. Often, nonlinear ODEs are abstracted by a hybrid automaton with linear dynamics [5]. Other methods linearize the nonlinear dynamics on-the-fly [3, 15]. Recent approaches extend this concept to the abstraction of the nonlinear dynamics by polynomials [1], which results in a tighter enclosure of the reachable set. Examples of tools for reachability analysis of nonlinear systems are Adriadne [8], C2E2 [16], CORA [2], DynIbex [36], Flow* [12], Isabelle/HOL [24], and Julia Reach [10].

We show in this work how the relation between the initial and reachable states can be preserved during reachability analysis. For *abstract interpretation*, the work in [21] demonstrates how the parameterization of zonotopes can be utilized to preserve relations between inputs and outputs of computer programs. In [13], the relation between the initial and reachable states is used to compute inner-approximations of reachable sets. Similarly, the approach in [11] utilizes this relation to tighly enclose the intersections with guard sets for hybrid system reachability analysis.

Our new method is not only applicable to reachability analysis but also falsification. If a computed reachable set violates a specification, falsification aims to provide the initial states and the input signals that lead to the violation. The problem of finding such initial states and input signals is known as *safety falsification*. For linear systems with piecewise constant inputs, [7] extracts falsifying initial states and input signals from the computed reachable set by solving a linear program. A similar concept is used in [9], where reachability analysis is utilized to efficiently determine falsifying trajectories for hybrid systems with linear continuous dynamics. The set of initial states resulting in a falsification can also be computed with backward reachability analysis using inner-approximations of reachable sets. Approaches to compute inner-approximations of reachable sets exist for linear systems [19] as well as for nonlinear systems [13, 22, 29].

In this paper, we preserve the relation between the initial and the reachable states for nonlinear systems. We use polynomial zonotopes instead of Taylor models since the former is a generalization of Taylor models [26, Prop. 4]. Our novel method allows us to obtain subsets within a pre-computed reachable set almost instantly. Using numerical examples, we demonstrate that our method reduces the computation time for *safety falsification*, *optimization over reachable sets*, and *motion-primitive based control* significantly. To some extent this work proposes a novel method for unifying reachability analysis and falsification.

## 1.2 Notation

Sets are denoted by calligraphic letters, matrices by uppercase letters, vectors by lowercase letters, and set operations by typewriter font (e.g., `interval`). Given a vector $b \in \mathbb{R}^n$, $b_{(i)}$ refers to the $i$-th entry. Given a matrix $A \in \mathbb{R}^{n \times m}$, $A_{(i,\cdot)}$ represents the $i$-th matrix row, $A_{(\cdot,j)}$ the $j$-th column, and $A_{(i,j)}$ the $j$-th entry of matrix row $i$. The concatenation of two matrices $C$ and $D$ is denoted by $[C\ D]$. The symbols $\mathbf{0}$ and $\mathbf{1}$ represent matrices of zeros and ones of proper dimension and the empty matrix is denoted by $[\ ]$. Left multiplication of a matrix $M \in \mathbb{R}^{m \times n}$ with a set $\mathcal{S} \subset \mathbb{R}^n$ is defined as $M \otimes \mathcal{S} = \{Ms \mid s \in \mathcal{S}\}$, the Minkowski addition of two sets $\mathcal{S}_1 \subset \mathbb{R}^n$ and $\mathcal{S}_2 \subset \mathbb{R}^n$ is defined as $\mathcal{S}_1 \oplus \mathcal{S}_2 = \{s_1 + s_2 \mid s_1 \in \mathcal{S}_1, s_2 \in \mathcal{S}_2\}$, and the Cartesian product of two sets $\mathcal{S}_1 \subset \mathbb{R}^n$ and $\mathcal{S}_2 \subset \mathbb{R}^m$ is defined as $\mathcal{S}_1 \times \mathcal{S}_2 = \{[s_1\ s_2]^T \mid s_1 \in \mathcal{S}_1, s_2 \in \mathcal{S}_2\}$. Given two set operations $\mathtt{A}$ and $\mathtt{B}$, and a set $\mathcal{S} \subset \mathbb{R}^n$, the composition of the set operations is denoted by $\mathtt{A}(\mathtt{B}(\mathcal{S})) = (\mathtt{A} \circ \mathtt{B})(\mathcal{S})$. The power set of a set $\mathcal{S} \subset \mathbb{R}^n$ is denoted by $2^{\mathcal{S}}$. We further introduce a n-dimensional interval as $\mathcal{I} := [l, u]$, $\forall i\ l_{(i)} \leq u_{(i)}$, $l, u \in \mathbb{R}^n$. The unit hypercube $[-\mathbf{1}, \mathbf{1}] \subset \mathbb{R}^p$ is denoted by $\mathcal{I}_p$. Given a center vector $c \in \mathbb{R}^n$ and a generator matrix $G \in \mathbb{R}^{n \times m}$, a zonotope is $\mathcal{Z} := \{c + \sum_{i=1}^m \alpha_i\, G_{(\cdot,i)} \mid \alpha_i \in [-1, 1]\}$. For a concise notation, we use the shorthand $\mathcal{Z} = \langle c, G \rangle_Z$.

## 2 SET REPRESENTATION

### 2.1 Parameterization

A prerequisite for preserving relations between states is that the states inside the initial set are parameterized. As shown in Table 1, not all set representations fulfill this requirement. We demonstrate this exemplary for the vertex (V-representation) and the halfspace-representation (H-representation) of a polytope; next are the definitions.

Definition 1. *(V-Representation) Given m vertices $v_i \in \mathbb{R}^n$, the vertex representation of $\mathcal{P} \subset \mathbb{R}^n$ is defined as*

$$\mathcal{P} := \left\{ \sum_{i=1}^m \delta_i v_i \;\middle|\; \delta_i \geq 0,\ \sum_{i=1}^m \delta_i = 1 \right\}.$$

We use the shorthand $\mathcal{P} = \langle [v_1\ \dots\ v_m] \rangle_V$.

Definition 2. *(H-Representation) Given a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $b \in \mathbb{R}^m$, the halfspace representation of $\mathcal{P} \subset \mathbb{R}^n$ is defined as*

$$\mathcal{P} := \{x \mid Ax \leq b\}.$$

Each point $p \in \mathcal{P}$ can be parameterized by specific values $\overline{\delta}_i$ when using the V-representation (not possible for the H-representation):

$$p = \sum_{i=1}^m \overline{\delta}_i v_i,\ \overline{\delta}_i \geq 0,\ \sum_{i=1}^m \overline{\delta}_i = 1. \tag{1}$$

In general, the above parameterization is not unique [38]. For parameterized sets, we introduce evaluation functions.

Definition 3. *(Evaluation Function) Given a set $\mathcal{S} \subset \mathbb{R}^n$ that is parameterized by the parameter vector $d \in \mathcal{D} \subset \mathbb{R}^m$, the evaluation function $\lfloor \mathcal{S} \rfloor : \mathcal{D} \to 2^{\mathcal{S}}$ returns the set $\overline{\mathcal{S}}$ that corresponds to a specific value $\overline{d} \in \mathcal{D}$ of the parameter vector d:*

$$\lfloor \mathcal{S} \rfloor (\overline{d}) = \overline{\mathcal{S}},$$

*where the parameter domain $\mathcal{D} \subset \mathbb{R}^m$ satisfies*

$$\bigcup_{d \in \mathcal{D}} \lfloor \mathcal{S} \rfloor (d) = \mathcal{S}.$$

Example 1. *For a polytope $\mathcal{P} = \langle [v_1\ \dots\ v_m] \rangle_V$ in V-representation, the parameter domain is*

$$\mathcal{D} = \left\{ [\delta_1\ \dots\ \delta_m]^T \;\middle|\; \delta_i \geq 0,\ \sum_{i=1}^m \delta_i = 1 \right\}$$

*and the evaluation function is*

$$\lfloor \mathcal{P} \rfloor (\delta) = \left\{ \sum_{i=1}^m \delta_i v_i \right\}. \tag{2}$$

**Table 1: Characterization of set representations w.r.t. parameterization and dependency preservation of set operations.**

| Set Representation | Parameter- | Dependency Preservation | | | |
|---|---|---|---|---|---|
| | ization | Linear Trans. | Minkowski Sum | Cart. Product | Quad. Map |
| Interval | × | | | | |
| Zonotopes | √ | √ | √ | √ | √ |
| Polytopes (H-Rep.) | × | | | | |
| Polytopes (V-Rep.) | √ | √ | × | × | × |
| Ellipsoids | √ | √ | × | × | × |
| Support Functions | × | | | | |
| Level Sets | × | | | | |
| Star Sets | √ | √ | √ | √ | × |
| Taylor Models | √ | √ | √ | √ | √ |
| Polynomial Zonotopes | √ | √ | √ | √ | √ |

## 2.2 Dependency Preservation

We require a set representation that preserves the relation between states for all relevant set operations. First, we demonstrate dependency preservation for linear maps of V-representations:

Example 2. *Given a scalar $M \in \mathbb{R}$ and a one-dimensional polytope $\mathcal{P} = \langle [v_1, v_2] \rangle_V$, its linear map is computed as*

$$M \otimes \mathcal{P} = \langle [Mv_1 \; Mv_2] \rangle_V. \tag{3}$$

*Let us introduce the polytope $\mathcal{P} = \langle [-1 \; 3] \rangle_V$, the point $p = 2 \in \mathcal{P}$, and the scalar $M = 2$. According to* (1)*, the point $p \in \mathcal{P}$ can be parameterized by the values $\overline{\delta} = [\overline{\delta_1} \; \overline{\delta_2}]^T = [0.25 \; 0.75]^T$. The computation of the linear transformation according to* (3) *yields*

$$M \otimes \mathcal{P} = \langle [\widehat{v}_1 \; \widehat{v}_2] \rangle_V = \langle [-2 \; 6] \rangle_V.$$

*If we evaluate the result for $\overline{\delta}$ corresponding to the point $p$, we obtain*

$$\lfloor M \otimes \mathcal{P} \rfloor (\overline{\delta}) \overset{(2)}{=} \sum_{i=1}^{2} \overline{\delta}_i \widehat{v}_i = 4 = Mp.$$

*The implementation of the linear map in* (3) *is therefore dependency-preserving.*

Next, we consider a quadratic map, which is not dependency-preserving for the V-representation:

Example 3. *Given a scalar $Q \in \mathbb{R}$ and a one-dimensional polytope $\mathcal{P} = \langle [v_1 \; v_2] \rangle_V$, its quadratic map is computed as*

$$\begin{aligned} \mathsf{sq}(Q, \mathcal{P}) &= \left\{ x^T Q x \;\middle|\; x \in \mathcal{P} \right\} \\ &= \begin{cases} \left[ 0 \;\; \max(|v_1|, |v_2|)^2 Q \right], & 0 \in \mathcal{P} \\ \left[ \min(v_1^2, v_2^2) Q \;\; \max(v_1^2, v_2^2) Q \right], & \text{otherwise} \end{cases}. \end{aligned} \tag{4}$$

*Let us introduce the polytope $\mathcal{P} = \langle [-1 \; 3] \rangle_V$, the point $p = 2 \in \mathcal{P}$, and the scalar $Q = 2$. According to* (1)*, the point $p \in \mathcal{P}$ can be parameterized by the values $\overline{\delta} = [\overline{\delta_1} \; \overline{\delta_2}]^T = [0.25 \; 0.75]^T$. Computation of the quadratic map according to* (4) *yields*

$$sq(Q, \mathcal{P}) = \langle [\widehat{v}_1 \; \widehat{v}_2] \rangle_V = \langle [0 \; 18] \rangle_V.$$

*If we evaluate the computed quadratic map for $\overline{\delta}$ corresponding to the point $p$, we obtain*

$$\lfloor \mathsf{sq}(Q, \mathcal{P}) \rfloor (\overline{\delta}) \overset{(2)}{=} \sum_{i=1}^{2} \overline{\delta}_i \widehat{v}_i = 13.5 \neq \mathsf{sq}(Q, p) = 8,$$

*which is not identical to* $\mathsf{sq}(Q, p)$. *The above implementation is therefore not dependency-preserving.*

Let us finally define dependency preservation:

Definition 4. *(Dependency Preservation) Given an implementation of a set operation* $\mathsf{A}$ *and a set $\mathcal{S} \subset \mathbb{R}^n$ parameterized by $d \in \mathcal{D} \subset \mathbb{R}^m$, we call the implementation of* $\mathsf{A}$ *dependency-preserving if*

$$\forall d \in \mathcal{D} : \; \mathsf{A}\left( \lfloor \mathcal{S} \rfloor (d) \right) \subseteq \lfloor \mathsf{A}(\mathcal{S}) \rfloor (d). \tag{5}$$

Table 1 shows a summary of the set operations that are dependency-preserving for different set representations.

## 2.3 Set Operation Properties

We introduce some additional properties for set operations which we require for later derivations. Given two sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$ with $\mathcal{S}_1 \subseteq \mathcal{S}_2$, all unary set operations $\mathsf{A}$ used in this work satisfy

$$\mathsf{A}(\mathcal{S}_1) \subseteq \mathsf{A}(\mathcal{S}_2). \tag{6}$$

Furthermore, given sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$ and $\mathcal{S}_3, \mathcal{S}_4 \subset \mathbb{R}^m$ with $\mathcal{S}_1 \subseteq \mathcal{S}_2$ and $\mathcal{S}_3 \subseteq \mathcal{S}_4$, all binary set operations $\mathsf{B}$ used in this work satisfy

$$\mathsf{B}(\mathcal{S}_1, \mathcal{S}_3) \subseteq \mathsf{B}(\mathcal{S}_2, \mathcal{S}_4). \tag{7}$$

The properties (6) and (7) equivalently hold for the composition of set operations:

Lemma 1. *Given two set operations* $\mathsf{A}$ *and* $\mathsf{B}$ *that satisfy* (6)*, the composition* $\mathsf{A} \circ \mathsf{B}$ *also satisfies* (6)*.*

Proof. Since $\mathsf{A}$ and $\mathsf{B}$ satisfy (6), it holds for two sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$ with $\mathcal{S}_1 \subseteq \mathcal{S}_2$ that

$$(\mathsf{A} \circ \mathsf{B})(\mathcal{S}_1) = \mathsf{A}(\underbrace{\mathsf{B}(\mathcal{S}_1)}_{\overset{(6)}{\subseteq} \mathsf{B}(\mathcal{S}_2)}) \overset{(6)}{\subseteq} \mathsf{A}(\mathsf{B}(\mathcal{S}_2)) = (\mathsf{A} \circ \mathsf{B})(\mathcal{S}_2).$$

□

The result of Lemma 1 equally holds for compositions involving binary set operations that satisfy (7). Next, we show that the composition of two dependency-preserving set operations is dependency-preserving:

Lemma 2. *The composition* A ∘ B *of two dependency-preserving operations* A *and* B *is dependency-preserving as well.*

$$\forall d \in \mathcal{D}: \; (\mathsf{A} \circ \mathsf{B}) \left( \underline{\lfloor \mathcal{S} \rfloor}(d) \right) \subseteq \underline{\lfloor (\mathsf{A} \circ \mathsf{B})(\mathcal{S}) \rfloor}(d). \tag{8}$$

Proof. Since B is dependency-preserving, it holds according to (5) that

$$\forall d \in \mathcal{D}: \; \mathsf{B} \left( \underline{\lfloor \mathcal{S} \rfloor}(d) \right) \subseteq \underline{\lfloor \mathsf{B}(\mathcal{S}) \rfloor}(d). \tag{9}$$

Then, using (9) and (6), it holds that

$$\forall d \in \mathcal{D}: \; \mathsf{A} \left( \mathsf{B} \left( \underline{\lfloor \mathcal{S} \rfloor}(d) \right) \right) \subseteq \mathsf{A} \left( \underline{\lfloor \mathsf{B}(\mathcal{S}) \rfloor}(d) \right). \tag{10}$$

Since A is dependency-preserving, it holds according to (5) that

$$\begin{aligned} \forall d \in \mathcal{D}: (\mathsf{A} \circ \mathsf{B}) \left( \underline{\lfloor \mathcal{S} \rfloor}(d) \right) = \mathsf{A} \left( \mathsf{B} \left( \underline{\lfloor \mathcal{S} \rfloor}(d) \right) \right) &\overset{(10)}{\subseteq} \mathsf{A} \left( \underline{\lfloor \mathsf{B}(\mathcal{S}) \rfloor}(d) \right) \\ &\overset{(5)}{\subseteq} \underline{\lfloor \mathsf{A}(\mathsf{B}(\mathcal{S})) \rfloor}(d) = \underline{\lfloor (\mathsf{A} \circ \mathsf{B})(\mathcal{S}) \rfloor}(d), \end{aligned}$$

which is identical to (8). □

Since the reachability algorithm used in this work applies dependency-preserving set operations only, is follows that the algorithm is dependency-preserving too, as discussed in Sec. 3.

## 2.4 Polynomial Zonotopes

The concept presented in this work requires a parameterized set representation for which all operations used by the reachability algorithm are dependency-preserving. As shown in Table 1, a non-convex set representation that satisfies these requirements are polynomial zonotopes. In this work we use their sparse representation [26]:

Definition 5. *(Polynomial Zonotope) Given a generator matrix of dependent generators $G \in \mathbb{R}^{n \times h}$, a generator matrix of independent generators $G_I \in \mathbb{R}^{n \times q}$, and an exponent matrix $E \in \mathbb{Z}_{\geq 0}^{m \times h}$, a polynomial zonotope is defined as*

$$\mathcal{PZ} := \left\{ \underbrace{\sum_{i=1}^{h} \left( \prod_{k=1}^{m} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)}}_{f_{G,E}(\alpha)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \alpha_k, \beta_j \in [-1,1] \right\}, \tag{11}$$

*where $f_{G,E}: \mathbb{R}^m \to \mathbb{R}^n$ is a multivariate polynomial function, and $\alpha = [\alpha_1 \; \dots \; \alpha_m]^T \in \mathbb{R}^m$.*

The scalars $\alpha_k$ are called dependent factors and the scalars $\beta_j$ independent factors. Consequently, the term $f_{G,E}(\alpha)$ is called the dependent part, and the term $\sum_{j=1}^{q} \beta_j G_{I(\cdot,j)}$ is called the independent part. We introduce the shorthand $\mathcal{PZ} = \langle G, G_I, E \rangle_{PZ}$.

Every polynomial zonotope can equivalently be represented by a polynomial zonotope without independent generators:

Proposition 1. *Given a polynomial zonotope $\mathcal{PZ} = \langle G, G_I, E \rangle_{PZ}$, $\mathcal{PZ}$ can be equivalently represented without independent generators as follows:*

$$\mathcal{PZ} = \langle G, G_I, E \rangle_{PZ} = \left\langle [G \; G_I], [\,], \begin{bmatrix} E & \mathbf{0} \\ \mathbf{0} & I_q \end{bmatrix} \right\rangle_{PZ}, \tag{12}$$

*where $I_q \in \mathbb{R}^{q \times q}$ is the identity matrix.*

Proof. The result in (12) follows directly from the substitution of the independent factors $\beta_j$ in the definition of polynomial zonotopes in (11) with additional dependent factors $\alpha_{m+1} = \beta_1, \dots, \alpha_{p+q} = \beta_q$. □

Despite the result of Prop. 1, the independent part of the polynomial zonotope is crucial for order reduction [26]. For polynomial zonotopes, the points inside the set are parameterized by both the dependent factors $\alpha_k$ and the independent factors $\beta_j$. Using Prop. 1, we assume without loss of generality that the initial set has no independent generators. Consequently, it is sufficient to choose the parameter vector as $d = \alpha = [\alpha_1 \; \dots \; \alpha_m]^T$, the parameter domain as $\mathcal{D} = \mathcal{I}_m$, and the evaluation function as

$$\begin{aligned} \underline{\lfloor \mathcal{PZ} \rfloor}(\alpha) &= f_{G,E}(\alpha) \oplus \left\{ \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \beta_j \in [-1,1] \right\} \\ &= \langle f_{G,E}(\alpha), G_I \rangle_Z . \end{aligned} \tag{13}$$

However, finding a parameterization for a point inside a polynomial zonotope can be computational expensive. For reachability analysis, the initial set is often an axis-aligned box, for which the parameterization is unique and trivial to compute as shown by the following example:

Example 4. *We consider the polynomial zonotope*

$$\mathcal{PZ} = \left\{ \underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix} \alpha_1 + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \alpha_2}_{f_{G,E}(\alpha)} \;\middle|\; \alpha_1, \alpha_2 \in [-1,1] \right\}$$

*and the point $p = [0.5 \; 0.4]^T$. Trivially, the point $p$ can be parameterized with $\overline{\alpha} = [0.5 \; 0.4]^T$, so that $p = f_{G,E}(\overline{\alpha})$.*

Next, we integrate parameterized and dependency-preserving set representations in reachability analysis to obtain reachable subsets.

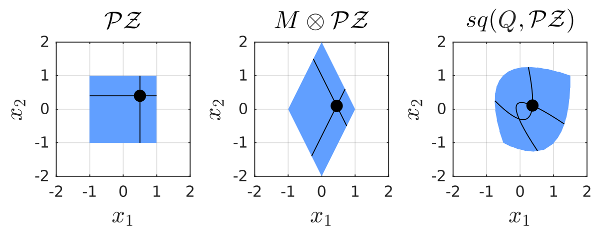

**Figure 2: Sets resulting from the application of different set operations to the polynomial zonotope $\mathcal{PZ}$ from Example 4. The point p and its transformations are depicted by the black dots. Black lines correspond to the points where $\alpha_1 = 0.5 = $ const. and $\alpha_2 = 0.4 = $ const. .**

## 3 REACHABILITY ANALYSIS

We first recall some preliminaries for reachability analysis, followed by our novel algorithm for computing reachable subsets.

### 3.1 Preliminaries

In this paper we consider nonlinear systems of the form

$$\dot{x}(t) = f(x(t), u(t)),\ x(t) \in \mathbb{R}^n,\ u(t) \in \mathbb{R}^m, \tag{14}$$

where $x$ is the state vector and $u$ is the input vector. The reachable set is defined as follows:

Definition 6. *(Reachable Set) Let $\xi(t, x_0, u(\cdot))$ denote the solution to* (14) *for an initial state $x(0) = x_0$ and the input trajectory $u(\cdot)$. The reachable set for an initial set $\mathcal{X}_0 \subset \mathbb{R}^n$ and a set of possible input values $\mathcal{U} \subset \mathbb{R}^m$ is*

$$\mathcal{R}^e_{\mathcal{X}_0}(t) := \Big\{\xi(t, x_0, u(\cdot)) \,\Big|\, x_0 \in \mathcal{X}_0, \forall \tau \in [0,t]\ u(\tau) \in \mathcal{U}\Big\}.$$

The superscript $e$ on $\mathcal{R}^e_{\mathcal{X}_0}(t)$ denotes the exact reachable set, which cannot be computed for general nonlinear systems. Therefore, we compute a tight over-approximation $\mathcal{R}(t) \supseteq \mathcal{R}^e_{\mathcal{X}_0}(t)$ with the operation `reach` defined by Alg. 1, which is taken from [26]. Alg. 1 is based on the abstraction of the nonlinear function $f(\cdot)$ by a Taylor expansion of order $\kappa$:

$$\begin{aligned}\dot{x}_{(i)} &= f_{(i)}(z(t)) \\ &\in \sum_{j=0}^{\kappa} \frac{\Big((z(t) - z^*)^T \nabla\Big)^j f_{(i)}(\tilde{z})}{j!}\Bigg|_{\tilde{z}=z^*} \oplus \mathcal{L}_{(i)}(t),\end{aligned}$$

where we introduce the shorthand $z = [x^T\ u^T]^T$ and the Nabla operator $\nabla = \sum_{i=1}^{n+m} e_i \frac{\partial}{\partial z_{(i)}}$, with $e_i$ being orthogonal unit vectors. The set $\mathcal{L}_{(i)}(t)$ is the Lagrange remainder, defined in [1, Eq. (2)], and the vector $z^* \in \mathbb{R}^{n+m}$ is the expansion point for the Taylor series.

Within Alg. 1, Alg. 2 is executed in line 7 to compute the set of abstraction errors. Alg. 1 and Alg. 2 require the following set operations: `zono` returns an enclosing zonotope, $\boxplus$ denotes the exact addition as defined in [26, Prop. 9], `reduce` denotes the order reduction of a polynomial zonotope (see [26, Prop. 16]), and `enlarge` enlarges a set by a given scalar factor $\lambda$. The operation `taylor` returns the matrices $w, A, B, D, E$ storing the coefficients of the Taylor series evaluation of the nonlinear function $f(\cdot)$ at the expansion point $z^*$. The definitions of the operations $\mathcal{R}^{p,\Delta}$ [1, Eq. (9)], $\texttt{post}^\Delta$ [1, Sec. 4.1], `varInputs` [1, Sec. 4.2], and `lagrangeRemainder` [1, Sec. 4.1] are identical to the ones in [1]. The implementations of all required set operations for polynomial zonotopes are dependency-preserving (see Def. 4), as shown in Table 1. Exact addition and order reduction are not included in Table 1 since they do not apply to all set representations. A visualization of dependency preservation is shown in Fig. 2 for some numerical examples.

The accuracy of the obtained reachable sets is almost entirely determined by the reachable sets $\mathcal{R}(t_s)$ at points in time $t_s$ since only these sets are propagated [1], while the reachable sets of time intervals $\mathcal{R}(\tau_s)$ only fill the time gaps. Consequently, subsequent derivations focus on the reachable sets at points in time $\mathcal{R}(t_s)$.

**Algorithm 1** $\mathcal{R}(t_f) = \texttt{reach}(\mathcal{X}_0, \Psi(\tau_s), z^*_s)$

**Require:** Initial set $\mathcal{X}_0$, sets of initial abstraction errors $\Psi(\tau_s)$, expansion points $z^*_s$, time horizon $t_f$, time step $r$, input set $\mathcal{U}$ represented as a zonotope, factor $\lambda$, desired zonotope order $\rho_d$.
**Ensure:** Final reachable set $\mathcal{R}(t_f)$.

```
1:  $t_0 = 0,\ s = 0,\ \mathcal{R}(0) = \mathcal{X}_0,\ \mathcal{U}^\Delta = \mathcal{U} \oplus (-u^c)$
2:  while $t_s < t_f$ do
3:      $w, A, B, D, E \leftarrow \texttt{taylor}(z^*_s)$
4:      $\mathcal{V}(t_s) = w \oplus Bu^c \oplus \frac{1}{2}\texttt{sq}(D, R(t_s) \times \mathcal{U})$
5:      repeat
6:          $\overline{\Psi}(\tau_s) = \texttt{enlarge}(\Psi(\tau_s), \lambda)$
7:          $\Psi^\Delta(\tau_s) = \texttt{abstrErr}(\mathcal{R}(t_s), \overline{\Psi}(\tau_s))$
8:          $\Psi(\tau_s) = \mathcal{V}(t_s) \oplus \Psi^\Delta(\tau_s)$
9:      until $\Psi(\tau_s) \subseteq \overline{\Psi}(\tau_s)$
10:     $\mathcal{R}^\Delta(\tau_s) = \mathcal{R}^{p,\Delta}(\Psi^\Delta(\tau_s), r)$
11:     $\widehat{\mathcal{R}}(t_{s+1}) = \Big(e^{Ar} \otimes \mathcal{R}(t_s) \boxplus \Gamma(r) \otimes \mathcal{V}(t_s)\Big)$
                $\oplus \mathcal{R}^\Delta(\tau_s)$
12:     $\mathcal{R}(t_{s+1}) = \texttt{reduce}(\widehat{\mathcal{R}}(t_{s+1}), \rho_d)$
13:     $t_{s+1} = t_s + r,\ s := s + 1$
14: end while
```

$\Big\}$ lines 3–12: $\mathcal{R}(t_{s+1}) = \texttt{post}(\mathcal{R}(t_s), \Psi(\tau_s), z^*_s)$

**Algorithm 2** $\Psi^\Delta(\tau_s) = \texttt{abstrErr}(\mathcal{R}(t_s), \overline{\Psi}(\tau_s))$

**Require:** Reachable set $\mathcal{R}(t_s)$, set of applied abstr. errors $\overline{\Psi}(\tau_s)$.
**Ensure:** Set of abstraction errors $\Psi^\Delta(\tau_s)$.

```
1: $\mathcal{Z}_z(t_s) = \texttt{zono}(\mathcal{R}(t_s)) \times \mathcal{U}$
2: $\mathcal{R}^\Delta_z(\tau_s) = \texttt{post}^\Delta(\mathcal{R}(t_s), \overline{\Psi}(\tau_s), A) \times \mathcal{U}$
3: $\mathcal{R}^\Delta_z(\tau_s) = \texttt{zono}(\mathcal{R}^\Delta_z(\tau_s))$
4: $\mathcal{V}^\Delta(\tau_s) = \texttt{varInputs}(\mathcal{Z}_z(t_s), \mathcal{R}^\Delta_z(\tau_s), \mathcal{U}^\Delta, B, D)$
5: $\mathcal{R}(\tau_s) = \mathcal{R}(t_s) \oplus \mathcal{R}^\Delta_z(\tau_s)$
6: $\mathcal{L}(\tau_s) = \texttt{lagrangeRemainder}(\mathcal{R}(\tau_s), E, z^*)$
7: $\Psi^\Delta(\tau_s) = \mathcal{V}^\Delta(\tau_s) \oplus \mathcal{L}(\tau_s)$
```

### 3.2 Reachable Subsets

In this section we show how to efficiently obtain reachable subsets within a pre-computed reachable set as presented in Fig. 3. The main idea is illustrated in Example 5, followed by Lemmas leading to the main result in Theorem 1.

Example 5. *We consider the following two-dimensional system*

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= (1 - x_1^2)x_2 - x_1,\end{aligned} \tag{15}$$

*with the initial set*

$$\mathcal{X}_0 = \left\{ \begin{bmatrix}-1\\1\end{bmatrix} + \begin{bmatrix}0.2\\0\end{bmatrix}\alpha_1 + \begin{bmatrix}0\\0.2\end{bmatrix}\alpha_2 \,\middle|\, \alpha_1, \alpha_2 \in [-1,1] \right\}, \tag{16}$$

*and the initial point $x_0 = [-0.9\ 1.08]^T \in \mathcal{X}_0$. The initial point $x_0$ can be parameterized with $\overline{\alpha} = [0.5\ 0.4]^T$, so that $x_0 = \lfloor \overline{\mathcal{X}_0} \rfloor(\overline{\alpha})$. The computation of the reachable set for a time horizon of $t_f = 1s$ with*

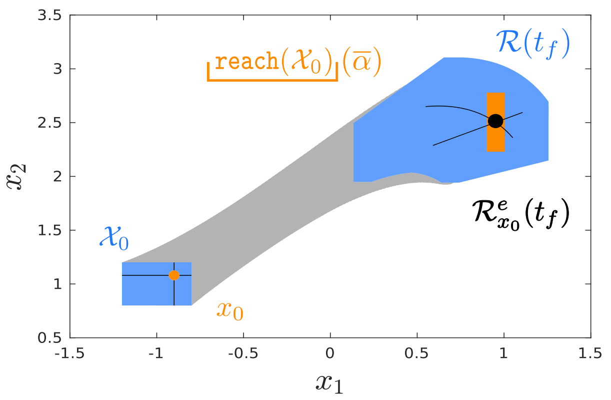

**Figure 3: Visualization of the reachable set from Example 5. The black lines correspond to the points where $\alpha_1 = 0.5 =$ const. and $\alpha_2 = 0.4 =$ const. holds.**

*Alg. 1 yields*

$$\mathcal{R}(t_f) = \Bigg\{ \begin{bmatrix} 0.73 \\ 2.52 \end{bmatrix} + \begin{bmatrix} 0.25 \\ -0.1 \end{bmatrix} \alpha_1 + \begin{bmatrix} 0.26 \\ 0.2 \end{bmatrix} \alpha_2 + \begin{bmatrix} -0.04 \\ -0.09 \end{bmatrix} \alpha_1^2 - \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} \alpha_1 \alpha_2 + \begin{bmatrix} 0.05 \\ 0 \end{bmatrix} \beta_1 + \begin{bmatrix} 0 \\ 0.27 \end{bmatrix} \beta_2 \ \Bigg| \ \alpha_1, \alpha_2, \beta_1, \beta_2 \in [-1, 1] \Bigg\}. \tag{17}$$

*As visualized in Fig. 3, the exact reachable set $\mathcal{R}^e_{x_0}(t_f)$ for the initial point $x_0$ can be enclosed by evaluating (17) for the parameter values $\overline{\alpha} = [0.5\ 0.4]^T$ corresponding to the initial point $x_0$.*

We now prove the correctness of the concept demonstrated by Example 5. Let us start with the computation of the abstraction error returned by Alg. 2:

Lemma 3. *Given the reachable sets $\mathcal{R}^{(1)}(t_s)$,$\mathcal{R}^{(2)}(t_s) \subset \mathbb{R}^n$ with $\mathcal{R}^{(1)}(t_s) \subseteq \mathcal{R}^{(2)}(t_s)$ and the sets of applied abstraction errors $\overline{\Psi}^{(1)}(\tau_s)$, $\overline{\Psi}^{(2)}(\tau_s) \subset \mathbb{R}^n$ with $\overline{\Psi}^{(1)}(\tau_s) \subseteq \overline{\Psi}^{(2)}(\tau_s)$, it holds that*

$$\Psi^{\Delta(1)}(\tau_s) \subseteq \Psi^{\Delta(2)}(\tau_s)$$

$$\text{with } \Psi^{\Delta(1)}(\tau_s) = \mathtt{abstrErr}\big(\mathcal{R}^{(1)}(t_s), \overline{\Psi}^{(1)}(\tau_s)\big) \tag{18}$$

$$\Psi^{\Delta(2)}(\tau_s) = \mathtt{abstrErr}\big(\mathcal{R}^{(2)}(t_s), \overline{\Psi}^{(2)}(\tau_s)\big),$$

*so that* abstrErr *as defined by Alg. 2 satisfies* (7).

Proof. As visualized in Fig. 4, Alg. 2 is a composition of unary set operations that satisfy (6) and binary set operations that satisfy (7). Therefore, it holds according to Lemma 1 that abstrErr as defined by Alg. 2 satisfies (7). □

Using the dependency-preserving properties of operations, we show that Alg. 1 is dependency-preserving. We start with the post operator, i.e., a single iteration of Alg. 1:

Lemma 4. *Given a reachable set $\mathcal{R}(t_s) \subset \mathbb{R}^n$ represented as a polynomial zonotope, it holds that*

$$\forall \alpha \in \mathcal{I}_m : \mathtt{post}\left( \lfloor \mathcal{R}(t_s) \rfloor (\alpha), \Psi(\tau_s), z_s^* \right) \subseteq \lfloor \mathtt{post}(\mathcal{R}(t_s), \mathbf{0}, z_s^*) \rfloor (\alpha), \tag{19}$$

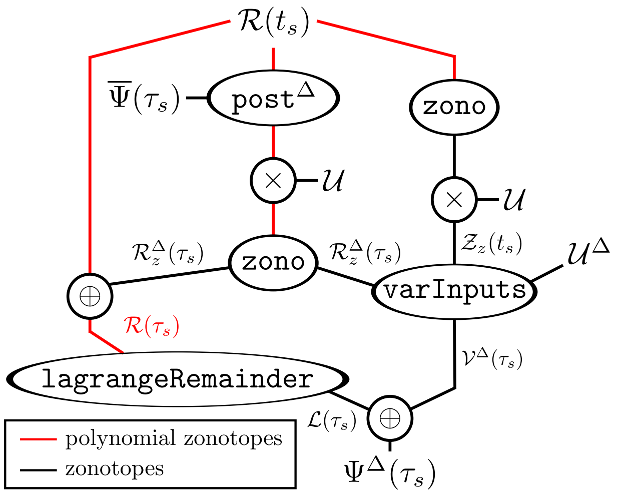

**Figure 4: Flow chart for Alg. 2.**

*where $\Psi(\tau_s)$ is the set of abstraction errors resulting from the computation of* $\mathtt{post}(\mathcal{R}(t_s), \mathbf{0}, z_s^*)$ *and $z_s^*$ is the expansion point of the Taylor series.*

Proof. A flow chart for the post operation in Alg. 1 is shown in Fig. 5. First, we consider the operations A, B, and C illustrated by the blocks in Fig. 5:

**Operation** A**:** As visualized in Fig. 5, operation A is defined by a composition of dependency-preserving operations so that A is dependency-preserving according to Lemma 2.

**Operation** B**:** For the repeat-until-loop in Lines 5-9 of Alg. 1 it is sufficient to consider only the values from the last iteration since only these are applied in subsequent computations. The operation abstrErr satisfies (7) according to Lemma 3. As visualized in Fig. 5, the binary operation B is defined by a composition of unary operations satisfying (6) and binary operations satisfying (7), so that B satisfies (7) according to Lemma 1.

**Operation** C**:** According to Fig. 5, operation C is defined as

$$\mathtt{C}(\mathcal{R}(t_s)) = \mathtt{A}(\mathcal{R}(t_s)) \oplus \mathtt{B}(\mathcal{R}(t_s), \Psi(\tau_s)). \tag{20}$$

Since A is dependency-preserving, it holds according to (5) that

$$\forall \alpha \in \mathcal{I}_m : \ \mathtt{A}\left( \lfloor \mathcal{R}(t_s) \rfloor (\alpha) \right) \subseteq \lfloor \mathtt{A}(\mathcal{R}(t_s)) \rfloor (\alpha). \tag{21}$$

Since $\lfloor \mathcal{R}(t_s) \rfloor (\alpha) \subseteq \mathcal{R}(t_s)$ and B satisfies (7),

$$\forall \alpha \in \mathcal{I}_m : \ \mathtt{B}\left( \lfloor \mathcal{R}(t_s) \rfloor (\alpha), \Psi(\tau_s) \right) \subseteq \mathtt{B}(\mathcal{R}(t_s), \Psi(\tau_s)). \tag{22}$$

Furthermore, since B returns a zonotope that does not contain the polynomial zonotope factors $\alpha$,

$$\forall \alpha \in \mathcal{I}_m : \lfloor \mathtt{A}(\mathcal{R}(t_s)) \rfloor (\alpha) \oplus \mathtt{B}(\mathcal{R}(t_s), \Psi(\tau_s)) = \lfloor \mathtt{A}(\mathcal{R}(t_s)) \oplus \mathtt{B}(\mathcal{R}(t_s), \Psi(\tau_s)) \rfloor (\alpha). \tag{23}$$

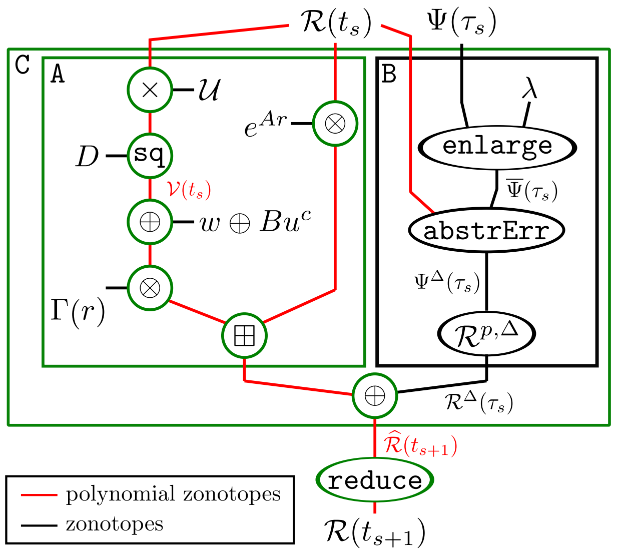

**Figure 5: Flow chart for Alg. 1. Dependency-preserving operations are marked in green.**

Consequently,

$$\begin{aligned}
&\forall \alpha \in \mathcal{I}_m: \\
&\mathsf{C}\left(\lfloor \mathcal{R}(t_s) \rfloor(\alpha)\right) \overset{(20)}{=} \mathsf{A}\left(\lfloor \mathcal{R}(t_s) \rfloor(\alpha)\right) \oplus \mathsf{B}\left(\lfloor \mathcal{R}(t_s) \rfloor(\alpha), \Psi(\tau_s)\right) \\
&\overset{(21)}{\subseteq} \lfloor \mathsf{A}(\mathcal{R}(t_s)) \rfloor(\alpha) \oplus \mathsf{B}\left(\lfloor \mathcal{R}(t_s) \rfloor(\alpha), \Psi(\tau_s)\right) \\
&\overset{(22)}{\subseteq} \lfloor \mathsf{A}(\mathcal{R}(t_s)) \rfloor(\alpha) \oplus \mathsf{B}(\mathcal{R}(t_s), \Psi(\tau_s)) \\
&\overset{(23)}{=} \lfloor \mathsf{A}(\mathcal{R}(t_s)) \oplus \mathsf{B}(\mathcal{R}(t_s), \Psi(\tau_s)) \rfloor(\alpha) \overset{(20)}{=} \lfloor \mathsf{C}(\mathcal{R}(t_s)) \rfloor(\alpha),
\end{aligned}$$

which proves that operation C is dependency-preserving (see (5)).
As visualized in Fig. 5, the post operation is defined by the composition of dependency-preserving operations resulting in a dependency-preserving operation according to Lemma 2. □

Using the result for a single iteration, we now prove that Alg. 1 is dependency-preserving:

Lemma 5. *Given an initial set $\mathcal{X}_0 \subset \mathbb{R}^n$ represented as a polynomial zonotope, it holds that*

$$\begin{aligned}
\forall \alpha \in \mathcal{I}_m: &\ \mathtt{reach}\left(\lfloor \mathcal{X}_0 \rfloor(\alpha), \Psi(\tau_s), z_s^*\right) \\
&\subseteq \lfloor \mathtt{reach}(\mathcal{X}_0, \mathbf{0}, z_s^*) \rfloor(\alpha)
\end{aligned} \tag{24}$$

*where $\Psi(\tau_s)$, $s = 1, \ldots, N$ are the sets of abstraction errors resulting from the computation of* reach$(\mathcal{R}(t_s), \mathbf{0}, z_s^*)$ *in Alg. 1, $z_s^*$, $s = 1, \ldots, N$ are the expansion points of the Taylor series, $N = \lceil \frac{t_f}{r} \rceil$ is the number of time steps, $t_f$ is the time horizon, and $r$ is the time step size.*

Proof. The reach operation as defined by Alg. 1 can be expressed equivalently as

$$\mathcal{R}(t_f) = \mathtt{reach}(\mathcal{X}_0, \Psi(\tau_s), z_s^*) = \mathtt{post}_N(\mathcal{X}_0, \Psi(\tau_s), z_s^*),$$

where post$_N$ denotes the $N$-times consecutive composition of the post operation

$$\mathcal{R}(t_f) = \underbrace{\mathtt{post}(\ldots \mathtt{post}(\mathcal{X}_0, \Psi(\tau_1), z_1^*) \ldots, \Psi(\tau_N), z_N^*)}_{\mathtt{post}_N(\mathcal{X}_0, \Psi(\tau_s), z_s^*)}.$$

Then, since the post operation is dependency-preserving (see Lemma 4) and the composition of dependency-preserving operations yields a dependency-preserving operation (see Lemma 2), operation reach$(\mathcal{X}_0, \Psi(\tau_s), z_s^*)$ = post$_N(\mathcal{X}_0, \Psi(\tau_s), z_s^*)$ is dependency-preserving. □

Finally, we formulate the main result:

Theorem 1. *Given an initial set $\mathcal{X}_0 \subset \mathbb{R}^n$ represented as a polynomial zonotope,*

$$\begin{aligned}
\forall \alpha \in \mathcal{I}_m: \mathcal{R}^e_{x_0}(t_f) &\subseteq \lfloor \mathtt{reach}(\mathcal{X}_0, \mathbf{0}, z_s^*) \rfloor(\alpha) \\
\text{with } x_0 &= \lfloor \mathcal{X}_0 \rfloor(\alpha),
\end{aligned} \tag{25}$$

*where $z_s^*$ and* reach *are as in Lemma 5.*

Proof. As shown in [1], Alg. 1 computes an over-approximation of the exact reachable set as follows:

$$\mathcal{R}^e_{x_0}(t_f) \subseteq \mathtt{reach}\left(\lfloor \mathcal{X}_0 \rfloor(\alpha), \Psi(\tau_s), z_s^*\right), \tag{26}$$

where $\Psi(\tau_s)$, $s = 1, \ldots, N$ are the sets of abstraction errors resulting from the computation of reach$(\mathcal{X}_0, \mathbf{0}, z_s^*)$ so that according to Lemma 5 we have:

$$\begin{aligned}
\forall \alpha \in \mathcal{I}_m: \mathcal{R}^e_{x_0}(t_f) &\overset{(26)}{\subseteq} \mathtt{reach}\left(\lfloor \mathcal{X}_0 \rfloor(\alpha), \Psi(\tau_s), z_s^*\right) \\
&\subseteq \lfloor \mathtt{reach}(\mathcal{X}_0, \mathbf{0}, z_s^*) \rfloor(\alpha),
\end{aligned}$$

which concludes the proof. □

Since Theorem 1 holds for all points $x_0 \in \mathcal{X}_0$ inside the initial set $\mathcal{X}_0$, it is obvious that Theorem 1 equally holds for all sets $\widehat{\mathcal{X}}_0 \subseteq \mathcal{X}_0$. Furthermore, Theorem 1 also holds at intermediate time steps $\mathcal{R}(t_s)$, $s = 1, \ldots, N$ with $N = \lceil \frac{t_f}{r} \rceil$. Since the time interval reachable set $\mathcal{R}(\tau_s)$ is computed by adding uncertainty to the time point reachable set $\mathcal{R}(t_s)$ (see Line 5 of Alg. 2), Theorem 1 equally holds for the reachable set $\mathcal{R}(\tau_s)$ of time intervals $\tau_s = [t_s, t_{s+1}]$.

## 3.3 Computational Complexity

We now derive the computational complexity for extracting a reachable subset from the final reachable set according to (13), where

$$f_{G,E}(\alpha) = \sum_{i=1}^{h} \Bigg( \underbrace{\prod_{k=1}^{m} \alpha_k^{E_{(k,i)}}}_{P_i} \Bigg) G_{(\cdot,i)}. \tag{27}$$

Computation of each $P_i$ in (27) requires $m$ exponentiations and $m-1$ multiplications, and computation of $P_i G_{(\cdot,i)}$ requires $n$ multiplications. Since there are $h$ terms $P_i G_{(\cdot,i)}$ in (27), computation of all terms requires $h(2m+n-1)$ operations. The computation of the outer sum in (27) requires $n(h-1)$ additions, so that the computation of $f_{G,E}(\alpha)$ requires in total $h(2m+n-1) + n(h-1)$ operations. It holds for the number of dependent factors $m$ and the number of dependent generators $h$ that $m = c_m n$ and $h = c_h n$ with $c_m, c_h \in \mathbb{R}_{\geq 0}$. The complexity for the extraction of a reachable

**Table 2: Comp. times for the numerical examples in [s].**

| Application | Computation Time in [s] | |
|---|---|---|
| | New Approach | Prev. Solution |
| Falsification (2D) | 0.12 | 0.38 |
| Falsification (12D) | 0.13 | 10.4 |
| Optimization | 0.13 | 172 |
| Safe Control | 0.06 | 21.4 |

subset is therefore $O(h(2p + n - 1) + n(h - 1)) = O(n^2)$. Since the computation of the reachable set with the conservative polynomialization approach has complexity $O(n^5)$ [1], our novel extraction of reachable subsets is computational much more efficient.

## 4 APPLICATIONS

There are a lot of different applications for the result presented in this paper. In this section, we introduce some of the possible applications and demonstrate their efficiency using numerical examples. However, due to limited space, we only present the general concepts and omit implementation details. The implementation of our approach including all numerical examples will be made publicly available in the next release of the CORA toolbox [2]. Also, we carried out all computations in MATLAB on a 2.9GHz quad-core i7 processor with 32GB memory.

### 4.1 Falsifying States

Let us consider a specification defined by a linear inequality constraint $a^T x \leq b$, $a \in \mathbb{R}^n, b \in \mathbb{R}$. If the final reachable set $\mathcal{R}(t_f)$ for the initial set $\mathcal{X}_0$ does not fulfill the specification, it would be helpful to know which initial states result in a violation. According to Theorem 1, the states inside the set $\mathcal{S} \subseteq \mathcal{X}_0$ defined as

$$\mathcal{S} = \bigcup_{\alpha \in \mathcal{B}} \llcorner \mathcal{X}_0 \lrcorner (\alpha)$$
$$\text{with } \mathcal{B} = \left\{ \alpha \in \mathcal{I}_m \mid a^T \otimes \llcorner \mathcal{R}(t_f) \lrcorner (\alpha) \leq b \right\}$$

are guaranteed to fulfill the specification. The set of states $\mathcal{F} \subseteq \mathcal{X}_0$ that potentially result in the violation of the specification can consequently be computed as $\mathcal{F} = \mathcal{X}_0 \setminus \mathcal{S}$.

Theorem 1 is used to efficiently determine an initial point as well as a suitable input trajectory falsifying the specification. The initial point $x^* \in \mathcal{X}_0$ that results in the largest violation of the specification can be computed by solving the optimization problem

$$x^* = \llcorner \mathcal{X}_0 \lrcorner (\alpha^*)$$
$$\text{with } \alpha^* = \underset{\alpha \in \mathcal{I}_p}{\operatorname{argmax}} \; a^T \otimes \llcorner \mathcal{R}(t_f) \lrcorner (\alpha). \quad (28)$$

Since this reduces the *safety falsification* task to finding a suitable input trajectory, falsifying trajectories can be found much more efficiently as shown by an example:

Example 6. *We consider the reachability problem from Example 5 in combination with the specification $x_1 + 2x_2 \leq 6.4$. As shown in Fig. 6, the final reachable set violates the constraint, where in green the safe states $\mathcal{S}$ and in orange the falsifying states $\mathcal{F}$ are visualized. Since the reachability problem from Example 5 does not include uncertain inputs, a falsifying trajectory can simply be determined by solving the optimization problem in* (28)*, which takes* 0.12 *seconds (see Table 2). As a comparison, we also determined a falsifying trajectory using the simulated annealing algorithm of the falsification toolbox S-TALIRO* [4]. *Since the simulated annealing algorithm is non-deterministic we computed the average computation time from* 10 *executions, which results in the value* 0.38 *seconds (see Table 2).*

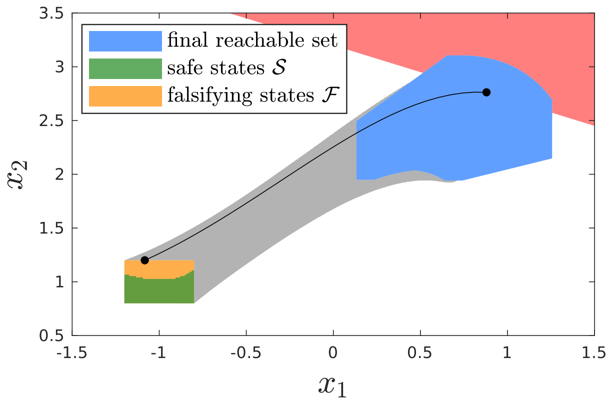

**Figure 6: Visualization of Example 6. The determined falsifying trajectory is shown in black.**

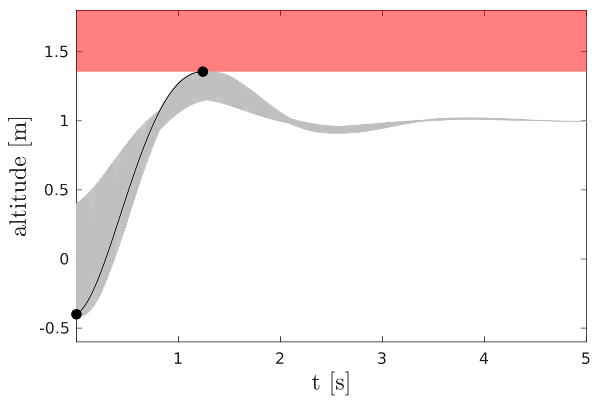

**Figure 7: Visualization of Example 7. The determined falsifying trajectory is shown in black.**

We demonstrate the scalability of our approach with the system dimension by a second example:

Example 7. *We consider the 12-dimensional quadrotor benchmark from the ARCH 19 competition* [25] *in combination with the specification $x_3 \leq 1.355m \; \forall t \in [0s, 5s]$ (see Fig. 7). Since the benchmark does not include uncertain inputs, a falsifying trajectory can be computed by solving* (28)*, which takes* 0.13 *seconds (see Table 2). The computation time of the simulated annealing algorithm from S-TALIRO averaged over* 10 *executions is* 10.4 *seconds (see Table 2).*

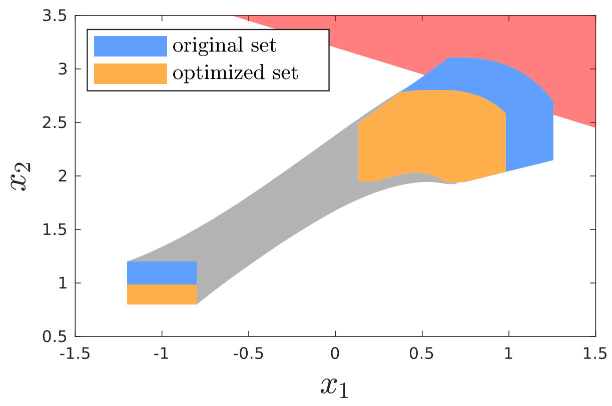

**Figure 8: Visualization of Example 8.**

## 4.2 Optimization over Reachable Sets

Since reachability analysis is computational expensive, optimization over reachable sets is often infeasible. However, with our new approach it is possible to achieve major speed-ups for optimizing over reachable sets:

Example 8. *We consider the reachability analysis problem from Example 5 and the inequality constraint* $[1\ 2]\ x \leq 6.4$*. As shown in Fig. 8, the final reachable set violates the constraint. We want to determine the modified initial set* $\mathcal{X}_0^* \subset \mathcal{X}_0$ *with the maximum volume for which the final reachable set satisfies the constraint, which can be formulated as an optimization problem: We use the upper and lower bounds* $\underline{\alpha}_1, \overline{\alpha}_1, \underline{\alpha}_2, \overline{\alpha}_2$ *for the factors* $\alpha_1$ *and* $\alpha_2$ *as the variables for the optimization problem. Consequently, the initial set from Example 5 becomes*

$$\mathcal{X}_0(\underline{\alpha}_1, \overline{\alpha}_1, \underline{\alpha}_2, \overline{\alpha}_2) = \left\{ \begin{bmatrix} -1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0.2 \\ 0 \end{bmatrix} \alpha_1 + \begin{bmatrix} 0 \\ 0.2 \end{bmatrix} \alpha_2 \ \middle|\ \alpha_1 \in [\underline{\alpha}_1, \overline{\alpha}_1],\ \alpha_2 \in [\underline{\alpha}_2, \overline{\alpha}_2] \right\},$$

*where we denote the dependence of the initial set on the bounds* $\underline{\alpha}_1, \overline{\alpha}_1, \underline{\alpha}_2, \overline{\alpha}_2$ *by* $\mathcal{X}_0(\underline{\alpha}_1, \overline{\alpha}_1, \underline{\alpha}_2, \overline{\alpha}_2)$*. The optimization problem can be formulated as*

$$\begin{aligned} &\max_{\underline{\alpha}_1, \overline{\alpha}_1, \underline{\alpha}_2, \overline{\alpha}_2} \ \texttt{volume}\big(\mathcal{X}_0(\underline{\alpha}_1, \overline{\alpha}_1, \underline{\alpha}_2, \overline{\alpha}_2)\big) \\ &s.t.\ [1\ 2] \otimes \texttt{reach}\big(\mathcal{X}_0(\underline{\alpha}_1, \overline{\alpha}_1, \underline{\alpha}_2, \overline{\alpha}_2), \mathbf{0}, z_s^*\big) \leq 6.4, \end{aligned} \tag{29}$$

*where the operator* volume$(\mathcal{S})$ *returns the volume of a set* $\mathcal{S} \subset \mathbb{R}^n$*. The optimization problem* (29) *is hard to solve since each evaluation of the constraint function requires the computationally expensive execution of the reachability algorithm. However, if we exploit the dependency preservation between the initial states and the reachable states introduced in this paper, the constraint can be equivalently formulated as*

$$\max_{\substack{\alpha_1 \in [\underline{\alpha}_1, \overline{\alpha}_1] \\ \alpha_2 \in [\underline{\alpha}_2, \overline{\alpha}_2]}} [1\ 2] \otimes \underline{|\mathcal{R}(t_f)|} \left([\alpha_1\ \alpha_2]^T\right) \leq 6.4, \tag{30}$$

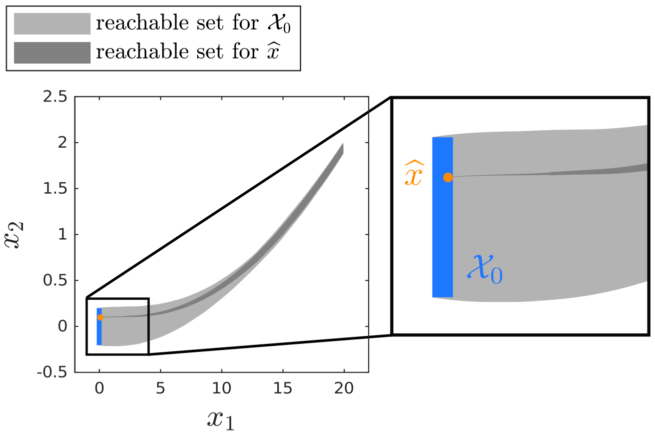

**Figure 9: Visualization of Example 9 with the initial set $\mathcal{X}_0$ and the measured state $\widehat{x}$.**

*where* $\mathcal{R}(t_f)$ *is the final reachable set (see* (17)*). Inserting the numerical values for* $\mathcal{R}(t_f)$ *from* (17) *into* (30) *yields*

$$\max_{\substack{\alpha_1 \in [\underline{\alpha}_1, \overline{\alpha}_1] \\ \alpha_2 \in [\underline{\alpha}_2, \overline{\alpha}_2]}} 0.05\alpha_1 + 0.66\alpha_2 - 0.22\alpha_1^2 - 0.2\alpha_1\alpha_2 \leq 0.04.$$

*The volume of the initial set can be computed as*

$$\texttt{volume}\big(\mathcal{X}_0(\underline{\alpha}_1, \overline{\alpha}_1, \underline{\alpha}_2, \overline{\alpha}_2)\big) = 0.04\left(\overline{\alpha}_1 - \underline{\alpha}_1\right)\left(\overline{\alpha}_2 - \underline{\alpha}_2\right),$$

*which simplifies the optimization problem* (29) *to*

$$\begin{aligned} &\max_{\underline{\alpha}_1, \overline{\alpha}_1, \underline{\alpha}_2, \overline{\alpha}_2} \ 0.04\left(\overline{\alpha}_1 - \underline{\alpha}_1\right)\left(\overline{\alpha}_2 - \underline{\alpha}_2\right) \\ &s.t. \max_{\substack{\alpha_1 \in [\underline{\alpha}_1, \overline{\alpha}_1] \\ \alpha_2 \in [\underline{\alpha}_2, \overline{\alpha}_2]}} 0.05\alpha_1 + 0.66\alpha_2 - 0.22\alpha_1^2 - 0.2\alpha_1\alpha_2 \leq 0.04. \end{aligned} \tag{31}$$

*The solution for the optimization problem* (31) *is visualized in Fig. 8. As shown in Table 2 solving the optimization problem* (29) *using MATLABs* `fmincon` *with the* `interior-point` *algorithm takes* 221 *seconds, but solving the simplified optimization problem* (31) *only takes* 0.1 *seconds.*

## 4.3 Motion-Primitive Based Control

In this section, we consider a scenario where a maneuver automaton is used to control a system. The approach in [39] constructs a provably safe maneuver automaton by using reachability analysis. In particular, for each motion-primitive of the maneuver automaton, the reachable set for an initial set $\mathcal{X}_0$ is computed offline. During online application our novel approach can directly extract the reachable set for a measured state $\widehat{x}$ from the offline-computed reachable set. Generally, since the reachable set for $\widehat{x}$ is much smaller than the reachable set for $\mathcal{X}_0$, planning with the reachable set for $\widehat{x}$ greatly reduces the conservatism as demonstrated with a numerical example:

Example 9. *We consider the example of the turn-left maneuver of an autonomous car from* [39, Sec. 6] *with the measured velocity* $\widehat{v} = 20.2\frac{m}{s}$*, the measured orientation* $\widehat{\phi} = 0.01rad$*, and the measured* *position $\widehat{x}_1 = 0.1m$, $\widehat{x}_2 = 0.1m$. The reachable set for the initial set $\mathcal{X}_0$ and reachable set extracted for then measured point $\widehat{x} = [\widehat{v}\ \widehat{\phi}\ \widehat{x}_1\ \widehat{x}_2]^T$ is visualized in Fig. 9. As shown in Table 2, the extraction of the reachable set for the measured point from the offline-computed reachable set is significantly faster than the computation of the reachable set using Alg. 1.*

## 5 CONCLUSION

In this paper, we showed that the computation of the reachable set for nonlinear systems with the conservative polynomialization approach using polynomial zonotopes preserves the dependence between the reachable states and the initial states. Since this novel method supports the efficient computation of reachable subsets inside pre-computed reachable sets, many possible applications are opening up. For the three applications *safety falsification*, *optimization over reachable sets*, and *motion-primitive based control* we demonstrated with numerical examples that using our novel method results in significant speed-ups compared to the previous solutions. Besides, our method for extracting reachable subsets has complexity $\mathcal{O}(n^2)$ and is therefore computationally much more efficient than to compute the reachable subset from scratch, which has complexity $\mathcal{O}(n^5)$. Furthermore, this paper provides to some extent a method for unifying reachability analysis and falsification.

## ACKNOWLEDGMENTS

The author gratefully acknowledges financial support from the German Research Foundation (DFG) project faveAC under grant number AL 1185/5-1.

## REFERENCES

[1] M. Althoff. 2013. Reachability Analysis of Nonlinear Systems using Conservative Polynomialization and Non-Convex Sets. In *Hybrid Systems: Computation and Control*. 173–182.
[2] M. Althoff. 2015. An Introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*. 120–151.
[3] M. Althoff and et al. 2008. Reachability Analysis of Nonlinear Systems with Uncertain Parameters using Conservative Linearization. In *Proc. of the 47th IEEE Conference on Decision and Control*. 4042–4048.
[4] Y. Annapureddy and et al. 2011. S-TaLiRo: A Tool for Temporal Logic Falsification for Hybrid Systems. In *Proc. of Tools and Algorithms for the Construction and Analysis of Systems*. 254–257.
[5] E. Asarin and et al. 2007. Hybridization Methods for the Analysis of Nonlinear Systems. *Acta Informatica* 43 (2007), 451–476.
[6] S. Bak and P. S. Duggirala. 2017. HyLAA: A Tool for Computing Simulation-Equivalent Reachability for Linear Systems. In *Proc. of the 20th International Conference on Hybrid Systems: Computation and Control*. 173–178.
[7] S. Bak and P. S. Duggirala. 2017. Simulation-Equivalent Reachability of Large Linear Systems with Inputs. In *Proc. of International Conference on Computer Aided Verification*. 401–420.
[8] L. Benvenuti and et al. 2014. Assume-guarantee verification of nonlinear hybrid systems with ARIADNE. *International Journal of Robust and Nonlinear Control* 24 (2014), 699–724.
[9] S. Bogomolov and et al. 2019. Falsification of hybrid systems using symbolic reachability and trajectory splicing. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*. ACM, 1–10.
[10] Sergiy Bogomolov and et al. 2019. JuliaReach: a toolbox for set-based reachability. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*. ACM, 39–44.
[11] X. Chen and et al. 2012. Taylor Model Flowpipe Construction for Non-linear Hybrid Systems. In *Proc. of the 33rd IEEE Real-Time Systems Symposium*.
[12] X. Chen and et al. 2013. Flow*: An Analyzer for Non-Linear Hybrid Systems. In *Proc. of Computer-Aided Verification (LNCS 8044)*. Springer, 258–263.
[13] X. Chen, S. Sankaranarayanan, and E. Ábrahám. 2014. Under-approximate flowpipes for non-linear continuous systems. In *2014 Formal Methods in Computer-Aided Design (FMCAD)*. IEEE, 59–66.
[14] A. Chutinan and B. H. Krogh. 2003. Computational Techniques for Hybrid System Verification. *IEEE Trans. Automat. Control* 48, 1 (2003), 64–75.
[15] T. Dang and et al. 2010. Accurate Hybridization of Nonlinear Systems. In *Hybrid Systems: Computation and Control*. 11–19.
[16] P. S. Duggirala and et al. 2015. C2E2: A verification tool for stateflow models. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 68–82.
[17] P. S. Duggirala and M. Viswanathan. 2016. Parsimonious, Simulation Based Verification of Linear Systems. In *Proc. of International Conference on Computer Aided Verification*. 477–494.
[18] G. Frehse and et al. 2011. SpaceEx: Scalable Verification of Hybrid Systems. In *Proc. of the 23rd International Conference on Computer Aided Verification (LNCS 6806)*. Springer, 379–395.
[19] A. Girard and et al. 2006. Efficient Computation of Reachable Sets of Linear Time-Invariant Systems with Inputs. In *Hybrid Systems: Computation and Control (LNCS 3927)*. Springer, 257–271.
[20] A. Girard and C. Le Guernic. 2008. Efficient Reachability Analysis for Linear Systems using Support Functions. In *Proc. of the 17th IFAC World Congress*. 8966–8971.
[21] E. Goubault and S. Putot. 2015. A zonotopic framework for functional abstractions. *Formal Methods in System Design* 47, 3 (2015), 302–360.
[22] Eric Goubault and Sylvie Putot. 2019. Inner and outer reachability for the verification of control systems.. In *HSCC*. 11–22.
[23] J. Hoefkens and et al. 2001. *Scientific Computing, Validated Numerics, Interval Methods*. Springer, Chapter Verified High-Order Integration of DAEs and Higher-Order ODEs, 281–292.
[24] F. Immler. 2015. Tool Presentation: Isabelle/HOL for Reachability Analysis of Continuous Systems. In *Proc. of the 2nd Workshop on Applied Verification for Continuous and Hybrid Systems*. 180–187.
[25] F. Immler and et al. 2019. ARCH-COMP19 Category Report: Continuous and Hybrid Systems with Nonlinear Dynamics. In *ARCH19. 6th International Workshop on Applied Verification of Continuous and Hybrid Systems*.
[26] N. Kochdumper and M. Althoff. 2019. Sparse Polynomial Zonotopes: A Novel Set Representation for Reachability Analysis. *arXiv preprint arXiv:1901.01780* (2019).
[27] H. Kong and et al. 2017. Safety verification of nonlinear hybrid systems based on invariant clusters. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*. ACM, 163–172.
[28] A. B. Kurzhanski and P. Varaiya. 2000. Ellipsoidal Techniques for Reachability Analysis. In *Hybrid Systems: Computation and Control (LNCS 1790)*. Springer, 202–214.
[29] M. Li and et al. 2018. Safe over-and under-approximation of reachable sets for autonomous dynamical systems. In *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 252–270.
[30] J. Liu and et al. 2011. Computing semi-algebraic invariants for polynomial dynamical systems. In *Proceedings of the ninth ACM international conference on Embedded software*. ACM, 97–106.
[31] K. Makino and M. Berz. 2009. Rigorous Integration of Flows and ODEs using Taylor Models. In *Proc. of Symbolic-Numeric Computation*. 79–84.
[32] N. Matringe and et al. 2010. Generating invariants for non-linear hybrid systems by linear algebraic methods. In *International Static Analysis Symposium*. Springer, 373–389.
[33] I. M. Mitchell and et al. 2005. A Time-Dependent Hamilton–Jacobi Formulation of Reachable Sets for Continuous Dynamic Games. *IEEE Trans. Automat. Control* 50 (2005), 947–957.
[34] P. Prabhakar and M. Viswanathan. 2011. A Dynamic Algorithm for Approximate Flow Computations. In *Hybrid Systems: Computation and Control*. 133–142.
[35] R. Ray and A. Gurung. 2015. Parallel state space exploration of linear systems with inputs using XSpeed. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. ACM, 285–286.
[36] Julien Alexandre Dit Sandretto and Alexandre Chapoutot. 2016. DynBEX: a Differential Constraint Library for Studying Dynamical Systems.
[37] Stefan Schupp and et al. 2017. HyPro: A C++ library of state set representations for hybrid systems reachability analysis. In *NASA Formal Methods Symposium*. Springer, 288–294.
[38] B. Schürmann and M. Althoff. 2016. Closed-Form Expressions of Convex Combinations for Controller Design. In *Proc. of American Control Conference*. 2795–2801.
[39] B. Schürmann and M. Althoff. 2017. Convex Interpolation Control with Formal Guarantees for Disturbed and Constrained Nonlinear Systems. In *Proc. of Hybrid Systems: Computation and Control*. 121–130.