Utilizing Dependencies to Obtain Subsets of Reachable Sets

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

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

Matthias Althoff  
Technische Universität München 
althoff@tum.de

ABSTRACT

Reachability analysis has become a fundamental method supporting formally-correct synthesis, robust model predictive control, set-based observers, fault detection, invariant computation, and conformance checking, to name only 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.

Keywords

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

1. INTRODUCTION

In this work, we present a novel concept to directly extract a reachable set within a precomputed 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 $\hat{R}$ for any initial set $\hat{X}_0 \subseteq X_0$ directly from the reachable set $R$ by evaluation of an analytical equation (see Fig. 1), which is computationally much more efficient than computing $\hat{R}$ with a reachability algorithm. As we demonstrate in Sec. 3 this method offers great advantages for applications where reachable sets have to be computed for many different subsets $X_0 \subseteq X_0$, like e.g., 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 e.g., polytopes [17], zonotopes [19], ellipsoids [27], support functions [20], and oriented hyperrectangles [35]. Other approaches use simulations to compute reachable set [7, 16]. Examples of tools for reachability analysis of linear systems are C2E2 [15], CORA [2], HyDRA [36], Hylaa [6], Julia Reach [10], SpaceEx [17], and XSpeed [34].

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 [26, 29, 31]. Optimization-based approaches reformulate reachability analysis as an optimization problem [13, 32]. The approach in [13] optimizes the outward translation of polytope halfspaces to obtain a flowpipe, whereas [32] expresses reachability analysis with Hamilton-Jacobi equations. Other approaches abstract the solution space directly: The work in [20] uses validated simulations for the construction of bounded flowpipes, [53] approximates the solution of the ODE with Bernstein polynomials; Taylor models computed from iterations, such as the Picard iteration, were first proposed in [22], and later extended to include uncertain inputs [11]. Approaches based on an abstraction of the state space compute simplified differential 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][14]. Recent approaches extend this concept to the abstraction of the non-linear dynamics by polynomials [2], which results in a tighter enclosure of the reachable set. Examples of tools for reachability analysis of nonlinear systems are Adriadne [8], C2E2 [15], CORA [2], Dynabex [35], Flow* [12], Isabelle/HOL [23], and Julia Reach [10].

Our new method is not only applicable for reachability analysis, but also for 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. In graph theory, symbolic reachability analysis can be used to determine falsifying initial states [18]. The approach in [9] combines symbolic reachability analysis with reachability analysis for continuous systems 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 [21][28].

In this work, we preserve the relation between the initial and the reachable states for nonlinear systems. We use polynomial zonotopes instead of Taylor models since polynomial zonotopes are a generalization of Taylor models [25, Prop. 4]. Our novel approach enables us to almost instantly obtain subsets within a pre-computed reachable set. As we demonstrate with numerical examples, our novel method results in a significant reduction of the computation time for safety falsification, optimization over reachable sets, and motion-primitive based control. To some extent this work provides a 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 \( A \in \mathbb{R}^{n \times m} \), \( A_{(i,j)} \) represents the \( i \)-th matrix row, \( A_{(i,j)} \) the \( j \)-th column, and \( A_{(i,:)} \) the \( j \)-th entry of matrix row \( i \). The concatenation of two matrices \( C \) and \( D \) is denoted by \([C, D]\). The symbols 0 and 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 \( S \subset \mathbb{R}^n \) is defined as \( M \odot S = \{Ms \mid s \in S\} \), the Minkowski addition of two sets \( S_1 \subset \mathbb{R}^n \) and \( S_2 \subset \mathbb{R}^n \) is defined as \( S_1 \oplus S_2 = \{s_1 + s_2 \mid s_1 \in S_1, s_2 \in S_2\} \), and the Cartesian product of two sets \( S_1 \subset \mathbb{R}^n \) and \( S_2 \subset \mathbb{R}^n \) is defined as \( S_1 \times S_2 = \{s_1 s_2 \mid s_1 \in S_1, s_2 \in S_2\} \). Given two set operations \( \Lambda \) and \( \Xi \), and a set \( S \subset \mathbb{R}^n \), the composition of the set operations is denoted by \( \Lambda(\Xi(S)) = (\Lambda \circ \Xi)(S) \). The power set of a set \( S \subset \mathbb{R}^n \) is denoted by \( 2^S \). We further introduce a \( n \)-dimensional interval as \( I := [l, u] \), \( \forall i(l(i) \leq u(i), l, u \in \mathbb{R}^n \). The unit hypercube \([-1, 1] \subset \mathbb{R}^n \) is denoted by \( I_n \). Given a center vector \( c \in \mathbb{R}^n \) and a generator matrix \( G \in \mathbb{R}^{n \times m} \), a zonotope is \( I := \{c + \sum_{i=1}^m a_i G_{(i,:)} \mid a_i \in [-1, 1]\} \). For a concise notation, we use the shorthand \( I = \langle c, G \rangle \).

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 I, 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, which are defined first:

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

\[
P := \left\{ \sum_{i=1}^m \delta_i v_i \left| \delta_i \geq 0, \sum_{i=1}^m \delta_i = 1 \right. \right\}.
\]

We use the shorthand \( P = \langle [v_1, \ldots, v_m] \rangle \).

**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 \( P \subset \mathbb{R}^n \) is defined as

\[
P := \{ x \mid Ax \leq b \}.
\]

Each point \( p \in P \) can be parameterized by specific values \( \delta_i \) when using the V-representation (not possible for the H-representation):

\[
p = \sum_{i=1}^m \delta_i v_i, \quad \delta_i \geq 0, \sum_{i=1}^m \delta_i = 1.
\]

In general, the above parameterization is not unique [37]. For parameterized sets, we introduce evaluation functions:

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

\[
\Xi(d) = \mathcal{S},
\]

where the parameter domain \( \mathcal{D} \subset \mathbb{R}^m \) satisfies

\[
\bigcup_{d \in \mathcal{D}} \Xi(d) = S.
\]

**Example 1.** For a polytope \( P = \langle [v_1, \ldots, v_m] \rangle \) in V-representation, the parameter domain is

\[
\mathcal{D} = \left\{ [\delta_1, \ldots, \delta_m] \right| \delta_i \geq 0, \sum_{i=1}^m \delta_i = 1 \right\}
\]

and the evaluation function is

\[
\Xi_P(\delta) = \sum_{i=1}^m \delta_i v_i.
\]

2.2 Dependency Preservation

We require a set representation that preserves the relation between states for all relevant set operations. We first demonstrate dependency preservation for linear maps of V-representations:
Table 1: Characterization of set representations with respect to parameterization and dependency preservation of set operations.

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

Example 2. Given a scalar \( M \in \mathbb{R} \) and a one-dimensional polytope \( P = \{[v_1, v_2]\}_V \), its linear map is computed as

\[
M \otimes P = \{[M v_1, M v_2]\}_V.
\]  

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

\[
M \otimes P = \{[\delta_1, \delta_2]\}_V = \{[-2, 6]\}_V.
\]

If we evaluate the result for \( \delta \) corresponding to the point \( p \), we obtain

\[
M \otimes P(\delta) = 2 \sum_{i=1}^{2} \delta_i v_i = 4 = Mp.
\]

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 the V-representation of a one-dimensional polytope \( P = \{[v_1, v_2]\}_V \subset \mathbb{R} \), its quadratic map is computed as

\[
sq(Q, P) = \{x^T Q x \mid x \in P\} = \begin{cases} \{0, Q \max(|v_1|, |v_2|)^2\}, & 0 \in P \\ \{Q \min(v_1, v_2)^2, Q \max(v_1, v_2)^2\}, & \text{otherwise} \end{cases}
\]  

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

\[
sq(Q, P) = \{[\delta_1, \delta_2]\}_V = \{[0, 18]\}_V.
\]

If we evaluate the computed quadratic map for \( \delta \) corresponding to the point \( p \), we obtain

\[
sq(Q, P)(\delta) = \sum_{i=1}^{2} \delta_i v_i = 13.5 \neq sq(Q, p) = 8,
\]

which is not identical to \( sq(Q, p) \). 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 \( A \) and a set \( S \subset \mathbb{R}^n \) parameterized by \( d \in D \subset \mathbb{R}^m \), we call the implementation of \( A \) dependency-preserving if

\[
\forall d \in D : A(S(d)) \subseteq A(S)(d).
\]  

A summary of the set operations that are dependency-preserving is shown in Table 1 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 \( S_1, S_2 \subset \mathbb{R}^n \) with \( S_1 \subset S_2 \), it holds trivially that all unary set operations \( A \) satisfy

\[
A(S_1) \subseteq A(S_2).
\]  

Furthermore, given sets \( S_1, S_2 \subset \mathbb{R}^n \) and \( S_3, S_4 \subset \mathbb{R}^m \) with \( S_1 \subset S_2 \) and \( S_3 \subset S_4 \), all binary set operations \( B \) used in this work satisfy

\[
B(S_1, S_3) \subseteq B(S_2, S_4).
\]  

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

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

**Proof.** Since \( A \) and \( B \) satisfy (6), it holds for two sets \( S_1, S_2 \subset \mathbb{R}^n \) with \( S_1 \subset S_2 \) that

\[
(A \circ B)(S_1) = A(B(S_1)) \subseteq A(B(S_2)) = (A \circ B)(S_2).
\]

The result of Lemma (8) 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 \( A \circ B \) of two dependency-preserving operations \( A \) and \( B \) is dependency-preserving:

\[
\forall d \in D : \quad (A \circ B)(\mathcal{S}((d))) \subseteq (A \circ B)((S)(d)).
\]

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

\[
\forall d \in D : \quad B(\mathcal{S}((d))) \subseteq B((S)(d)).
\]

Using (8) and (9), it holds that

\[
\forall d \in D : \quad A(\mathcal{S}((d))) \subseteq A((S)(d)).
\]

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

\[
\forall d \in D : \quad (A \circ B)((S)(d)) = (A \circ B)((S)(d)).
\]

which is identical to (5). \( \square \)

As we will show in Sec. 3, the reachability algorithm used in this work only applies dependency-preserving set operations, so that the reachability algorithm is dependency-preserving, too.

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 this requirement are polynomial zonotopes. In this work we use their sparse representation [25]:

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

\[
\mathcal{P}Z := \left\{ \sum_{i=1}^{h} \prod_{k=1}^{m} \alpha_{k}^{E_{k}(i,j)} G_{i,j} + \sum_{j=1}^{q} \beta_{j} G_{I}(i,j) \mid \alpha_{k}, \beta_{j} \in [-1,1] \right\},
\]

where \( f_{G,E}(\alpha) : \mathbb{R}^{m} \rightarrow \mathbb{R}^{n} \) is a multivariate polynomial function, and \( \alpha = \alpha_{1}, \ldots, \alpha_{m} \in \mathbb{R}^{m} \).

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 \( \sum_{j=1}^{q} \beta_{j} G_{I}(i,j) \) is called the independent part. We introduce the shorthand \( \mathcal{P}Z = \langle G, G_I, E \rangle_{\mathcal{P}Z} \).

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

Proposition 1. Given a polynomial zonotope \( \mathcal{P}Z = \langle G, G_I, E \rangle_{\mathcal{P}Z} \), \( \mathcal{P}Z \) can be equivalently represented without independent generators:

\[
\mathcal{P}Z = \langle G, G_I, [E] \rangle_{\mathcal{P}Z} \left\{ \begin{bmatrix} E & 0 \\ 0 & I_{q} \end{bmatrix} \right\}_{\mathcal{P}Z},
\]

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 \( \alpha_{m+1} = \beta_{1}, \ldots, \alpha_{p+q} = \beta_{q} \). \( \square \)

Despite the result of Prop. 1, the independent part of the polynomial zonotope is crucial for order reduction [25]. 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 can assume without loss of generality that the initial set has no independent generators. Consequently, it suffices to choose the parameter vector as \( d = \alpha = [\alpha_{1}, \ldots, \alpha_{m}]^{T} \), the parameter domain as \( D = \mathbb{Z}_{m} \), and the evaluation function as

\[
\mathcal{P}Z(\alpha) = f_{G,E}(\alpha) \oplus \left\{ \sum_{j=1}^{q} \beta_{j} G_{I}(i,j) \mid \beta_{j} \in [-1,1] \right\} \quad (13)
\]

Finding a parameterization for a point inside a polynomial zonotope can be computationally demanding. 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

\[
\mathcal{P}Z = \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix} \alpha_{1} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \alpha_{2} \mid \alpha_{1}, \alpha_{2} \in [-1,1] \right\}
\]

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

Next, we show how to 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 \( \mathcal{P}Z \) from Example 4. The point \( p \) and its transformations are depicted by the black dots. Black lines correspond to the points where \( \alpha_{1} = 0.5 = \text{const} \) and \( \alpha_{2} = 0.4 = \text{const} \).
**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 \( X_0 \subset \mathbb{R}^n \) and a set of possible input values \( U \subset \mathbb{R}^m \) is

\[
\mathcal{R}_{X_0}(t) = \{ \xi(t, x_0, u(\cdot)) | x_0 \in X_0, \forall t \in [0, t], u(\cdot) \in U \}. 
\]

The superscript \( e \) on \( \mathcal{R}_{X_0}(t) \) denotes the exact reachable set, which cannot be computed for general nonlinear systems. Therefore, we compute a tight over-approximation \( \mathcal{R}(t) \supseteq \mathcal{R}_{X_0}(t) \) with the operation **reach** defined by Alg. 1, which is taken from [25]. Alg. 1 is based on the abstraction of the nonlinear function \( f(\cdot) \) by a Taylor expansion of order \( \kappa \):

\[
\dot{x}(t) = f_0(z(t)) + \sum_{j=0}^{\kappa} \binom{\kappa}{j} (z(t) - z^*)^j f_j(\hat{z}) \oplus L(j, t),
\]

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 \( L(j, \cdot) \) 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 [25 Prop. 9], **reduce** denotes the order reduction of a polynomial zonotope (see [25 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 \( \mathcal{R}^{p, \lambda} [1 \text{ Eq. (9)}] \), **post** [1 \text{ Sec. 4.1}], **varInputs** [1 \text{ Sec. 4.2}], and **lagrangeRemainder** [1 \text{ 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. 3), as shown in Table 1. Exact addition and order reduction are not included in Table 1 since they are not applicable 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 \( \mathcal{R}(t) \) at points in time \( t \), since only these sets are propagated [1], while the reachable sets of time intervals \( \mathcal{R}(t) \) only fill the time gaps. Consequently, subsequent derivations focus on the reachable sets at points in time \( \mathcal{R}(t) \).

### 3.2 Reachable Subsets

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

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

\[
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= (1 - x_2^2)x_2 - x_1,
\end{align*}
\]

the initial set \( X_0 = \left\{ \begin{bmatrix} -1 \\ 1 \end{bmatrix} + 0.2 \alpha_1 + 0 \alpha_2 | \alpha_1, \alpha_2 \in [-1,1] \right\} \),

and the initial point \( x_0 = [-0.9, 1.08]^T \in X_0 \). The initial point \( x_0 \) can be parameterized with \( \sigma = [0.5, 0.4]^T \), so that \( x_0 = \Phi_{X_0}(\sigma) \). Computation of the reachable set for a time horizon of \( t_f = 1s \) with Alg. 2 yields

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

As visualized in Fig. 3 the exact reachable set \( \mathcal{R}_{X_0}^{(2)}(t_f) \) for the initial point \( x_0 \) can be enclosed by evaluating (17) for the parameter values \( \sigma = [0.5, 0.4]^T \) 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 \( \mathcal{R}^{(1)}(t), \mathcal{R}^{(2)}(t) \subset \mathbb{R}^n \) with \( \mathcal{R}^{(1)}(t) \subseteq \mathcal{R}^{(2)}(t) \) and the sets of applied abstraction errors \( \Psi^{(1)}(t), \Psi^{(2)}(t) \subset \mathbb{R}^n \) with \( \Psi^{(1)}(t) \subseteq \Psi^{(2)}(t) \),
it holds that
\[ \Psi^\Delta(\tau_s) \subseteq \Psi^\Lambda(\tau_s) \]
with
\[ \Psi^\Delta(\tau_s) = \text{abstrErr}(R^{(1)}(t_s), \overline{\Psi}(\tau_s)) \] \tag{18}
\[ \Psi^\Lambda(\tau_s) = \text{abstrErr}(R^{(2)}(t_s), \underline{\Psi}(\tau_s)) \]
so that \text{abstrErr} as defined by Alg. 2 satisfies (7).

**Proof.** As visualized in Fig. 4, Alg. 2 is compositioned by unary set operations that satisfy (7) and binary set operations that satisfy (7). It therefore holds according to Lemma 1 that \text{abstrErr} as defined by Alg. 2 satisfies (7).

\[ \forall \alpha \in \mathcal{I}_m : A\left(\overline{\Psi}(R(t_s))\right) \subseteq A\left(\overline{\Psi}(R(t_s))\right) \] \tag{20}
Since A is dependency-preserving, it holds according to (5) that
\[ \forall \alpha \in \mathcal{I}_m : A\left(\overline{\Psi}(R(t_s))\right) \subseteq A\left(\overline{\Psi}(R(t_s))\right) \] \tag{21}
Since \( R(t_s) \) is the set of abstraction errors resulting from the computation of \( \text{post}(R(t_s), 0, z^*_s) \) 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. We first consider the operations \( A, B, C \) and \( D \) 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 line (5) of Alg. 1 it is sufficient to consider only the values from the last iteration, since only these matter for subsequent computations. The operation \text{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
\[ C(R(t_s)) = A(R(t_s)) \cup B(R(t_s), \Psi(\tau_s)) \] \tag{22}
Further, since B returns a zonotope that does not contain the polynomial zonotope factors \( \alpha \),
\[ \forall \alpha \in \mathcal{I}_m : A(R(t_s)) \cup B(R(t_s), \Psi(\tau_s)) \]
\[ = A(R(t_s)) \cup B(R(t_s), \Psi(\tau_s)) \] \tag{23}
Consequently,
\[ \forall \alpha \in \mathcal{I}_m : C(R(t_s)) \]
which proves that operation C is dependency-preserving (see (5)).

**Lemma 4.** Given a reachable set \( R(t_s) \subseteq \mathbb{R}^n \) represented as a polynomial zonotope, it holds that
\[ \forall \alpha \in \mathcal{I}_m : \text{post}(R(t_s) \alpha, \Psi(\tau_s), z^*_s) \subseteq \text{post}(R(t_s), 0, z^*_s) \] \tag{19}
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 5.** Given an initial set \( \chi_0 \subseteq \mathbb{R}^n \) represented as a polynomial zonotope, it holds that
\[ \forall \alpha \in \mathcal{I}_m : \text{reach}(\chi_0 \alpha, \Psi(\tau_s), z^*_s) \subseteq \text{reach}(\chi_0, 0, z^*_s) \] \tag{24}
in Alg. 1, \( z_t^s \), \( s = 1, \ldots, N \) are the expansion points of the Taylor series, \( N = \lceil \frac{t_f}{\tau} \rceil \) 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

\[
\mathcal{R}(t_f) = \text{reach}(X_0, \Psi(\tau_s), z_t^s) = \text{post}_N(X_0, \Psi(\tau_s), z_t^s),
\]

where \( \text{post}_N \) denotes the \( N \)-times consecutive composition of the post operation

\[
\mathcal{R}(t_f) = \text{post}(\ldots \text{post}(X_0, \Psi(\tau_1), z_1^1) \ldots, \Psi(\tau_N), z_N^1)).
\]

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 \( \text{reach}(X_0, \Psi(\tau_s), z_t^s) = \text{post}_N(X_0, \Psi(\tau_s), z_t^s) \) is dependency-preserving.

Finally, we formulate the main result:

**Theorem 1.** Given an initial set \( X_0 \subset \mathbb{R}^n \) represented as a polynomial zonotope,

\[
\forall \alpha \in \mathcal{I}_m : \quad \mathcal{R}_{\alpha}^c(t_f) \subseteq \left[ \text{reach}(X_0, 0, z_t^s) \right](\alpha)
\]

with \( x_0 = \psi_0(\alpha) \),  

where \( z_t^s \) and \( \text{reach} \) are as in Lemma 3.

**Proof.** As shown in 1, Alg. 1 computes an over-approximation of the exact reachable set

\[
\mathcal{R}_{\alpha}^c(t_f) \subseteq \text{reach} \left( (X_0(\alpha), \Psi(\tau_s), z_t^s) \right),
\]

where \( \Psi(\tau_s), s = 1, \ldots, N \) are the sets of abstraction errors resulting from the computation of \( \text{reach}(X_0, 0, z_t^s) \) so that according to Lemma 5

\[
\forall \alpha \in \mathcal{I}_m : \quad \mathcal{R}_{\alpha}^c(t_f) \supseteq \left[ \text{reach}(X_0(\alpha), \Psi(\tau_s), z_t^s) \right] \subseteq \left[ \text{reach}(X_0, 0, z_t^s) \right](\alpha),
\]

which concludes the proof.

---

**3.3 Computational Complexity**

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

\[
f_{G,E}(\alpha) = \sum_{i=1}^{h} \left( \prod_{k=1}^{m} a_{E(k,i)}^{E(k,i)} \right) G_{(i)}.
\]

Computation of each \( P_i \) in (27) requires \( m \) exponentiations and \( m - 1 \) multiplications, and computation of \( P_i G_{(i)} \) requires \( n \) multiplications. Since there are \( h \) terms \( P_i G_{(i)} \) 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 \( c_m, c_h \in \mathbb{R}_{>0} \). The complexity for the extraction of a reachable subset is therefore \( \mathcal{O}(h(2m + n - 1) + n(h - 1)) = \mathcal{O}(n^2) \). Since the computation of the reachable set with the conservative polynomialization approach has complexity \( \mathcal{O}(n^3) \), our novel extraction of reachable subsets is computationally much more efficient.

---

**4. APPLICATIONS**

There are a lot of different applications for the result presented in this work. In this section, we introduce some of the possible applications and demonstrate the obtained efficiency by numerical examples. Due to space limitations, we only present the general concepts and omit implementation details. All computations are carried out 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 \( ax \leq b \). If the final reachable set \( \mathcal{R}(t_f) \) for the initial set \( X_0 \) does not fulfill the specification, it would be useful to know which initial states result in a violation. According to Theorem 1, the states inside the set \( S \subseteq X_0 \)}
defined as
\[ S = \bigcap_{\alpha \in B} \mathcal{X}(\alpha), \]
with \[ B = \left\{ \alpha \in \mathcal{B} \mid a \otimes \mathcal{R}(t_f)(\alpha) \leq b \right\}, \]
are guaranteed to fulfill the specification. The set of states \( S \subseteq X_0 \) that potentially result in a violation of the specification can consequently be computed as \( F = X_0 \setminus S \).

Example 6. We consider the reachability problem from Example 3 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 \( S \) and in orange the falsifying states \( F \) are visualized. Since the reachability problem from Example 3 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 3).

As a comparison, we also determined a falsifying trajectory using the simulated annealing algorithm of S-TALIRO [4]. Since the benchmark does not include uncertain inputs for optimizing over reachable sets: it is possible to achieve major speed-ups for optimizing over reachable sets:

Example 8. We consider the reachability analysis problem from Example 3 and the inequality constraint \( |x| \leq 6.4 \). As shown in Fig. 8 the final reachable set violates the constraint. We want to determine the modified initial set \( X_0 \subseteq X_0 \) 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 \( \bar{\alpha}_1, \overline{\alpha}_1, \bar{\alpha}_2, \overline{\alpha}_2 \) for the factors \( \alpha_1 \) and \( \alpha_2 \) as the variables for the optimization problem. With this, the initial set from Example 3 becomes
\[ X_0(\alpha_1, \overline{\alpha}_1, \bar{\alpha}_2, \overline{\alpha}_2) = \left\{ \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0.2 \\ 0 \end{bmatrix} \alpha_1 + \begin{bmatrix} 0 \\ 0.2 \end{bmatrix} \alpha_2 \right\}, \quad \alpha_1 \in [\underline{\alpha}_1, \bar{\alpha}_1], \quad \alpha_2 \in [\underline{\alpha}_2, \overline{\alpha}_2]. \]

where we denote the dependence of the initial set on the bounds \( \alpha_1, \alpha_1, \bar{\alpha}_2, \overline{\alpha}_2 \) by \( X_0(\alpha_1, \overline{\alpha}_1, \bar{\alpha}_2, \overline{\alpha}_2) \). The optimization
The optimization problem can be formulated as
\[
\max_{\alpha_1 \in [\alpha_1, \pi_1], \alpha_2 \in [\alpha_2, \pi_2]} \text{volume}(X_0(\alpha_1, \pi_1, \alpha_2, \pi_2))
\]
\[
\text{s.t. } [1, 2] \otimes \text{reach}(X_0(\alpha_1, \pi_1, \alpha_2, \pi_2), 0, z^*_0) \leq 6.4.
\]
(29)

The reachable set for the initial set and the reachable states introduced in this work can be equivalently formulated as
\[
\max_{\alpha_1 \in [\alpha_1, \pi_1], \alpha_2 \in [\alpha_2, \pi_2]} [1, 2] \otimes \text{reach}(R(t_f)) \leq 6.4.
\]
(30)

where \( R(t_f) \) is the final reachable set (see (27)). Inserting the numerical values for \( R(t_f) \) from (17) into (30) yields
\[
\max_{\alpha_1 \in [\alpha_1, \pi_1], \alpha_2 \in [\alpha_2, \pi_2]} 0.05\alpha_1 + 0.66\alpha_2 - 0.22\alpha_1^2 - 0.2\alpha_1\alpha_2 \leq 0.04.
\]
(31)

The volume of the initial set can be computed as
\[
\text{volume}(X_0(\alpha_1, \pi_1, \alpha_2, \pi_2)) = 0.04(\pi_1 - \alpha_1)(\pi_2 - \alpha_2),
\]
which simplifies the optimization problem (29) to
\[
\max_{\alpha_1, \pi_1, \alpha_2, \pi_2} 0.04(\pi_1 - \alpha_1)(\pi_2 - \alpha_2)
\]
\[
\text{s.t. } 0.05\alpha_1 + 0.66\alpha_2 - 0.22\alpha_1^2 - 0.2\alpha_1\alpha_2 \leq 0.04.
\]
(31)

The solution for the optimization problem (31) is visualized in Fig. 8. As shown in Table 3 solving the optimization problem (29) takes 221 seconds, but solving the simplified optimization problem (31) only takes 0.1 seconds.

4.3 Motion-Primitive Based Control

Let us consider a scenario where a maneuver automaton is used for the control of a system. The approaches from (27) construct 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 \( X_0 \) is computed offline. During online application our novel approach can directly extract the reachable set for a measured state \( \hat{x} \) from the offline-computed reachable set. Since the reachable set for \( \hat{x} \) is in general much smaller than the reachable set for \( X_0 \), planning with the reachable set for \( \hat{x} \) 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 (27) Sec. 6 with the measured velocity \( \hat{v} = 20.2\text{m/s} \), the measured orientation \( \hat{\phi} = 0.01\text{rad} \), and the measured position \( \hat{x}_1 = 0.1\text{m} \), \( \hat{x}_2 = 0.1\text{m} \). The reachable set for the initial set \( X_0 \) and reachable set extracted for then measured point \( \hat{x} = [\hat{x}_1, \hat{\phi}, \hat{x}_2]^T \) is visualized in Fig. 8. As shown in Table 3 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. 4.

5. CONCLUSION

We showed that the computation of the reachable set for nonlinear systems using the conservative polynomialization approach using polynomial zonotopes preserves the dependence between the reachable states and the initial states. Since this novel concept enables 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 concept results in significant speed-ups compared to the previous solutions of these problems. Our method for extracting reachable subsets has complexity \( O(n^2) \) and is therefore computational much more efficient than to compute the reachable subset from scratch, which has complexity \( O(n^3) \). To some extent this work provides a method for unifying reachability analysis and falsification.

6. REFERENCES

[1] M. Althoff. Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In Hybrid Systems: Computation and Control, pages 173–182, 2013.

[2] M. Althoff. An introduction to CORA 2015. In Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems, pages 120–151, 2015.

[3] M. Althoff and et al. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In Proc. of the 47th IEEE Conference on Decision and Control, pages 4042–4048, 2008.

[4] Y. Annunparalleled and et al. S-TaLiRo: A tool for temporal logic falsification for hybrid systems. In Proc. of Tools and Algorithms for the Construction and Analysis of Systems, pages 254–257, 2011.

[5] E. Asarin and et al. Hybridization methods for the analysis of nonlinear systems. Acta Informatica, 43:451–476, 2007.

[6] S. Bak and P. S. Duggirala. HyLAA: A tool for computing simulation-equivalent reachability for linear systems. In Proc. of the 20th International Conference on Hybrid Systems: Computation and Control, pages 173–178, 2017.

[7] S. Bak and P. S. Duggirala. Simulation-equivalent reachability of large linear systems with inputs. In Proc. of International Conference on Computer Aided Verification, pages 401–420, 2017.
[8] L. Benvenuti and et al. Assume-guarantee verification of nonlinear hybrid systems with ARIADNE. International Journal of Robust and Nonlinear Control, 24:699–724, 2014.

[9] S. Bogomolov and et al. Falsification of hybrid systems using symbolic reachability and trajectory splicing. In Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, pages 1–10. ACM, 2019.

[10] S. Bogomolov and et al. Juliareach: a toolbox for set-based reachability. In Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, pages 39–44. ACM, 2019.

[11] X. Chen and et al. Taylor model flowpipe construction for non-linear hybrid systems. In Proc. of the 33rd IEEE Real-Time Systems Symposium, 2012.

[12] X. Chen and et al. Flow*: An analyzer for non-linear hybrid systems. In Proc. of Computer-Aided Verification, LNCS 8044, pages 258–263. Springer, 2013.

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

[14] T. Dang and et al. Accurate hybridization of nonlinear systems. In Hybrid Systems: Computation and Control, pages 11–19, 2010.

[15] P. S. Duggirala and et al. C2e2: A verification tool for stateflow models. In International Conference on Tools and Algorithms for the Construction and Analysis of Systems, pages 68–82. Springer, 2015.

[16] P. S. Duggirala and M. Viswanathan. Parsimonious, simulation based verification of linear systems. In Proc. of International Conference on Computer Aided Verification, pages 477–494, 2016.

[17] G. Frehse and et al. SpaceEx: Scalable verification of hybrid systems. In Proc. of the 23rd International Conference on Computer Aided Verification, LNCS 6806, pages 379–395. Springer, 2011.

[18] R. Ghosh and et al. Automated symbolic reachability analysis; with application to delta-notch signaling automata. In International Workshop on Hybrid Systems: Computation and Control, pages 233–248. Springer, 2003.

[19] A. Girard and et al. Efficient computation of reachable sets of linear time-invariant systems with inputs. In Hybrid Systems: Computation and Control, LNCS 3927, pages 257–271. Springer, 2006.

[20] A. Girard and C. Le Guernic. Efficient reachability analysis for linear systems using support functions. In Proc. of the 17th IFAC World Congress, pages 8966–8971, 2008.

[21] E. Goubault and S. Putot. Inner and outer reachability for the verification of control systems. In HSCC, pages 11–22, 2019.

[22] J. Hoeftsmens and et al. Scientific Computing, Validated Numerics, Interval Methods, chapter Verified High-Order Integration of DAEs and Higher-Order ODEs, pages 281–292. Springer, 2001.

[23] F. Immler. Tool presentation: Isabelle/HOL for reachability analysis of continuous systems. In Proc. of the 2nd Workshop on Applied Verification for Continuous and Hybrid Systems., pages 180–187, 2015.

[24] F. Immler and et al. ARCH-COMP19 category report: Continuous and hybrid systems with nonlinear dynamics. In ARCH19. 6th International Workshop on Applied Verification of Continuous and Hybrid Systems, 2019.

[25] N. Kochdumper and M. Althoff. Sparse polynomial zonotopes: A novel set representation for reachability analysis. arXiv preprint arXiv:1901.01780, 2019.

[26] H. Kong and et al. Safety verification of nonlinear hybrid systems based on invariant clusters. In Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control, pages 163–172. ACM, 2017.

[27] A. B. Kurzhanski and P. Varaiya. Ellipsoidal techniques for reachability analysis. In Hybrid Systems: Computation and Control, LNCS 1790, pages 202–214. Springer, 2000.

[28] M. Li and et al. Safe over- and under-approximation of reachable sets for autonomous dynamical systems. In International Conference on Formal Modeling and Analysis of Timed Systems, pages 252–270. Springer, 2018.

[29] J. Liu and et al. Computing semi-algebraic invariants for polynomial dynamical systems. In Proceedings of the ninth ACM international conference on Embedded software, pages 97–106. ACM, 2011.

[30] K. Makino and M. Berz. Rigorous integration of flows and ODEs using Taylor models. In Proc. of Symbolic-Numeric Computation, pages 79–84, 2009.

[31] N. Matringe and et al. Generating invariants for non-linear hybrid systems by linear algebraic methods. In International Static Analysis Symposium, pages 373–389. Springer, 2010.

[32] I. M. Mitchell and et al. A time-dependent Hamilton–Jacobi formulation of reachable sets for continuous dynamic games. IEEE Transactions on Automatic Control, 50:947–957, 2005.

[33] P. Prabhakar and M. Viswanathan. A dynamic algorithm for approximate flow computations. In Hybrid Systems: Computation and Control, pages 133–142, 2011.

[34] R. Ray and A. Gurung. Parallel state space exploration of linear systems with inputs using xspeed. In Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control, pages 285–286. ACM, 2015.

[35] J. A. D. Sandretto and A. Chapotut. Dynex: a differential constraint library for studying dynamical systems. 2016.

[36] S. Schupp and et al. Hypro: A C++ library of state set representations for hybrid systems reachability analysis. In NASA Formal Methods Symposium, pages 288–294. Springer, 2017.

[37] B. Schürmann and M. Althoff. Closed-form expressions of convex combinations for controller design. In Proc. of American Control Conference, pages 2795–2801, 2016.

[38] O. Stursberg and B. H. Krogh. Efficient representation and computation of reachable sets for hybrid systems. In Hybrid Systems: Computation and Control, LNCS 2623, pages 482–497. Springer, 2003.