Scalable Verification of Cyber-Physical Systems

Kostiantyn Potomkin

School of Computing
Newcastle University

This dissertation is submitted for the degree of
Doctor of Philosophy

January 2024

Abstract

In this thesis, several challenges in the verification of cyber-physical systems are considered. First, verification methods are generally not scalable, i.e. they suffer when performed on high-dimensional systems and require sometimes unreasonable computational time. Verification of hybrid systems, in addition, involves the computation of complex operations. Next, nonlinear systems are harder to reason as opposed to linear systems. Also, there is a pressing need to reason about safety in a limited time or even instantly. Thus, the following solutions are presented to address scalability and performance issues in verification.

To begin with, we leverage the structure of systems when safety properties are defined only in low dimensions. In particular, an algorithm is proposed to exploit decomposition in the reachability analysis of linear hybrid systems. It allows to verify systems with up to thousands of state variables without additional approximation error for linear hybrid systems with a low number of constraints.

Next, a data-driven framework to handle nonlinear, even black-box, systems is provided. It is based on Koopman operator and Fourier Features, well-known approximation techniques which, in some cases, could exactly represent the original system. Then, two options are offered to handle nonlinear initial sets created by such linearization: (i) utilize interval arithmetics along with refinement steps and calls to the SMT solver and (ii) combine polynomial zonotopes with efficient set operations to obtain a tight approximation for nonlinear reachable sets. This enables an extremely fast verification in comparison with state-of-the-art tools for nonlinear systems as it shown on several nonlinear system benchmarks.

Finally, we developed an algorithm which verifies the system on-the-fly. It generates reachable regions by simulations, which are enclosed by barrier certificates to provide formal guarantees. These barrier certificates are produced by neural networks and verified by SMT solvers. Although the algorithm is currently restricted by scalability of FOSSIL, it already demonstrate promising results in verification both for nonlinear models and in online settings.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Motivation

Cyber-physical systems [33] are usually defined as systems where software components interact with the environment. One of the core features of such systems is that they can independently adjust their dynamics and behaviour. Examples of such systems are self-flying drones, autonomous cars, traffic and transport management systems, etc. Cyber-physical systems play a crucial role in safety-critical situations, especially where human control is either impossible or might be dangerous. These areas include military or defence systems [6], nuclear stations [5], etc. Cyber-physical systems become more complex and cover more functunality [145]. Unreliable systems might lead to dangerous situations [163]. Cyber-physical systems are already widely used in daily life and current predictions [110] say that the role of autonomous systems will increase in future and more of such devices will be around.

System safety is usually referred to as the management of the risks or strategy of identifying potential issues, hazards and the possible ways of handling them. Fault tolerance and fault avoidance are some of the ways to address safety concerns, while in some cases one can debate to allow potentially 'dangerous' behaviour of the system. On the other hand, faulty behavior could cost life or leads to very expensive losses.

Thus, it is of utmost importance to provide safety guarantees, i.e. that the system with the provided initial state cannot reach any specified set of unwanted or unsafe states. These states could include dangerous states of the system which are very expensive to recover from and states where system are not expected to operate so its behavior is unpredictable. The safety of cyber-physical systems has attracted much attention recently both from academical [80] and industrial worlds [50]. This corresponds to verification process in a technical standard ISO/IEC15288, so the system meets requirements and specifications and that it fulfills its intended purpose.

Model-based design [23, 121] has attracted considerable interest as an approach to design novel systems. This approach comprises of three main steps. First, a formal mathematical model of the desired system is designed, along with the elements of its environment. Secondly, a designer utilizes one of the state-of-the-art approaches to verify

the constructed system. For this purpose, safety properties, i.e. constraints which must be satisfied to deem a system safe. If the system is unsafe, then a designer redefines a model or refines the settings and runs the verification again. Lastly, once the model is verified, it is ready for deployment out of the model-based design framework and being actually built. Since we do not consider the actual implementation of the system, in the following, we refer to a formal model of a system just as a system for simplicity. Model-based design lies on the edge between mathematics, engineering, and computer science [113], where each step is a ground for further deep research for a better design of complex embedded systems.

Hybrid systems [24] is a commonly-used formalism to model cyber-physical systems, which exchibit mixed discrete-continuous dynamics. A simple example of a hybrid system is a thermostat model with two nodes: heating and cooling (see Figure 1.1). These nodes contain continuous dynamics, *i.e.* differential equations describing temperature dynamics. Nodes are linked with each other via transitions. In this case, the transitions define constraints on the maximal and minimal temperatures, respectively. Thus, if the temperature reaches some threshold, a jump to another mode occurs.

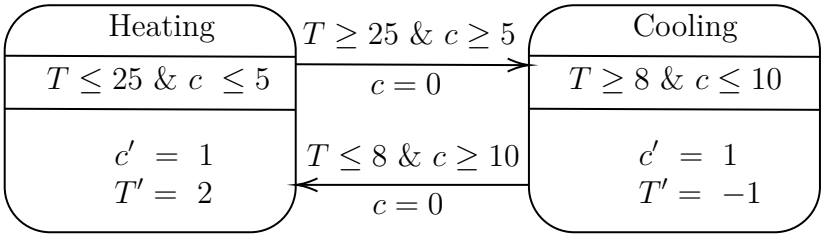


Fig. 1.1. Example of thermostat hybrid system. There are two nodes heating and cooling. Each has its own dynamics and invariants, constraints on the state variables at the corresponding location. Transitions represent discrete jumps between locations. Each transition contains guards, *i.e.* constraints which once satisfied enable jumps. In addition, there is an assignment, where discrete dynamics is defined.

Interestingly, there are systems whose dynamics is unknown or undefined upon modelling. Such systems are called black-box systems [158]. Since their dynamics can be obtained by data (e.g., simulations), these dynamical systems prevail especially in data-driven testing environments.

1.2 Verification of cyber-physical systems

The main idea of safety verification is to provide guarantees that the system under consideration with the provided initial state cannot violate a given safety property. There are several state-of-the-art approaches to verify the safety of the system. In this thesis, we combine the provided approaches in the developed algorithms.

Commercial companies usually utilize simulations to verify the safety of their systems [135]. Monte Carlo simulations [130] is a common way to test the designed models. ANSYS [138], Simulink/Stateflow MATLAB libraries [173] and COMSOL [93] are the main tools to run such simulations. However, the drawback of this approach is an inability

to cover all possible trajectories, especially in systems with large initial sets or high uncertainty. Thus, there is a chance that some trajectory or even trajectories are not covered and the system is actually unsafe (see Figure 1.2).

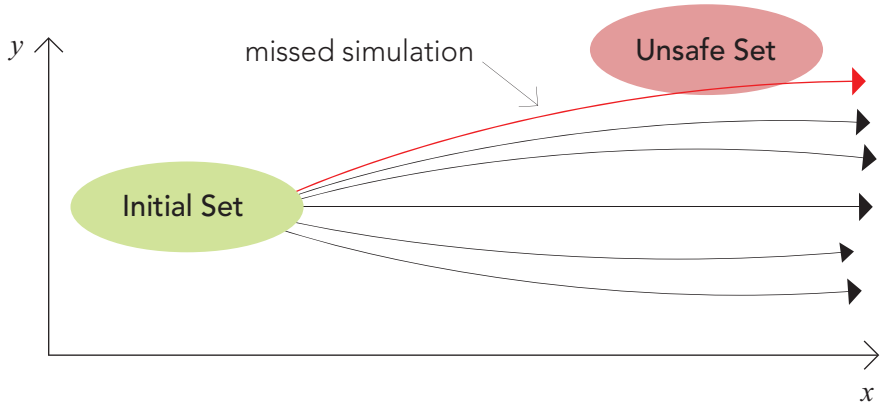


Fig. 1.2. Illustration of the missed trajectory in simulation approach. Green region is an initial set, red figure corresponds to the set of forbidden states and black curves are simulation trajectories from the initial region. The red curve represents a trajectory which intersects the set of bad states but is missed by a numerical simulation.

Simulation-Guided Analysis is a different approach which aims to tackle the highlighted issue of missed trajectories. The idea is to obtain simulation trajectories, and bloat them in a way that for each state from the set of initial states there is a trajectory in a bloated simulation. The main challenge in this technique is the calculation of a discrepancy function. A discrepancy function provides a mechanism for bounding the distance between adjacent trajectories as a function of the distance between the initial states for the trajectories [74]. [77] presents a systematic approach to compute such a function. A practical use of such algorithms is restricted, since most of the heuristics to compute discrepancy functions address only very specific models or problems [198].

While many methods strive to verify that the system is safe, another approach, namely falsification, provides evidence that the system is unsafe and is a great ground for test cases. Hylaa (HYbrid Linear Automata Analyzer) [34] is an example of such a verification tool which can scale to systems with hundreds of thousands of state variables. However, the main drawback of this approach is if counterexample, i.e. a trace leading to a set of unsafe states, is not found, then the result is inconclusive and one cannot reason about the safety of the system.

Formal verification techniques [23, 103] received more consideration of scientific community and emerged to address the issues of simulation-based methods [69, 87, 78].

Safety analysis can be done using flowpipe based reachability analysis [101, 137]. It aims at identifying states reachable by a system within a given time horizon (see Figure 1.3). Reachability analysis performs verification of hybrid systems by exploring all possible reachable states from the set of initial states and then checks the intersection with the unsafe region. If the intersection is not empty, then the system is unsafe [152]. Complexity of computing discrete jumps and interesections in particular, restrict current algorithms

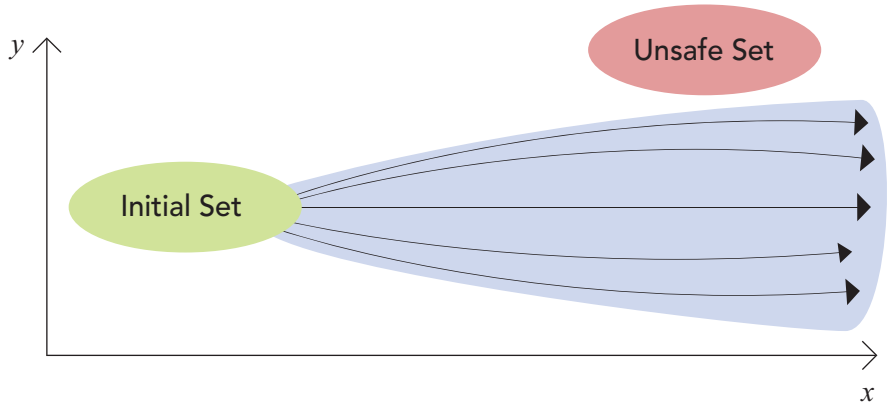


Fig. 1.3. Illustration of reachability analysis. Green region is an initial set, red region corresponds to the set of unsafe states and black curves are simulation trajectories. Blue set is an exact reachable set.

to support systems with up to dozens of state variables and, in some cases, hunderds. However, additional dimensions could provide system designers a venue for more descriptive models and take into account more environment dynamics.

Reachability analysis is implemented in several tools. These tools compute reachable state space by efficiently manipulating region space represented in terms of support functions [95], zonotopes [128] and Taylor models [64]. SpaceEx [84] is the most mature tool for linear hybrid systems with support for up to hundreds of state variables. CORA [19] is a toolbox which combines algorithms for verification of linear and nonlinear systems. In particular, for nonlinear systems techniques based on linearization and polynomialization are proposed. Flow* [65] is a tool for reachability analysis of hybrid nonlinear system based on Taylor model approximation. Although these tools are mature and demonstrate the best computational time in the field according to results from ARCH competition [21], it still takes more time to perform reachability on nonlinear systems in comparison with computing reachable sets for similar size linear models by several orders of magnitude. JuliaReach [53] emerged as a promising toolbox to handle both linear and nonlinear systems. The toolbox contains its own new efficient techniques, such as decompositional reachability for purely continous linear systems, and several algorithms from other tools, such as SpaceEx and Flow*, reimplemented. Currently, the toolbox perform similar to the original tools [112, 20] for nonlinear systems and for hybrid systems, however, outperform competitors in verification of purely continous linear systems. A wholly symbolic approach to hybrid systems verification (and more) is presented in [159], although it is hard to mechanise and it is based on quantifier elimination algorithms that have exceedingly high complexity. Given a linear switching system with a hyperplanar state-space partition, one approach computes ellipsoidal approximations of the reach set on the partition borders, without computing the full reach set [104].

A different approach to perform verification of hybrid systems relies on using constraint solvers [68] and checking whether proposed logical formulas are satisfiable. Reachability

properties [168, 169] and barrier certificates [192] can be verified via Satisfiability Modulo Theory (SMT) [90, 72, 46, 48] solvers.SMT solvers answer the question of whether a hybrid system runs into a set of unsafe states, but does not compute an actual reachable set. These techniques can be informative when a counterexample, i.e. a trace which leads to unsafe set, is required. This class of methods is usually divided into unbounded and bounded SMT algorithms. The former theoretically can always terminate, but might produce an unknown result in some cases. The latter produces only explicit results, however, given the unbounded problem is undecidable, algorithms might time out.

Additionally, barrier certificates have been introduced [161] to reason about the safety of the system without computing actual reachable sets explicitly. It is a function of the system state whose zero-level set separates the "unsafe" region from the system trajectories that start from a given initial set (see Figure 1.4). The existence of a barrier certificate entails that the system is safe. Multiple techniques have been introduced to compute barrier certificates. They mainly consist of (i) refinement loops where barrier certificate candidates are generated and (ii) verification using SMT solvers.

Approaches involving sum of squares programming are successful in computing barrier certificates for polynomial vector fields [157, 160]. Notably, these approaches rely on convex optimisation and hence can often result in solutions which are numerically sensitive and unsound. Numerically-robust inductive rules for both stability and safety of general dynamical systems have been presented in [91]. A recent approach aims at synthesising weaker barrier certificates using bilinear matrix inequalities [193], while [25] aims at including performance requirements in barrier certificates.

Early formal approaches for synthesis of barrier certificates include [116], which uses linear programs guided by simulation traces to generate candidate Lyapunov functions and barrier certificates. These candidates are then certified by an SMT-solver to guarantee the validity of the found solution, or provide a counter-example where the candidate is invalid. Usage of counter-examples and, more specifically, counter-example guided inductive synthesis (CEGIS) is a powerful and common approach for synthesising barrier certificates [9, 8, 107, 167]. Using neural networks to synthesise barrier certificates is treated in, *e.g.,* [114, 200–202]. However, these approaches are either unsound [114], or limited to the usage of ReLU-based activation functions within the network. The paper [183] is restricted only to support vector machines for synthesising barrier certificates for a specific class of systems (affine control robotic systems).

As it stated above there are multiple challenges and issues in applying formal verification to the real-world systems [177, 63, 22]. Despite recent advances, systems described by nonlinear ordinary differential equations are still hard to analyze, control, and verify. Hybrid dynamics creates additional complexity for formal techniques. Although it is possible to verify linear hybrid systems with up to 200 state variables the increased complexity and size of the models make the verification even harder. Thus, formal verification methods are not ready to be deployed on real-time systems for verification, while scalability issues prevent fully using these techniques in model-based design. There is a growing need for frameworks being capable to verify real-life systems, particularly

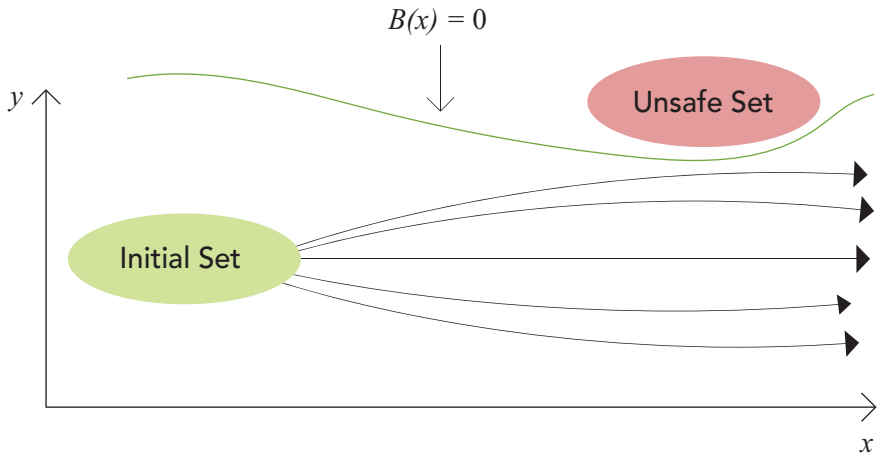


Fig. 1.4. Verification using a barrier certificate (green curve), which separates forbidden states (red) and initial set (green) with possible trajectories (black) from it. Green region is an initial set and red figure corresponds to the set of forbidden states.

high-dimensional. Verification on-the-fly instantly or in a short time is yet another relevant venue for formal techniques as more systems perform in autonomous scenarios.

1.3 Contribution

In this thesis, we present novel techniques for online and offline verification of cyber-physical systems. All the presented techniques and algorithms are compared with state-of-the-art tools on commonly used benchmarks. In the following we summarize the contribution presented in this thesis.

Contribution 1. Decompositional analysis of linear high-dimensional hybrid systems. Verification of systems with mixed discrete-continous dynamics is time consuming as it requires computing expensive intersection operations. Verification time grows even more rapidly for high-dimensional systems. There are approaches [52] to efficiently reason about purely continuous high-dimensional systems. To extend these methods for systems with mixed discrete-continous dynamics, we leveraged the specific structure of hybrid systems. In particular, we developed an algorithm for verification of high-dimensional hybrid systems with the extensive use of projections and decomposition. The advantages of the presented approach are:

(i) It allows us to compute reachable images of all state variables only in the specific time intervals, which are relevant for discrete jumps and only in their low-dimensional sub-spaces.

(ii) Any other computations are made only for the required state variables, thus, avoiding any unnecessary expensive computations.

(iii) The computational time for reachability analysis of linear hybrid systems is considerably reduced, in many cases without introducing additional approximation error, and systems with thousands of dimensions are supported.

Contribution 2. Verification of nonlinear systems via model transformation. Nonlinear systems is a major challenge in the verification community [63]. Many techniques have been presented to verify nonlinear systems [17, 62, 61, 19]. Most of these techniques rely on the approximation of nonlinear systems and representing them in simpler dynamics, e.g., linear. Typically, these verification algorithms suffer from complex and high-dimensional systems or necessity to consider long time intervals for reliable conclusions. Such abstraction techniques require either a coarse overappoximation or a very small time step to reason about the original system which eventually leads to excessive verification times. In this thesis, we propose to utilize Koopman operator [127] and random Fourier features [73] for approximation purposes. These approaches are widely used in different fields to analyse nonlinear systems. Moreover, extended dynamic mode decomposition [197] is applied to traces of the original system to obtain the best approximation. Given that simulation trajectories can be obtained for black-box systems, a model of the original system is not necessarily required. However, generated approximations cannot automatically be applied in reachability analysis as such transformations initialize new state variables where initial values are connected to the original state variables via nonlinear functions. For this purpose, we present two options to verify linearized systems with nonlinear initial sets:

(i) interval approximation with the refinement algorithm and calls to a SMT solver to verify the linearized system.

(ii) polynomial zonotopes approximation, which provides a tighter approximation of the non-convex initial set. This set representation can be efficiently propagated forward to compute the reachable set.

This algorithm computes reachable sets for the linearized system with nonlinear observables fast and outperforms the nonlinear solver. However, safe guarantees are provided only on the linearized system, while the approximation error currently can be calculated just statistically, which, in the provided evaluations, is relatively small and allows to verify safety properties.

Contribution 3. Online verification of autonomous systems. Verification of real-time systems has received much attention recently with some frameworks developed to tackle this problem [14, 186]. However, these approaches have their restrictions, for example, they work only with systems without uncertainty [31] or unsound [103]. Similarly, commercial companies deliver their autonomous systems and claim they are safe, although no formal guarantees on their safety [187] are provided. Companies utilize machine learning techniques to generate "safe control" while giving room for unsafe cases. In this work, various verification techniques are combined to provide safety guarantees. In particular,

we verify such systems by utilizing simulations of ODEs and neural networks along with formal verification by generating barrier certificates. We explore how barrier certificate formulations can be adapted for real-time reachable set computations. The key part of the framework is a neural network which is trained to generate barrier functions out of initial and unsafe sets. These barrier functions are then verified with the SMT solvers on-the-fly. The main features of the developed framework are:

(i) We verify both linear and nonlinear systems.

(ii) We obtain valid overapproximations of the reachable sets without explicitly solving the system dynamics.

(iii) The output of the algorithm provides guarantees on the safety of the autonomous system.

iiii The evaluation results of generated barrier certificate candidates from MetaNN demonstrate that the technique can be used for verification on-the-fly.

The above three contributions have been previously published in [54, 44, 43] and are in the process of preparing for submission to publication [7]. All the co-authors were notified and provided consent that these contributions as well as publications will be used as a basis for this thesis. The author of the thesis is the main developer of all the proposed algorithms, as well as responsible for all the evaluations and their interpretation. Additional proofs, evaluation results and content were added where possible for a more detailed and full description of the proposed approaches and challenges they address.

1.4 Structure

The thesis is structured as follows:

1. Chapter 2 provides a more detailed discussion of verification techniques and related works.

2. The discussion of the proposed decompositional approach is presented in Chapter 3.

3. We discuss the linearization techniques and the proposed reachability algorithms to handle nonlinear initial sets in Chapter 4.

4. Chapter 5 contains a description of how neural network can be used to produce safe barrier certificates and its use in a real-time case study of autonomous car. The chapter is based on the results published in [37] and [44].

5. We summarize the results presented in this thesis and outline future work in Chapter 6.

Chapter 2

Background

Verifying safe behaviour of cyber-physical systems is an important, timely topic and is receiving much attention from the research community [80]. The number of verification tools grow recently and friendly competitions such as International Competition on Verifying Continuous and Hybrid Systems(ARCH) [21] become more popular. These events gather participants with the wide range of tools, which can reason about linear and nonlinear systems, systems with continous and hybrid dynamics, etc.

In this chapter, we introduce notation and related verification techniques to use throughout the thesis. The structure of the chapter is the following:

1. Section 2.1 introduces notation to use throughout the thesis.
2. Section 2.2 defines the safety problem.
3. Section 2.3 provides an overview of reachability analysis and different ways of computing and storing reach sets.
4. Section 2.4 describes a verification technique based on using barrier certificates.

2.1 Notations

The real numbers are denoted by $\mathbb{R}$. The origin in $\mathbb{R}^n$ is written $\mathbf{0}_n$. Sets are denoted by calligraphic letters, matrices by uppercase letters, vectors by lowercase letters, and lists by bold uppercase letters. Let $\mathcal{C}_n \subseteq 2^{\mathbb{R}^n}$ be the set of n-dimensional compact convex sets. 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. Given a discrete set of positive integer indices $\mathcal{H} = \{h_1, \ldots, h_w\}$ with $1 \leq h_i \leq m \ \forall i \in \{1, \ldots, w\}$, $A_{(\cdot,\mathcal{H})}$ is used for $[A_{(\cdot,h_1)} \ \ldots \ A_{(\cdot,h_w)}]$, where $[C \ D]$ denotes the concatenation of two matrices C and D. The symbols $\mathbf{0}$ and $\mathbf{1}$ represent matrices of zeros and ones of proper dimension, the empty matrix is denoted by $[\,]$, and $I_n \in \mathbb{R}^{n \times n}$ is the identity matrix. Given an ordered list $\mathbf{L} = (l_1, \ldots, l_n)$, $\mathbf{L}_{(i)} = l_i$ refers to the i-th entry and $|\mathbf{L}| = n$ denotes the number of elements in the list. Moreover, the concatenation of

two lists $\mathbf{L}_1$ and $\mathbf{L}_2$ is denoted by $(\mathbf{L}_1, \mathbf{L}_2)$. We further introduce an n-dimensional interval as $\mathcal{I} = [l, u],\ \forall i\ l_{(i)} \leq u_{(i)},\ l, u \in \mathbb{R}^n$.

Given two vectors $x, y \in \mathbb{R}^n$, their dot product is $\langle x, y \rangle := \sum_{i=1}^{n} x_i \cdot y_i$. For $p \geq 1$, the p-norm of a matrix $A \in \mathbb{R}^{n \times n}$ is denoted $\|A\|_p$. The diameter of a set $\mathcal{X} \subseteq \mathbb{R}^n$ is $\Delta_p(\mathcal{X}) := \sup_{x,y \in \mathcal{X}} \|x - y\|_p$. The n-dimensional unit ball of the p-norm is $\mathcal{B}_p^n := \{x \in \mathbb{R}^n \mid 1 \geq \|x\|_p\}$.

Given sets $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{Y} \subseteq \mathbb{R}^m$, scalar $\lambda \in \mathbb{R}$, matrix $A \in \mathbb{R}^{n \times n}$, and vector $b \in \mathbb{R}^n$, we use the following operations on sets: scaling $\lambda\mathcal{X} := \{\lambda x \mid x \in \mathcal{X}\}$, linear map $A\mathcal{X} := \{Ax \mid x \in \mathcal{X}\}$, Minkowski sum $\mathcal{X} \oplus \mathcal{Y} := \{x + y \mid x \in \mathcal{X} \text{ and } y \in \mathcal{Y}\}$ (if $n = m$), affine map $(A, b) \odot \mathcal{X} := A\mathcal{X} \oplus \{b\}$, Cartesian product $\mathcal{X} \times \mathcal{Y} := \{(x, y) \mid x \in \mathcal{X}, y \in \mathcal{Y}\}$, intersection $\mathcal{X} \cap \mathcal{Y} := \{z \mid z \in \mathcal{X}, z \in \mathcal{Y}\}$ (if $n = m$), and convex hull $\mathrm{CH}(\mathcal{X}) := \{\lambda \cdot x + (1 - \lambda) \cdot y \mid x, y \in \mathcal{X}, 0 \leq \lambda \leq 1\}$.

Given two sets $\mathcal{X}, \mathcal{Y} \subseteq \mathbb{R}^n$, the Hausdorff distance is

$$d_H^p\big(\mathcal{X}, \mathcal{Y}\big) := \inf_{\varepsilon \in \mathbb{R}} \left\{\mathcal{Y} \subseteq \mathcal{X} \oplus \varepsilon\mathcal{B}_p^n \text{ and } \mathcal{X} \subseteq \mathcal{Y} \oplus \varepsilon\mathcal{B}_p^n\right\}.$$

2.2 Safety

First of all, we define the safety problem.

Definition 1. ***Safe system.*** *Given an initial set $\mathcal{X}_0 \subseteq \mathcal{X}$ and an unsafe set $\mathcal{X}_u \subseteq \mathcal{X}$, the system is called* safe *if all trajectories starting from any initial state $x_0 \in \mathcal{X}_0$ do not enter the unsafe set $\mathcal{X}_u$.*

In addition, we formally define hybrid systems, a popular formalism to model systems with mixed discrete-continuous dynamics.

Definition 2. ***Hybrid System*** *[12, 102, 133, 24, 109]. An n-dimensional hybrid system is a tuple, which is defined as follows:*

$$\mathcal{H} = (\delta, Loc, Flow, Inv, Grd, Asgn) \tag{2.1}$$

where:
- $\delta = \{x_1, ..., x_n\}$ is a finite set of variables;
- $Loc = \{l_1, ..., l_d\}$ is a finite set of locations;
- Flow is a mapping of the locations to ordinary differential equations of the form:

$$\frac{d}{dt}x(t) = f(x(t)), \quad t \geq 0, \tag{2.2}$$

with system variables δ and vector field $f : \mathcal{X} \to \mathbb{R}^n$. The differential equation (2.2) has a unique, continuous solution from any initial state $x(0) = x_0 \in \mathcal{X}$ under standard Lipschitz continuity of the vector field $f(\cdot)$.
- $Inv : Loc \to 2^{\mathbb{R}^n}$ is an invariant. It restricts the values that x can possibly take while remaining in the location;

- *$Grd : Loc \times Loc \to 2^{\mathbb{R}^n}$ associates to each transition a guard which should be satisfied for a transition to be taken;*
- *$Asgn : Loc \times Loc \to \mathbb{R}^{n\times n} \times \mathbb{R}^n$ maps each transition to an assignment which takes effect when a transition is taken.*

If $Grd(\ell, \ell') \neq \emptyset$, we call (ℓ, ℓ') a (discrete) transition.

In the following, we provide a discussion of the verification techniques which are fundamental for this thesis. In addition, a discussion of recent developments and works in the addressing related challenges is provided.

2.3 Reachability analysis

Reachability analysis works by iteratively applying continuous and discrete post operators to compute states reachable according to continuous and discrete dynamics, respectively. In the following we discuss different building blocks of reachability analysis algorithms.

2.3.1 Overview of reachability algorithms

Definition 3. ***Continuous-Post operator***. *Continuous-post operator, $Post_C$, computes the set of reachable states for a set of initial states $\mathcal{X}_0 \in \mathcal{C}_n$ and all input signals u over $\mathcal{U}$:*

$$Post_C(\mathcal{U}, \mathcal{X}_0) := \{\xi_{x_0,u}(t) \mid t \geq 0, x_0 \in \mathcal{X}_0, u(s) \in \mathcal{U} \text{ for all } s\}.$$

where $\xi_{x_0,u}(t)$ is the unique solution to eq. 2.2.

This can be used to define *continuous-post operator* for specific types of systems, e.g.,linear time-invariant systems. An n-dimensional linear time-invariant (*LTI system* $(A, B, \mathcal{U})$), with matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, and input domain $\mathcal{U} \in \mathcal{C}_m$, is a system of ODEs of the form

$$\dot{x}(t) = Ax(t) + Bu(t), \quad u(t) \in \mathcal{U}. \tag{2.3}$$

We denote the set of all n-dimensional LTI systems by $\mathcal{L}_n$. From now on, a vector $x \in \mathbb{R}^n$ is also called a (continuous) state. Given an initial state $x_0 \in \mathbb{R}^n$ and an input signal u such that $u(t) \in \mathcal{U}$ for all t, a *trajectory* of (2.3) is the unique solution $\xi_{x_0,u} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ with

$$\xi_{x_0,u}(t) = e^{At}x_0 + \int_0^t e^{A(t-s)}Bu(s)\, ds.$$

The *continuous-post operator* for a given LTI system $(A, B, \mathcal{U})$ and a set $\mathcal{X}_0 \in \mathcal{C}_n$ of initial states is defined as follows:

$$Post_C((A, B, \mathcal{U}), \mathcal{X}_0) := \{\xi_{x_0,u}(t) \mid t \geq 0, x_0 \in \mathcal{X}_0, u(s) \in \mathcal{U} \text{ for all } s\}.$$

A (symbolic) state of $\mathcal{H}$ is a pair $(\ell, \mathcal{X}) \in Loc \times 2^{\mathbb{R}^n}$.

Definition 4. ***Discrete-Post operator.*** *The discrete-post operator, $Post_D$, maps a symbolic state to a set of symbolic states by means of discrete transitions:*

$$Post_D(\ell, \mathcal{X}) := \bigcup_{\ell' \in Loc} \{(\ell', Asgn(\ell, \ell') \odot (\mathcal{X} \cap Inv(\ell) \cap Grd(\ell, \ell')) \cap Inv(\ell'))\} \tag{2.4}$$

The *reach set* of $\mathcal{H}$ from a set of initial symbolic states $\mathcal{R}_0$ of $\mathcal{H}$ is the smallest set $\mathcal{R}$ of symbolic states such that

$$\mathcal{R}_0 \cup \bigcup_{(\ell,\mathcal{X}) \in \mathcal{R}} Post_D(\ell, Post_C(Flow(\ell), \mathcal{X})) \subseteq \mathcal{R}. \tag{2.5}$$

Reachability is in general undecidable except for a restricted class of linear dynamical models [24, 108]. Furthermore, algorithmic solutions for sound relaxations of the problem [82] typically suffer from either the "curse of dimensionality" (exceeding time/space complexity with respect to system size) or limited practical utility (very coarse overapproximations returning too many "false alarms"), or from both issues. One reason is because many reachability techniques require the explicit computation of the system dynamics, which most often entails integrating ordinary differential equations (ODEs). Symbolic integration is restricted to limited cases of ODEs, and in general numerical integration is hard – solving Lipschitz continuous ODEs over compact domains is a PSPACE-complete problem [117].

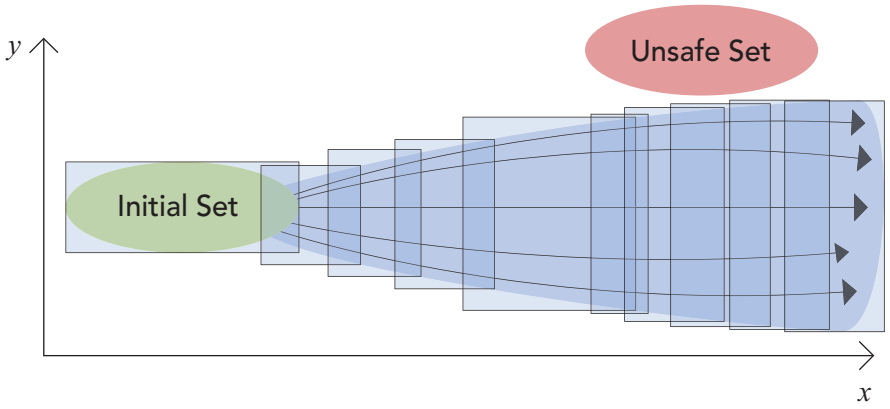


Fig. 2.1. Reachable set with box overapproximation. Green region is an initial set, red figure corresponds to the set of forbidden states and black curves are true trajectories from the initial region. The blue figure represents a reachable set, which covers all possible trajectories. Blue boxes correspond to overapproximation of the reachable set with some time interval.

2.3.2 Set Representations

Flowpipe based reachability analysis is known to be computationally expensive even for linear hybrid systems [136]. Thus, multiple approximation techniques have been introduced [94, 131, 29, 66, 47, 181] (see Figure 2.1). These set representations and

approximation techniques are the foundation of reachability algorithms [189, 136]. In this section, we define some commonly used set representations, which have been successfully used for different approximation techniques in reachability analysis [136] and are the basis for the techniques presented in this thesis.

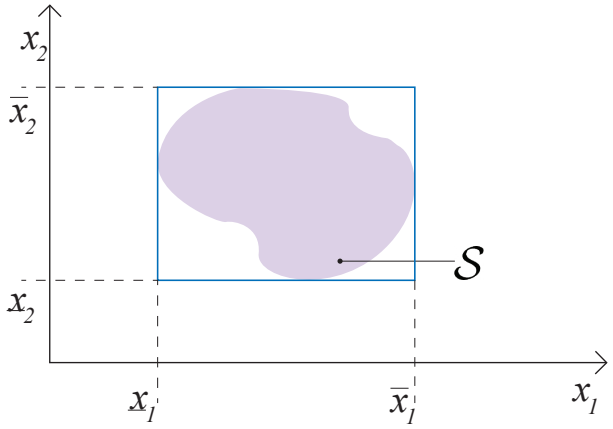


Fig. 2.2. Interval hull approximation of the non-convex sets.

Interval hull has emerged as one of the most popular approximating schemes [147] to approximate high-dimensional sets, because of its simplicity.

Definition 5. ***Interval Hull*** *[136]. A multidimensional interval hull* $\square(S)$ *is a set:*

$$\square(S) = [\underline{x_1}; \overline{x_1}] \times \ldots \times [\underline{x_n}; \overline{x_n}], \tag{2.6}$$

where for all i, $\underline{x_i} = \inf\{x_i : x \in S\}$ *and* $\overline{x_i} = \sup\{x_i : x \in S\}$.

Although an approach to approximate the set with an interval hull is simple and scalable, the drawback is that it introduces a coarse approximation.

One of the popular techniques to mitigate the limitations of the approach is to represent a set in a half-space representation, *i.e.,* as an intersection of half-spaces. An n-dimensional half-space is the set $\{\langle a, x\rangle \leq b \mid x \in \mathbb{R}^n\}$ parameterized by $a \in \mathbb{R}^n$, $b \in \mathbb{R}$. A polyhedron is an intersection of finitely many half-spaces, and a polytope is a bounded polyhedron.

Definition 6. ***Polytope.*** *Given a matrix* $H \in \mathbb{R}^{s\times n}$ *and vector* $d \in \mathbb{R}^s$*, the halfspace representation of a polytope* $\mathcal{P} \subset \mathbb{R}^n$ *is defined as*

$$\mathcal{P} := \{x \in \mathbb{R}^n \mid H\,x \leq d\}.$$

We use the shorthand $\mathcal{P} = \langle H, d\rangle_P$.

A halfspace $\mathcal{H} \subset \mathbb{R}^n$ is a special case of a polytope consisting of a single inequality constraint $h^T\,x \leq d$ with $h \in \mathbb{R}^n$, $d \in \mathbb{R}$. We use the shorthand $\mathcal{H} = \langle h, d\rangle_H$.

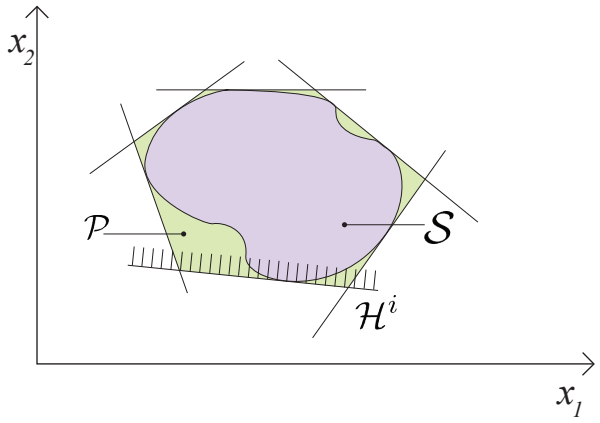


Fig. 2.3. H-Representation of the green polytope $\mathcal{P}$, which is an overapproximation of the set $\mathcal{S}$.

However, each set can be approximated by H-representation in infinite different ways. Support-function representation is a systematic way to overapproximate a set. It can be used to represent a flowpipe (a sequence of sets that covers the behaviors of a system) symbolically [95].

Definition 7. ***Support Functions*** *[136]. The support function of a set $\mathcal{S}$ is defined by:*

$$\begin{aligned} \rho_{\mathcal{S}}: \quad & \mathbb{R}^n \to \mathbb{R} \cup \{-\infty, \infty\}, \\ & \ell \mapsto \sup_{x \in \mathcal{S}} x \cdot \ell \end{aligned} \tag{2.7}$$

where x is a point and ℓ is a vector. A point x^*, such that $\langle x^*, \ell \rangle = \rho_{\mathcal{S}}(\ell)$ is called a support vector of $\mathcal{S}$ in direction ℓ. $\rho_{\mathcal{S}}(\ell)$ provides a value to correctly place a hyperplane $\mathcal{H}p$ orthogonal to ℓ, so that it touches $\mathcal{S}$ as in Figure 2.4. This hyperplane is referred to as a supporting hyperplane.

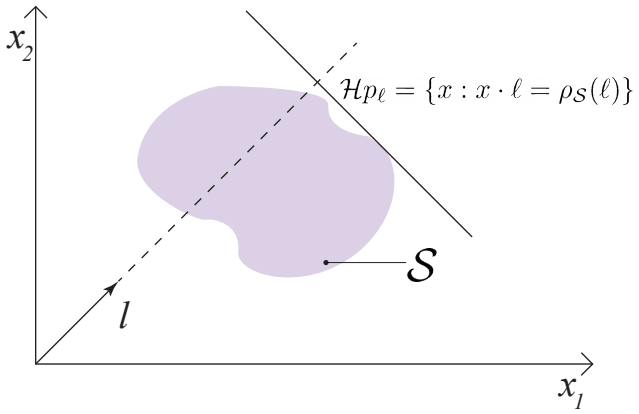


Fig. 2.4. Demonstration of the support hyperplane in the directions ℓ.

Support function can be used only to compute a convex hull of $\mathcal{S}$:

$$\mathrm{CH}(\mathcal{S}) = \bigcap_{\ell \in \mathbb{R}^n} \{x : x \cdot \ell \leq \rho_{\mathcal{S}}(\ell)\} \tag{2.8}$$

The Hausdorff distance of two sets $\mathcal{X}, \mathcal{Y} \in \mathcal{C}_n$ with $\mathcal{X} \subseteq \mathcal{Y}$ can also be expressed in terms of the support function as

$$d_H^p(\mathcal{X}, \mathcal{Y}) = \max_{\|d\|_p \leq 1} \rho_{\mathcal{Y}}(d) - \rho_{\mathcal{X}}(d).$$

Support function representation supports lazy calculations, e.g., computing the set operations on request. This can be very useful for efficient reachability analysis of hybrid systems, which contains operations suchs linear maps, intersections, Minkowski sums and convex hulls. Lazy operations allow only sets that are of interest, e.g., those that intersect with a linear constraint, to be then approximated [83].

$$\rho_{A\mathcal{X}}(\ell) = \rho_{\mathcal{X}}(A^\top \ell) \tag{2.9}$$
$$\rho_{\lambda\mathcal{X}}(\ell) = \rho_{\mathcal{X}}(\lambda\ell) = \lambda\rho_{\mathcal{X}}(\ell) \tag{2.10}$$
$$\rho_{\mathcal{X}\oplus\mathcal{Y}}(\ell) = \rho_{\mathcal{X}}(\ell) + \rho_{\mathcal{Y}}(\ell) \tag{2.11}$$
$$\rho_{\mathrm{CH}(\mathcal{X}\cap\mathcal{Y})}(\ell) = \max(\rho_{\mathcal{X}}(\ell), \rho_{\mathcal{Y}}(\ell)) \tag{2.12}$$

Another special type of polytopes are zonotopes, which can be defined using a center vector and generators:

Definition 8. ***Zonotope.*** *Given a center vector $c \in \mathbb{R}^n$ and a generator matrix $G \in \mathbb{R}^{n\times p}$, a zonotope $\mathcal{Z} \subset \mathbb{R}^n$ is defined as*

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

where the scalars α_i are called factors. We use the shorthand $\mathcal{Z} = \langle c, G \rangle_Z$.

Lazy calculations can be applied in analyses based on zonotopes [94, 96, 18, 15].

Appoximations using approaches defined above are restricted to represent only convex sets, while reachability analysis sometimes require to provide close approximations of non-convex sets. Taylor model arithmetic [142] can be utilized to compute tight non-convex enclosures for the image through a nonlinear function. It is based on a set representation called Taylor models:

Definition 9. ***Taylor Model.*** *Given a polynomial function $p : \mathbb{R}^s \to \mathbb{R}^n$, an interval domain $\mathcal{D} \subset \mathbb{R}^s$, and an interval remainder $\mathcal{Y} \subset \mathbb{R}^n$, a Taylor model $\mathcal{T}(x)$ is defined as*

$$\forall x \in \mathcal{D} : \quad \mathcal{T}(x) := \left\{ p(x) + y \;\middle|\; y \in \mathcal{Y} \right\}.$$

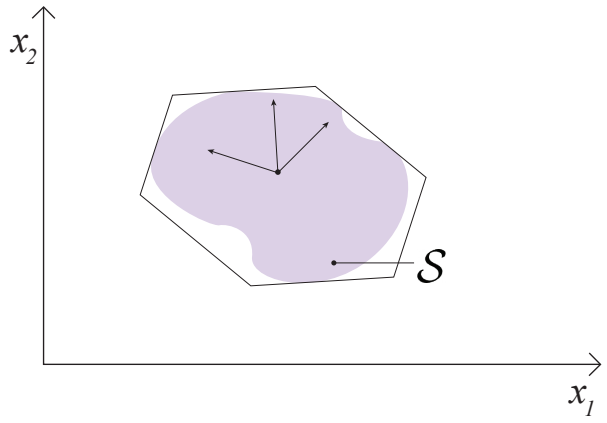


Fig. 2.5. Approximation of the set $\mathcal{S}$ by zonotope

The Taylor order $\kappa \in \mathbb{N}$ defines an upper bound for the polynomial degree of the polynomial $p(x)$. The set defined by a Taylor model is

$$\left\{\mathcal{T}(x) \;\middle|\; x \in \mathcal{D}\right\} = \left\{p(x) + y \;\middle|\; x \in \mathcal{D},\ y \in \mathcal{Y}\right\}.$$

For a concise notation we use the shorthand $\mathcal{T}(x) = \langle p(x), \mathcal{Y}, \mathcal{D}\rangle_T$.

The general concept of Taylor model arithmetic is to define rules on how to perform the arithmetic operations $+$, $-$, $\cdot$, and $/$ as well as elementary functions such as $\sin(x)$ or $\sqrt{x}$ on Taylor models [142, Sec. 2]. Since every nonlinear function represents a composition of arithmetic operations and elementary functions, the image through the function can then be computed by successively evaluating those rules. Given two one-dimensional Taylor models $\mathcal{T}_1(x) = \langle p_1(x), \mathcal{Y}_1, \mathcal{D}\rangle_T$ and $\mathcal{T}_2(x) = \langle p_2(x), \mathcal{Y}_2, \mathcal{D}\rangle_T$ the rules for addition and multiplication are for example given as

$$\begin{aligned}
\mathcal{T}_1(x) + \mathcal{T}_2(x) &:= \Big\langle p_1(x) + p_2(x), \mathcal{Y}_1 + \mathcal{Y}_2, \mathcal{D}\Big\rangle_T \\
\mathcal{T}_1(x) \cdot \mathcal{T}_2(x) &:= \Big\langle p_1(x) \cdot p_2(x), \mathcal{Y}_1 \cdot \mathcal{Y}_2 + \mathcal{I}_1 \cdot \mathcal{Y}_2 + \mathcal{Y}_1 \cdot \mathcal{I}_2, \mathcal{D}\Big\rangle_T,
\end{aligned}$$

where $\mathcal{I}_1 = \{p_1(x) \mid x \in \mathcal{D}\}$ and $\mathcal{I}_2 = \{p_2(x) \mid x \in \mathcal{D}\}$. The rules for elementary functions are obtained using a finite Taylor series expansion, where the order of the Taylor series is equal to the Taylor order κ. For $\sin(x)$ we for example obtain with $\kappa = 2$ the rule

$$\sin\Big(\mathcal{T}_1(x)\Big) := \Big\langle \sin(c) + \cos(c)\,(p_1(x) - c) - 0.5\,\sin(c)\,(p_1(x) - c)^2, \mathcal{Y}, \mathcal{D}\Big\rangle_T,$$

where the expansion point c is chosen as $c = p_1(c_d)$ with c_d being the center of the domain $\mathcal{D}$, and the interval $\mathcal{Y}$ computed according to [142, Sec. 2] encloses the remainder of the Taylor series. Due to the finite Taylor series approximation, Taylor model arithmetic yields a tight enclosure rather than the exact image. The accuracy of the enclosure can be improved by choosing a larger Taylor order.

Polynomial zonotopes are a novel non-convex set representation that has been originally introduced for reachability analysis of nonlinear systems [10]. We use the sparse representation of polynomial zonotopes [124][1]:

Definition 10. ***Polynomial Zonotope.*** *Given a constant offset $c \in \mathbb{R}^n$, 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 $E \in \mathbb{N}_0^{p \times h}$, a polynomial zonotope $\mathcal{PZ} \subset \mathbb{R}^n$ is defined as*

$$\mathcal{PZ} := \left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \alpha_k, \beta_j \in [-1, 1] \right\}.$$

The scalars α_k are called dependent factors since a change in their value affects multiplication with multiple generators. Consequently, the scalars β_j are called independent factors because they only affect multiplication with one generator. We use the shorthand $\mathcal{PZ} = \langle c, G, G_I, E \rangle_{PZ}$.

Using polynomial zonotopes for verification has two main advantages:

1. Due to the similarity with Taylor models the set defined by a Taylor model can be equivalently represented as a polynomial zonotope [124, Prop. 4].

2. Due to the similarity with zonotopes tight enclosing zonotopes can be computed efficiently for polynomial zonotopes [124, Prop. 5].

2.3.3 Reachability analysis techniques for different types of systems

Reachability analysis of linear systems generally scales better than algorithms for nonlinear systems. There are several techniques to improve efficiency of the algorithm for reachability analysis of linear systems. Linear hybrid systems can be explored efficiently in a symbolic way, e.g., using bounded model checking [57]. There are approaches which utilize a decomposition in the continuous state space. Schupp et al. perform such a decomposition by syntactic independence [179], which corresponds to dynamics with block-diagonal matrices (whereas our decomposition is generally applicable). For purely continuous systems there exist various decomposition approaches. For instance, an approach presented in [52], which decomposes the system into blocks and exploits the linear dynamics to avoid the wrapping effect, i.e. accumulation of approximation error over time. Other approaches for LTI systems are based on Krylov subspace approximations [105], time-scale decomposition [76, 98], similarity transformations [118, 119], projectahedra [100, 199], and sub-polyhedra abstract domains [180].

Flowpipe computation techniques for computing reachable set using Taylor Models [65], polynomial Zonotopes [13], sample trajectories [79], and SMT solvers [126] have all been

[1] In contrast to [124, Def. 1], we explicitly do not integrate the constant offset c in G. Moreover, we omit the identifier vector used in the original work [124] for simplicity.

investigated as potential techniques for nonlinear systems verification. These methods work directly on the nonlinear differential equations, which create scalability challenges.

There are approaches for nonlinear systems are based on projections with differential inclusions [28] and Hamilton-Jacobi methods [153, 60].

Domain contraction, the practice of narrowing the domain of possible solutions, has been used in solving constraint satisfaction over nonlinear real arithmetic [49, 81, 89, 99]. The same technique has been used in hybrid systems safety verification [169] and checking for intersection of flowpipes with guard sets for discrete transitions [16, 64].

Several techniques have been presented to shift from nonlinear differential equations to linear. Transforming a nonlinear ODE into a linear ODE by performing change of variables has been previously investigated [174]. The main goal is to a) search for the change of variables transformation such that nonlinear dynamical and hybrid systems become linear dynamical and hybrid systems and b) synthesize invariants for the linear system to prove safety. The process of approximating a nonlinear dynamical system as a piecewise linear system with uncertain inputs is called hybridization [30, 32, 71, 106, 39, 140, 62]. The inputs overapproximate the divergence between the linear and nonlinear dynamics in a subspace defined by the invariant of each mode in the hybrid system.

The first challenge in performing safety verification using hybridization is that the number of modes in the hybrid system might be prohibitively large.

Secondly, it requires computing intersections of flowpipes with the guards for discrete transitions, which might become expensive [96].

2.3.4 Intersection as part of reachability analysis

Computing intersections is a major challenge in reachability analysis of hybrid systems because it usually requires a conversion from efficient set representations (like zonotopes, support function, or Taylor models) to polytopes and back, often entailing additional approximation. Some methods to avoid computing these intersections have been investigated [16, 39].

A coarse approximation of an intersection of a set $\mathcal{X}$ and another set $\mathcal{Y}$ can be obtained by only detecting a nonempty intersection (which is generally easier to do) and then taking the original set $\mathcal{X}$ as overapproximation [155]. In general, the intersection between a polytope and polyhedral constraints (invariants and guards) can be computed exactly, but such an approach is not scalable [67]. Girard and Le Guernic consider hyperplanar constraints where reachable states are either zonotopes, in which case they work in a two-dimensional projection [96], or general polytopes [101, 136]. The tool SpaceEx approximates the intersection of polytopes and general polyhedra using template directions [84]. Frehse and Ray propose an optimization scheme for the intersection of a compact convex set $\mathcal{X}$, represented by its support function, and a polyhedron $\mathcal{Y}$, and this scheme is exact if $\mathcal{X}$ is a polytope [83]. The problem of performing intersections can also be cast in terms of finding separating hyperplanes [86, 51]. Althoff et al. approximate zonotopes by parallelotopes before considering the intersection [18]. For must semantics, Althoff and

Krogh use constant-dynamics approximation and obtain a nonlinear mapping [16]. Under certain conditions, Bak et al. apply a model transformation by replacing guard constraints by time-triggered constraints, for which intersection is easy [40].

Performing intersections in low dimensions allows for efficient computations that are not possible in high dimensions. For example, checking for emptiness of a polyhedron in constraint representation is a feasibility linear program, which can be solved in weakly polynomial time, but solutions in strongly polynomial time are only known in two dimensions [111].

2.4 Verification via barrier certificates

Barrier certificates, first introduced in [160], can be used to certify safety of dynamical systems.

Definition 11. ***Barrier Certificate.*** *Given an initial set $\mathcal{X}_0$ and unsafe set $\mathcal{X}_u$, a function $B : \mathcal{X} \to \mathbb{R}$ is called a* barrier certificate *for the system* (2.2) *if it satisfies*

$$\begin{aligned} &B(x) \leq 0 \ \forall x \in \mathcal{X}_0 \text{ and } B(x) > 0 \ \forall x \in \mathcal{X}_u, \\ &\frac{\partial B(x)}{\partial x} f(x) < 0, \ \forall x \in \mathcal{X} \text{ s.t. } B(x) = 0. \end{aligned} \tag{2.13}$$

Theorem 1 ([161, 162])**.** *Given the initial set $\mathcal{X}_0$ and unsafe set $\mathcal{X}_u$, the system is safe if there exists a barrier certificate $B(\cdot)$ satisfying the conditions* (2.2)*.*

The characterisation of a barrier certificate B relies on its *Lie derivative*, which enters the last requirement in (2.13). The Lie derivative of a continuously differentiable scalar function B with respect to a vector field f is

$$\dot{B}(x) = \nabla B(x) \cdot f(x) = \sum_{i=1}^{n} \frac{\partial B}{\partial x_i} f_i(x), \tag{2.14}$$

where x_i is the i^{th} element of the state: $x = (x_1, x_2, \ldots, x_n)$. The Lie derivative describes how the value of the barrier certificate varies along with the flow of f, namely along trajectories of the model. Indeed, if we consider a trajectory $x(t)$, with $x(0) \in \mathcal{X}_0$, and the value of $B(x)$ along this trajectory, the first condition ensures $B(x(0)) \leq 0$. The condition on the Lie derivative then guarantees that the evolution of $x(t)$ cannot make $B(x)$ become positive, and hence $x(t)$ cannot enter $\mathcal{X}_U$ (where $B(x)$ is positive). The concept of barrier certificates can be applied to both linear and nonlinear systems [161, 70].

Chapter 3

Reachability analysis of linear hybrid systems via block decomposition

Due to the computational complexity of reachability analysis of hybrid systems, tools are usually limited to systems with a small number of state variables. The state-of-the-art tools, such as SpaceEx, support systems with 200 state variables [84], while most of the other tools are restricted to dozens of state variables [177]. Some simulation-based approaches exist to support systems with up to billion dimensions [42], however, such approaches do not support hybrid setting and require a lot of constraints on the system under consideration and its settings. Therefore, an algorithm to perform reachability analysis on systems with mixed discrete-continuous dynamics and thousands of variables would provide a step forward to address a scalability issue in verification.

3.1 Chapter overview and structure

In this chapter, we describe a new reachability algorithm for linear hybrid systems, *i.e.,* , hybrid systems with dynamics given by linear differential equations and invariants and guards given by linear inequalities. We enhance both continuous and discrete operators and make sure that the involved costly computations, such as intersections, are performed in low-dimensional state space.

In particular, we improve the continuous-post operator by performing computations in high-dimensional state space only for time intervals relevant for the subsequent application of the discrete-post operator. To this end, we integrate (and modify) a recent reachability algorithm for (purely continuous) LTI systems [52], which we call $Post_C^{\square}$ in the following, in a new algorithm for linear hybrid systems.

Furthermore, the new discrete-post operator performs low-dimensional computations by leveraging the structure of the guard and assignment of a considered transition.

The $Post_C^{\square}$ algorithm decomposes the calculation of the reachable states into calculations in subspaces (called *blocks*). This decomposition has two benefits. The first benefit is that computations in lower dimensions are generally more efficient and thus the algorithm is highly scalable, *i.e.* enabling to reason about the systems with hunderds of state variables.

The second benefit is that the analysis for different subspaces is decoupled; hence one can effectively skip the computations for dimensions that are of no interest (e.g., for a safety property).

Conceptually, reachability analysis for hybrid systems is performed in a "hybrid loop" that interleaves a continuous-post algorithm and a discrete-post algorithm. If we consider $Post_C^{\square}$ as a black box, we can plug it into this hybrid loop, which we refer to as $Post_H$. However, there are two shortcomings of such a naive integration. First, $Post_H$ does not utilize the features from decoupling of $Post_C^{\square}$ at all. It results into a second flaw, that all operations besides $Post_C^{\square}$ are still performed in high dimensions, making $Post_H$ still suffer from scalability issues.

We demonstrate that, unlike in $Post_H$, it is possible to perform all computations in low dimensions. For this purpose, we modify $Post_C^{\square}$ as well as $Post_H$. Since most of the models involve guards and invariants where constraints are only in several dimensions, in common cases, our algorithm does not introduce an additional approximation error. Furthermore, our algorithm makes proper use of the second benefit of $Post_C^{\square}$ by computing the reachable states only in specific dimensions whenever possible.

We implemented the algorithm in JuliaReach, a toolbox for reachability analysis [53, 2], and we evaluate the algorithm on several benchmark problems, including a 1024-dimensional hybrid system (see Section 3.4). Our algorithm outperforms the naïve $Post_H$ and other state-of-the-art algorithms by several orders of magnitude. This chapter consist of the results which are basis of our paper [54].

In short, the contributions of this chapter can be described as follows:

- We show how to modify the decomposed reachability algorithm for LTI systems from [52] in order to integrate it efficiently into a new, decomposed reachability algorithm for linear hybrid systems.
- We exploit the decomposed structure of the algorithm to perform all operations of hybrid reachability analyisis in low dimensions.
- We only compute the reachable states in specific dimensions whenever possible.

The chapter is structured as follows:

- We recapitulate $Post_H$ in Section 3.2.
- We show how to modify the decomposed reachability algorithm for LTI systems $Post_C^{\square}$ in order to efficiently integrate it into a new, decomposed reachability algorithm for linear hybrid systems in Section 3.3.
- Finally, the evaluation of the proposed approach is presented in Section 3.4.

3.2 Reachability analysis of linear hybrid systems

Our reachability algorithm for linear hybrid systems is centered around the algorithm from [52], called $Post_C^{\square}$ for convenience, which implements $Post_C$ for LTI systems in a

compositional way. In this section, we first recall the $Post_C^{\square}$ algorithm. Two important properties of $Post_C^{\square}$ are that (1) the output is a sequence of *decomposed* sets and that (2) this sequence is computed efficiently in low dimensions.

After explaining the algorithm $Post_C^{\square}$, we incorporate it in a standard reachability algorithm for linear hybrid systems. However, this standard reachability algorithm will not make use of the above-mentioned properties. This motivates our new algorithm (presented in Section 3.3), which is a modification of both this standard reachability algorithm and $Post_C^{\square}$ to make optimal use of these properties.

3.2.1 Decomposed reachability analysis of LTI systems

The decomposition-based approach [52] follows a flowpipe-construction scheme using time discretization, which we briefly recall here. Given an LTI system $(A, B, \mathcal{U})$ and a set of initial (continuous) states $\mathcal{X}_0$, by fixing a time step δ we first compute a set $\mathcal{X}(0)$ that overapproximates the reach set up to time δ, a matrix $\Phi = e^{A\delta}$ that captures the dynamics of duration δ, and a set $\mathcal{V}$ that overapproximates the effect of the inputs up to time δ. We use β to denote the number of blocks in a partition. Let $\{\pi_j\}_{j=1}^{\beta}$ be a set of (contiguous) projection matrices that partition a vector $x \in \mathbb{R}^n$ into $x = [\pi_1 x, \ldots, \pi_\beta x]$. Given a set $\mathcal{X}$ and projection matrices $\{\pi_j\}_{j=1}^{\beta}$, we call $\pi_j\mathcal{X}$ a *block* of $\mathcal{X}$ and $\bigtimes_j \pi_j\mathcal{X} = \pi_1\mathcal{X} \times \cdots \times \pi_\beta\mathcal{X}$ the *Cartesian decomposition* with the block structure induced by projections π_j. We write $\widehat{\mathcal{X}}$ to indicate a decomposed set (i.e., a Cartesian product of lower-dimensional sets). For instance, given a nonempty set $\mathcal{X} \in \mathcal{C}_n$, its box approximation is the Cartesian decomposition into intervals (i.e., one-dimensional blocks). We can bound the approximation error by the radius of $\mathcal{X}$.

Proposition 1. *Let $\mathcal{X} \in \mathcal{C}_n$ be nonempty, $p = \infty$, $r_{\mathcal{X}}^p$ be the radius of the box approximation of $\mathcal{X}$, and let π_j be appropriate projection matrices. Then $d_H^p(\mathcal{X}, \bigtimes_j \pi_j\mathcal{X}) \leq \left\| r_{\mathcal{X}}^p \right\|_p$.*

We obtain an overapproximation of the reach set in time interval $[k\delta, (k+1)\delta]$, for step $k > 0$, with

$$\mathcal{X}(k) := \Phi\mathcal{X}(k-1) \oplus \mathcal{V} = \Phi^k\mathcal{X}(0) \oplus \bigoplus_{j=0}^{k-1}\Phi^j\mathcal{V}.$$

Algorithm $Post_C^{\square}$ decomposes this scheme. Fixing some block structure, let $\widehat{\mathcal{X}}(0) := \mathcal{X}_1(0) \times \cdots \times \mathcal{X}_\beta(0)$ be the corresponding Cartesian decomposition of $\mathcal{X}(0)$. We compute a sequence $\widehat{\mathcal{X}}(k) := \mathcal{X}_1(k) \times \cdots \times \mathcal{X}_\beta(k)$ such that $\mathcal{X}(k) \subseteq \widehat{\mathcal{X}}(k)$ for every k. Each low-dimensional set $\mathcal{X}_i(k)$ is computed as

$$\mathcal{X}_i(k) := \bigoplus_{j=1}^{\beta}(\Phi^k)_{ij}\mathcal{X}_j(0) \oplus \bigoplus_{j=0}^{k-1}[(\Phi^j)_{i1} \cdots (\Phi^j)_{i\beta}]\mathcal{V}.$$

The above sequences $\mathcal{X}(k)$ resp. $\widehat{\mathcal{X}}(k)$ are called flowpipes.

Safety properties can be given as a set of symbolic error states that should be avoided and be encoded as the guard of a transition to a new error location. In our implementation we can also check inclusion in the safe states (the complement) and do not require an encoding with additional transitions.

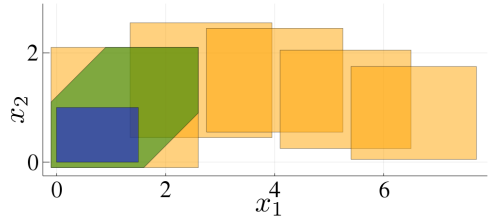


Fig. 3.1. Starting from the set of initial states $\mathcal{X}_0$ (blue set), we first compute the set $\mathcal{X}(0)$ by time discretization (green set), then decompose the set into intervals and obtain $\widehat{\mathcal{X}}(0)$ (orange box around $\mathcal{X}(0)$), and finally compute the (decomposed) flowpipe $\widehat{\mathcal{X}}(1), \ldots, \widehat{\mathcal{X}}(4)$ by propagating each of the intervals (other orange sets).

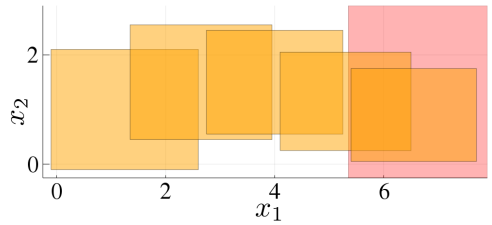


Fig. 3.2. The flowpipe from Figure 3.1 together with a guard (red).

Example We illustrate the algorithms with a running example throughout the chapter. For illustration purposes, the example is two-dimensional (and hence we decompose into one-dimensional blocks, but we emphasize that the approach also generalizes to higher-dimensional decomposition) and we consider a hybrid system with only a single location and one transition (a self-loop). Figure 3.1 depicts the flowpipe construction for the example.

3.2.2 Reachability analysis of linear hybrid systems

We now discuss a standard reachability algorithm for hybrid systems. Essentially, this algorithm interleaves the operators $Post_C$ and $Post_D$ following (2.5) until it finds a fixpoint. Here we use $Post_C^{\square}$ as the continuous-post operator.

We first compute a flowpipe $\widehat{\mathcal{X}} = \widehat{\mathcal{X}}(0), \ldots, \widehat{\mathcal{X}}(N)$ using $Post_C^{\square}$ as described above. Then we use $Post_D$ to take a discrete transition. According to (2.4), we want to compute $((A, b) \odot (\widehat{\mathcal{X}} \cap \mathcal{I}_1 \cap \mathcal{G})) \cap \mathcal{I}_2$, where (A, b) is a deterministic affine map $\mathcal{I}_1$ and $\mathcal{I}_2$ are the source and target invariant, and $\mathcal{G}$ is the guard. Frehse and Ray showed that for such maps the term can be simplified to

$$(A, b) \odot (\widehat{\mathcal{X}} \cap \mathcal{G}^*) \tag{3.1}$$

where the set $\mathcal{G}^*$ can be statically precomputed [83], which is usually easy because the sets $\mathcal{I}_1$, $\mathcal{G}$, and $\mathcal{I}_2$ are given as polyhedra in constraint representation (also called H-representation, as opposed to the (vertex) V-representation). Hence we ignore invariants in the rest of the presentation for simplicity.

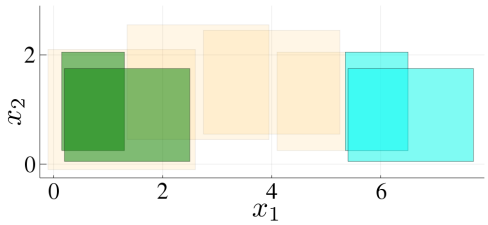


Fig. 3.3. The assignment shifts the intersection of the flowpipe and the guard from Figure 3.2 (cyan) to the green sets, which are both contained in the first set of the original flowpipe.

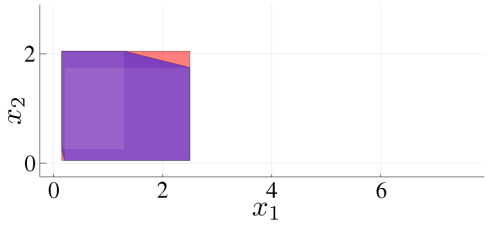


Fig. 3.4. Approximation of the union of the green sets from Figure 3.3 using the convex hull (purple) and the decomposed convex hull (red).

Example We continue with the flowpipe from Figure 3.2. The guard $\mathcal{G}$ is a half-space that is constrained in dimension x_1 and unconstrained in dimension x_2. Only the last two sets in the flowpipe intersect with the guard. The assignment here is a translation in dimension x_1. The resulting intersection, before and after the translation, is depicted in Figure 3.3.

Finally, the algorithm checks for a fixpoint, i.e., for inclusion of the symbolic states we computed with $Post_D$ in previously-seen symbolic states.

Example The green set in Figure 3.3 shows that the two sets we obtained from $Post_D$ are contained in $\widehat{\mathcal{X}}(0)$ computed before. (Recall that in this example we only consider a single location; hence the inclusion of the sets implies inclusion of the symbolic states.)

The steps outlined above describe one iteration of the standard reachability algorithm. Each symbolic state for which the fixpoint check was negative spawns a new flowpipe. Since this can lead to a combinatorial explosion, one typically applies a technique called *clustering* (cf. [84]), where symbolic states are merged after the application of $Post_D$. Here we consider clustering with a convex hull.

Example Assume that the fixpoint check above was negative for both sets that we tested. In Figure 3.4 we show the convex hull of the sets in purple.

Up to now, we have seen a standard incorporation of an algorithm for the continuous-post operator $Post_C$ (for which we used $Post_C^{\square}$) into a reachability algorithm for hybrid systems. Observe that $Post_C$ was used as a black box. Consequently, we could not make use of the properties of the specific algorithm $Post_C^{\square}$. In particular, apart from $Post_C^{\square}$, we performed all computations in high dimensions. In the next section, we describe a new algorithm that instead performs all computations in low dimensions.

Algorithm 1 Function `reach`

Require: $\mathcal{D} = (\Phi, \mathcal{V}(\cdot), \mathcal{X}(0))$: discrete system
N: total number of steps
blocks: list of constrained block indices
other_blocks: list of unconstrained block indices
constraints: linear constraints of the outgoing transitions

$all\_blocks \leftarrow blocks \cup other\_blocks$
$\widehat{\mathcal{X}}(0) \leftarrow decompose(\mathcal{X}(0), all\_blocks)$
$P \leftarrow identity(dim(\Phi))$
$Q \leftarrow \Phi$
$\hat{\mathcal{V}}_{\text{tmp}} \leftarrow []$
for $b_i \in blocks$ **do**
    $\hat{\mathcal{V}}_{\text{tmp}}[b_i] \leftarrow$
    $\{\mathbf{0}_{dim(b_i)}\}$
end for
for $k = 1$ to $N - 1$ **do**
    $\widehat{\mathcal{X}}_{\text{tmp}} \leftarrow []$
    for $b_j \in all\_blocks$ **do**
        $\hat{\mathcal{V}}_{\text{tmp}}[b_i] \leftarrow$
        $\hat{\mathcal{V}}_{\text{tmp}}[b_i] \oplus P[b_i, :] \odot \mathcal{V}$
    end for
    $\widehat{\mathcal{X}}_{\text{tmp}} = reach\_blocks(\widehat{\mathcal{X}}(0), \widehat{\mathcal{X}}_{\text{tmp}}, Q, \hat{\mathcal{V}}_{\text{tmp}}, blocks, all\_blocks)$
    if !isdisjoint($\widehat{\mathcal{X}}_{\text{tmp}}$, constraints) **then** ▷ compute high-dimensional sets only if intersection with guards is nonempty
        $\widehat{\mathcal{X}}_{\text{tmp}} = reach\_blocks(\widehat{\mathcal{X}}(0), \widehat{\mathcal{X}}_{\text{tmp}}, Q, \hat{\mathcal{V}}_{\text{tmp}}, other\_blocks, all\_blocks)$
    end if
    $\widehat{\mathcal{X}}(k) \leftarrow \widehat{\mathcal{X}}_{\text{tmp}}$
    $P \leftarrow Q$
    $Q \leftarrow Q \cdot \Phi$
end for
return $[\widehat{\mathcal{X}}(0), \widehat{\mathcal{X}}(1), ..., \widehat{\mathcal{X}}(N-1)]$

function REACH_BLOCKS($\widehat{\mathcal{X}}(0)$, $\widehat{\mathcal{X}}_{\text{tmp}}$, Q, $\hat{\mathcal{V}}_{\text{tmp}}$, *blocks*, *all_blocks*)
    for $b_i \in blocks$ **do**
        $\widehat{\mathcal{X}}_{\text{tmp}}[b_i] \leftarrow \{\mathbf{0}_{dim(b_i)}\}$
        for $b_j \in all\_blocks$ **do**
            $\widehat{\mathcal{X}}_{\text{tmp}}[b_i] \leftarrow \widehat{\mathcal{X}}_{\text{tmp}}[b_i] \oplus Q[b_i, b_j] \odot \widehat{\mathcal{X}}(0)[b_j]$
        end for
        $\widehat{\mathcal{X}}_{\text{tmp}}[b_i] \leftarrow \widehat{\mathcal{X}}_{\text{tmp}}[b_i] \oplus \hat{\mathcal{V}}_{\text{tmp}}[b_i]$
    end for return *return* $\widehat{\mathcal{X}}_{\text{tmp}}$
end function

3.3 Decomposed reachability analysis

We now present a new, decomposed reachability algorithm for linear hybrid systems. The algorithm uses a modified version of $Post_C^{\square}$ for computing flowpipes and has two major performance improvements over the algorithm seen before.

Recall that $Post_C^{\square}$ computes flowpipes consisting of decomposed sets. The first improvement is to exploit the decomposed structure to perform all other operations (intersection, affine map, inclusion check, and convex hull) in low dimensions.

The second improvement is to compute flowpipes in a sparse way. Roughly speaking, we are only interested in those dimensions of a flowpipe that are relevant to determine an

intersection with a guard. We only need to compute the other dimensions of the flowpipe if we detect such an intersection.

The algorithm starts as before: given an initial (symbolic) state, we compute $\mathcal{X}(0)$ (discretization) and decompose the set to obtain $\widehat{\mathcal{X}}(0)$. Next we want to compute a flowpipe, and this is where we deviate from the previous algorithm.

3.3.1 Computing a sparse flowpipe

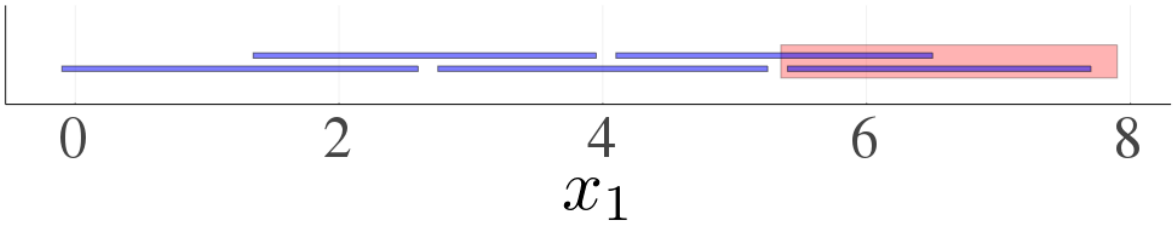
We modify $Post_C^{\square}$ in order to control the dimensions of the flowpipe. Recall that the black-box version of $Post_C^{\square}$ computed the flowpipe $\widehat{\mathcal{X}}(k) = \mathcal{X}_1(k) \times \cdots \times \mathcal{X}_\beta(k)$ for $k = 1, \ldots, N$, i.e., in all dimensions. This is usually not necessary for detecting an intersection with a guard. We will discuss this formally below, but want to establish some intuition first. Recall the running example from Section 3.2.1. The guard was only constrained in dimension x_1. This means that the bounds of the sets $\widehat{\mathcal{X}}(k) = \mathcal{X}_1(k) \times \mathcal{X}_2(k)$ in dimension x_2 are irrelevant. Consequently, we do not need to compute the sets $\mathcal{X}_2(k)$ at all (at least for the moment). We only compute those dimensions of a flowpipe that are necessary to determine intersection with the guards. Identifying these dimensions and projecting the guards accordingly has to be performed only once per transition and is often just a syntactic procedure.

In particular, Algorithm 1 describes the new implementation of $Post_C^{\square}$ to compute a sparse flowpipe. The algorithm starts the same way as the original algorithm in [52] with a decomposed set in line 2 and iteratively computes a sparse flowpipe only for the dimensions of interest (line 16). However, if we detect an intersection with a guard constraint (line 17), we compute the full-dimensional flowpipe for the corresponding time frame. The computation of the inputs $\hat{\mathcal{V}}_{\text{tmp}}$ remains the same as in the original algorithm.

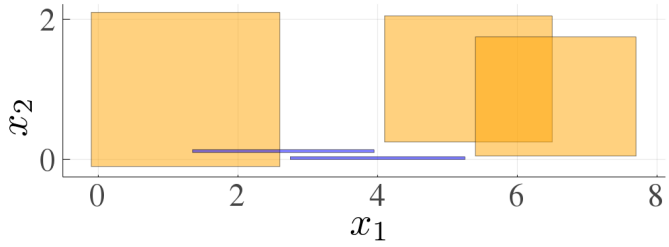
Example As discussed, we only compute the flowpipe $\mathcal{X}_1(1), \ldots, \mathcal{X}_1(4)$ for the first block (in dimension x_1), i.e., a sequence of intervals, which is depicted in Figure 3.5(a). Projecting the guard to x_1, we obtain a ray $\mathcal{G}_1$. As expected, we observe an intersection with the guard for the same time steps as before (namely steps $k = 3$ and $k = 4$).

3.3.2 Decomposing an intersection

Computing the intersection of two n-dimensional sets $\mathcal{X}$ and $\mathcal{G}$ in low dimensions is generally beneficial for performance; yet it is particularly interesting if one of the sets is a polytope that is not represented by its linear constraints. Common cases are the V-representation or zonotopes represented by their generators, which are used in several approaches [94, 96, 18, 15]. To compute the (exact) intersection of such a polytope $\mathcal{X}$ with a polyhedron $\mathcal{G}$ in H-representation, $\mathcal{X}$ needs to be converted to H-representation first. A polytope with m vertices can have $O\Big(\binom{m-n/2}{m-n}\Big)$ linear constraints [146]. (For two polytopes in V-representation *in general position* there is a polynomial-time intersection algorithm [88], but this assumption is not practical.) A zonotope with m generators can have $O\Big(m\binom{m}{n-1}\Big)$ linear constraints [18]. If $\mathcal{G}$ is a polytope in H-representation, checking



(a) The flowpipe from Figure 3.1 in dimension x_1 only consists of intervals (blue). The linear constraint $\mathcal{G}_1$ (red) is the guard $\mathcal{G}$ projected to x_1. For better visibility, we draw the sets thicker and add a slight offset to some of the intervals.



(b) Illustration of the effective flowpipe computation from Figure 3.1. Only the first set $\widehat{\mathcal{X}}(0)$ and the sets that intersect with the guard ($\widehat{\mathcal{X}}(3)$ and $\widehat{\mathcal{X}}(4)$) are computed in high dimensions.

Fig. 3.5. Illustration of the mixed sparse and high-dimensional flowpipe construction.

disjointness of $\mathcal{X}$ and $\mathcal{G}$ can also be solved more efficiently in low dimensions, e.g., for m linear constraints in $O(m)$ for $n \leq 3$ [45].

Next we discuss how to apply low-dimensional reasoning to the intersection $\widehat{\mathcal{X}} \cap \mathcal{G}$ of a decomposed set $\widehat{\mathcal{X}}$ and a polyhedron $\mathcal{G}$, or respectively detect emptiness of the intersection $\widehat{\mathcal{X}} \cap \mathcal{G}$ (which can often be achieved more efficiently). The key idea is to exploit that $\widehat{\mathcal{X}}$ is decomposed. For ease of discussion, we consider the case of two blocks (i.e., $\widehat{\mathcal{X}} = \mathcal{X}_1 \times \mathcal{X}_2$). Below we discuss the two cases that $\mathcal{G}$ is decomposed or not.

Intersection of two decomposed sets We first consider the case that $\mathcal{G}$ is also decomposed and agrees with $\widehat{\mathcal{X}}$ on the block structure, i.e., $\mathcal{G} = \widehat{\mathcal{G}} = \mathcal{G}_1 \times \mathcal{G}_2$ and $\mathcal{X}_1, \mathcal{G}_1 \subseteq \mathbb{R}^{n_1}$ for some n_1. Clearly

$$\widehat{\mathcal{X}} \cap \widehat{\mathcal{G}} = (\mathcal{X}_1 \times \mathcal{X}_2) \cap (\mathcal{G}_1 \times \mathcal{G}_2) = (\mathcal{X}_1 \cap \mathcal{G}_1) \times (\mathcal{X}_2 \cap \mathcal{G}_2)$$

because the Cartesian product and intersection distribute; thus

$$\widehat{\mathcal{X}} \cap \widehat{\mathcal{G}} = \emptyset \iff (\mathcal{X}_1 \cap \mathcal{G}_1 = \emptyset) \vee (\mathcal{X}_2 \cap \mathcal{G}_2 = \emptyset). \tag{3.2}$$

Now consider the second disjunct in (3.2) and assume that $\mathcal{G}_2$ is universal. We get $\mathcal{X}_2 \cap \mathcal{G}_2 = \emptyset \iff \mathcal{X}_2 = \emptyset$. In our context, $\widehat{\mathcal{X}}$ (and hence $\mathcal{X}_2$) is nonempty by construction. Hence (3.2) simplifies to

$$\widehat{\mathcal{X}} \cap \widehat{\mathcal{G}} = \emptyset \iff \mathcal{X}_1 \cap \mathcal{G}_1 = \emptyset,$$

so we *never* need to compute $\mathcal{X}_2$ to determine whether the intersection is empty. In practice, the set $\mathcal{G}^*$ from (3.1) takes the role of $\widehat{\mathcal{G}}$ and is often only constrained in *some*

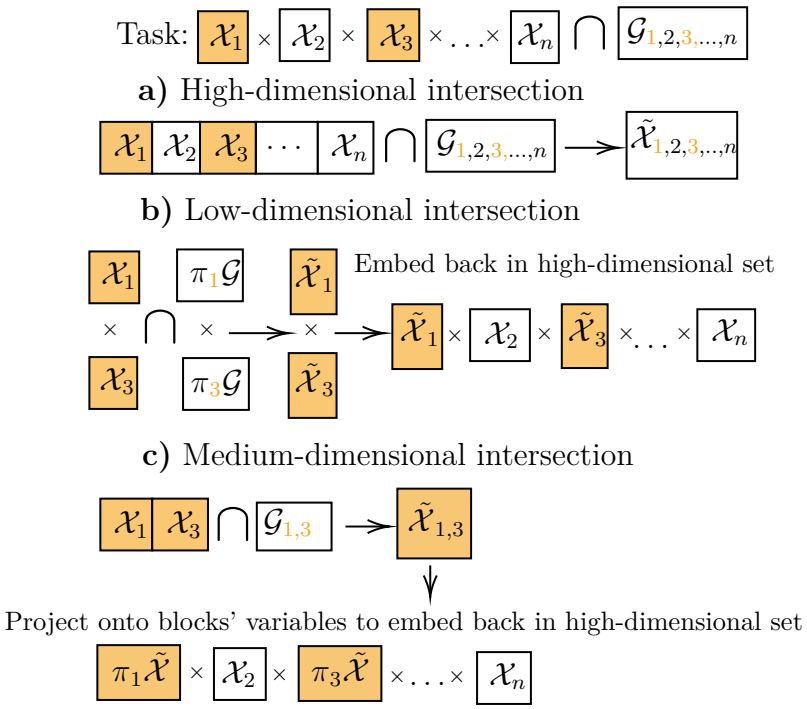


Fig. 3.6. Illustration of the different intersection algorithms. The task (top row) is to compute the intersection of a decomposed set $\widehat{\mathcal{X}}$ with a guard $\mathcal{G}$ that is constrained in blocks 1 and 3 (marked in orange). We write $\tilde{\mathcal{X}}$ for the new set after an intersection operation with set $\mathcal{X}$. **a)** The traditional high-dimensional intersection does not make use of the decomposed structure of $\widehat{\mathcal{X}}$. **b)** The low-dimensional intersection projects the guard to each block, computes the block-wise intersection, and embeds the result in high dimensions. **c)** The medium-dimensional intersection projects the guard to the constrained dimensions, computes the intersection with the Cartesian product of the corresponding sets (here: $\mathcal{X}_1$ and $\mathcal{X}_3$), projects the result to the original block structure, and finally embeds this result in high dimensions.

dimensions (and hence decomposed and universal in all other dimensions). We illustrate this algorithm in Figure 3.6 b).

Intersection of a decomposed and a non-decomposed set If $\mathcal{G}$ is not decomposed in the same block structure as $\mathcal{X}$, we can still decompose it at the cost of an approximation error. Let π_1 and π_2 be suitable projection matrices. Then

$$\widehat{\mathcal{X}} \cap \mathcal{G} \subseteq (\mathcal{X}_1 \cap \pi_1\mathcal{G}) \times (\mathcal{X}_2 \cap \pi_2\mathcal{G})$$

and hence

$$\begin{aligned} \widehat{\mathcal{X}} \cap \mathcal{G} = \emptyset &\impliedby (\mathcal{X}_1 \cap \pi_1\mathcal{G} = \emptyset) \vee (\mathcal{X}_2 \cap \pi_2\mathcal{G} = \emptyset) \\ &\impliedby \mathcal{X}_1 \cap \pi_1\mathcal{G} = \emptyset. \end{aligned} \tag{3.3}$$

From (3.3) we obtain 1) a sufficient test for emptiness of $\widehat{\mathcal{X}} \cap \mathcal{G}$ in terms of only $\mathcal{X}_1$ and 2) a more precise sufficient test in terms of $\mathcal{X}_1$ and $\mathcal{X}_2$ in low dimensions. If both tests fail, we can either fall back to the (exact) test in high dimensions or conservatively assume that the intersection is nonempty.

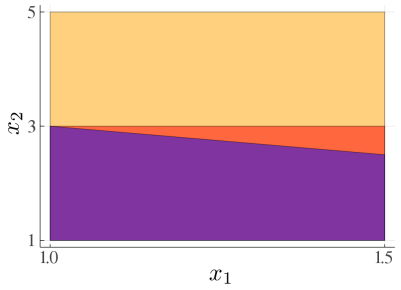


Fig. 3.7. Example of performing the different intersection algorithms. We intersect a three-dimensional hyperrectangle with the ranges $1 \leq x_1 \leq 5$, $1 \leq x_2 \leq 5$, $1 \leq x_3 \leq 5$ and a polyhedral guard with the linear constraints $x_1 + x_2 \leq 4$ and $x_1 \leq 1.5$ (observe that dimension x_3 is unconstrained). The exact intersection is the purple set, obtained using the high-dimensional intersection. The red set corresponds to the medium-dimensional intersection. The orange set corresponds to low-dimensional intersection.

The precision of the above scheme depends highly on the structure of $\widehat{\mathcal{X}}$ and $\mathcal{G}$. If several (but not all) blocks of $\mathcal{G}$ are constrained, instead of decomposing $\mathcal{G}$ into the low-dimensional block structure, one can alternatively compute the intersection for medium-dimensional sets to avoid an approximation error; we apply this strategy in the evaluation (Section 3.4). If $\mathcal{G}$ is compact, the following proposition shows that the approximation error is bounded by the maximal entry in the diameters of $\widehat{\mathcal{X}}$ and $\mathcal{G}$, and this bound is tight. This strategy is illustrated in Figure 3.6 c).

Proposition 2. *Let* $\widehat{\mathcal{X}} = \bigtimes_j \mathcal{X}_j \in \mathcal{C}_n$, $\mathcal{G} \in \mathcal{C}_n$, $\widehat{\mathcal{X}} \cap \mathcal{G} \neq \emptyset$, $\widehat{\mathcal{G}} := \bigtimes_j \pi_j \mathcal{G}$ *for appropriate projection matrices* π_j *corresponding to* $\mathcal{X}_j$, *and* $p = \infty$. *Then*

$$d_H^p(\widehat{\mathcal{X}} \cap \mathcal{G}, \widehat{\mathcal{X}} \cap \widehat{\mathcal{G}}) \leq \max_j \min(\left\|\Delta_p(\mathcal{X}_j)\right\|_p, \left\|\Delta_p(\pi_j \mathcal{G})\right\|_p).$$

Example Consider Figure 3.5(b). We have already identified the intersection with the flowpipe for time steps $k = 3$ and $k = 4$. The resulting sets are $\widehat{\mathcal{X}}(k) \cap \mathcal{G} = \mathcal{X}_1(k) \cap \mathcal{G}_1 \times \mathcal{X}_2(k)$, where $\mathcal{G}_1$ was the projection of $\mathcal{G}$ to x_1. We emphasize that we compute the intersections in low dimensions, that we need not compute $\mathcal{X}_2(1)$ and $\mathcal{X}_2(2)$ at all, and that in this example all computations are exact (i.e., we obtain the same sets as in Figure 3.3).

Now consider the case that $\mathcal{G}$ is not decomposed in the same structure as $\widehat{\mathcal{X}}$, e.g., the hyperplane $x_1 = x_2$. One option is to decompose $\mathcal{G}$ to the blocks of $\widehat{\mathcal{X}}$, i.e., $\mathcal{G}_1 := \pi_1 \mathcal{G}$, $\mathcal{G}_2 := \pi_2 \mathcal{G}$ and compute the block-wise intersection $(\mathcal{X}_1(k) \cap \mathcal{G}_1) \times (\mathcal{X}_2(k) \cap \mathcal{G}_2)$. Here $\mathcal{G}_1$ and $\mathcal{G}_2$ are universal, so we obtain a coarse approximation of the intersection (namely $\widehat{\mathcal{X}}(k)$ itself). Alternatively, computing the intersection $\widehat{\mathcal{X}}(k) \cap \mathcal{G}$ is exact but computationally more demanding. If we assume that the system has higher dimension, e.g., 10, then computing the intersection in two dimensions (i.e., with a 2D projection $(\mathcal{X}_1(k) \times \mathcal{X}_2(k)) \cap \pi \mathcal{G}$) is still exact and yet cheaper than computing the intersection in full dimensions (i.e., $\widehat{\mathcal{X}}(k) \cap \mathcal{G}$).

We exemplify the possible difference between the intersection algorithms in Figure 3.7. There we compare the three algorithms visualized in Figure 3.6. The high-dimensional

intersection is the most precise as expected because it does not suffer from a projection error. However, the result is a high-dimensional set instead of a decomposed high-dimensional set. In addition, this algorithm intersects the sets in three dimensions (including x_3) while the other algorithms ignore the unconstrained dimensions. The medium-dimensional intersection corresponds to the box approximation of the true intersection. We observe that if we cannot decompose the polyhedron to the same block structure as the decomposed set without a projection error (e.g., projecting $x_1 + x_2 \leq 4$ onto x_1 and x_2 individually results in one-dimensional universes), then the low-dimensional intersection can produce a large approximation error. Thanks to the second linear constraint $x_1 \leq 1.5$ and the fact that we treat each half-space separately, we are still able to compute a nontrivial intersection.

3.3.3 Decomposing an affine map

The next step after computing the intersection with the guard is the application of the assignment. We consider an affine map $A\widehat{\mathcal{X}} \oplus \{b\}$ with $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$. Affine-map decomposition has already been presented as part of the operator $Post_C^{\square}$ [52]:

$$A\widehat{\mathcal{X}} \oplus \{b\} \subseteq \bigtimes_i \bigoplus_j A_{ij}\mathcal{X}_j \oplus \{b_i\}$$

where A_{ij} is the block in the i-th block row and the j-th block column. We recall an error estimation.

Proposition 3. *[52, Prop. 3] Let $\mathcal{X} = \bigtimes_{j=1}^{\beta} \mathcal{X}_j \in \mathcal{C}_n$ be nonempty, $A \in \mathbb{R}^{n\times n}$, $q_j := \arg\max_i \left\|A_{ij}\right\|_p$ (the index of the block with the largest matrix norm in the j-th block column) so that $\alpha_j := \max_{i\neq q_j} \left\|A_{ij}\right\|_p$ is the second largest matrix norm in the j-th block column. Let $\alpha_{max} := \max_j \alpha_j$ and $\Delta_{sum} := \sum_j \Delta_\infty(\mathcal{X}_j)$. Then*

$$\begin{aligned} & d_H^p(A\mathcal{X}, \bigtimes_i \bigoplus_j A_{ij}\mathcal{X}_j) \\ = & \max_{\|d\|_p \leq 1} \textstyle\sum_{i,j} \rho_{\mathcal{X}_j}(A_{ij}^T d_i) - \rho_{\mathcal{X}_j}\left(\sum_k A_{kj}^T d_k\right) \\ \leq & (\beta - 1)\textstyle\sum_j \alpha_j \Delta_\infty(\mathcal{X}_j) \leq \dfrac{n}{2}\alpha_{max}\Delta_{sum}. \end{aligned}$$

In particular, if only one block per block column of matrix A is nonzero, the approximation is exact [52]. For example, consider a two-block scenario and a block-diagonal matrix A, i.e., $A_{12} = A_{21} = 0$. Then

$$\begin{aligned} \begin{pmatrix} A_{11} & 0 \\ 0 & A_{22} \end{pmatrix} \mathcal{X}_1 \times \mathcal{X}_2 \oplus \{b_1\} \times \{b_2\} \\ = (A_{11}\mathcal{X}_1 \oplus \{b_1\}) \times (A_{22}\mathcal{X}_2 \oplus \{b_2\}). \end{aligned}$$

In practice, affine maps with such a structure are unrealistic for the $Post_C^{\square}$ operator but typical for assignments. Prominent cases include resets, translations, and scalings, for which A is even diagonal and hence block diagonal for any block structure.

Example Recall that, after computing the intersections, we ended up with the same sets as in Figure 3.3. In our example, the assignment is a translation in dimension x_1. Hence, as mentioned above, the application of the decomposed assignment is also exact. In particular, the translation only affects $\mathcal{X}_1(k)$ and we obtain the same result as in Figure 3.3.

3.3.4 Inclusion check for decomposed sets

Our algorithm is now fully able to take transitions. Observe that *all* sets ever occurring in scheme (2.5) using the algorithm are decomposed. The following proposition gives an exact low-dimensional fixpoint check under this condition.

Proposition 4. *Let* $\widehat{\mathcal{X}} = \bigtimes_j \mathcal{X}_j \in \mathcal{C}_n, \widehat{\mathcal{G}} = \bigtimes_j \mathcal{G}_j \in \mathcal{C}_n$ *be nonempty sets with identical block structure. Then*

$$\widehat{\mathcal{X}} \subseteq \widehat{\mathcal{G}} \iff \bigwedge_j \mathcal{X}_j \subseteq \mathcal{G}_j.$$

3.3.5 Decomposing a convex hull

As the last part of the algorithm, we decompose the computation of the convex hull. We exploit that all sets in the same flowpipe share the same block structure.

Proposition 5. *Let* $\widehat{\mathcal{X}} = \bigtimes_j \mathcal{X}_j \in \mathcal{C}_n, \widehat{\mathcal{G}} = \bigtimes_j \mathcal{G}_j \in \mathcal{C}_n$ *be nonempty sets with identical block structure. Then*

$$\mathrm{CH}(\widehat{\mathcal{X}} \cup \widehat{\mathcal{G}}) \subseteq \bigtimes_j \mathrm{CH}(\mathcal{X}_j \cup \mathcal{G}_j).$$

For the decomposition operations proposed before (intersection, affine map, and inclusion), there are common cases where the approximations were exact. In these cases it is always beneficial to perform the decomposed operations instead of the high-dimensional counterparts. The decomposition of the convex hull, however, always incurs an approximation error, which we can bound by the radius of the box approximation and by the block-wise difference in bounds.

Proposition 6. *Let* $\widehat{\mathcal{X}} = \bigtimes_j \mathcal{X}_j \in \mathcal{C}_n, \widehat{\mathcal{G}} = \bigtimes_j \mathcal{G}_j \in \mathcal{C}_n$ *be nonempty sets with identical block structure and let* r^∞ *be the radius of the box approximation of* $\mathrm{CH}(\widehat{\mathcal{X}} \cup \widehat{\mathcal{G}})$*. Then*

$$\begin{aligned} & d_H^p(\mathrm{CH}(\widehat{\mathcal{X}} \cup \widehat{\mathcal{G}}), \bigtimes_j \mathrm{CH}(\mathcal{X}_j \cup \mathcal{G}_j)) \\ \leq\ & \min\left(\|r^\infty\|_\infty, \max_{\|d\|_p \leq 1} \textstyle\sum_j |\rho_{\mathcal{X}_j}(d_j) - \rho_{\mathcal{G}_j}(d_j)| \right). \end{aligned}$$

Example Figure 3.4 shows the decomposed convex hull of the sets $\widehat{\mathcal{X}}(3) \cap \mathcal{G}$ and $\widehat{\mathcal{X}}(4) \cap \mathcal{G}$ after applying the translation. Since each block is one-dimensional in our example, we obtain the box approximation.

3.3.6 Discussion

The combination of the decomposed set operations outlined above results in a reachability algorithm that is sound. This immediately follows from the fact that all operations compute an overapproximation.

Proposition 7 (Soundness)**.** *The analysis using Algorithm 1 and the aforementioned procedures to apply the decomposed set operations (intersection, affine map, inclusion check, and convex hull) results in an overapproximation of the reach set.*

When using the above algorithm, the question about the choice of the decomposition arises. Unfortunately, there is generally no automatic way to find the "best" decomposition.

Regarding LTI systems, different decompositions generally result in different flowpipes [52]. Unless the LTI system consists of fully decoupled sub-components (which is usually not the case), the only "best" decomposition is the degenerate case of a single big block.

Regarding hybrid systems given a fixed decomposition of the LTI systems, there is indeed a unique "best" decomposition: a single block for all constrained dimensions. Finding this "best" decomposition is trivial. However, this block may be too big in terms of computational cost.

We can generally say that a finer partition in the decomposition (i.e., using more and smaller blocks) only ever degrades precision. However, while lower-dimensional set operations have lower computational cost, a decrease in precision may actually affect the rest of the algorithm in a negative way. For instance, we may not be able to prove that a transition cannot be taken. Thus statements about computational cost are generally not possible in the case of hybrid systems.

In the next section we shall see that in practical use cases the decomposition is always beneficial and that using the block size of the constrained dimensions is a good heuristic.

Our reachability algorithm benefits greatly from a low number of constrained dimensions, especially in high-dimensional models. High-dimensional models are scarcely available, but those models that we found supported the hypothesis that this number is typically low. In particular, the ARCH-COMP competition [84] represents the state of the art in reachability analysis and considers a scalable model of a powertrain from [16] where only one dimension is constrained. In the next section we consider another scalable model with just three constrained dimensions.

3.4 Evaluation

We implemented the ideas presented in Section 3.3 in JuliaReach [2, 53]. JuliaReach comprises two main software libraries: LazySets.jl and Reachability.jl. The former addresses set-based computations, while the latter contains set-based algorithms for reachability. The decomposed operations presented in Section 3.3 is a new addition to LazySets.jl for the purpose of this evaluation, and it is of independent interest. On the other hand, a

hybrid reachability scheme using decomposed sets in the discrete post-operator has been added to Reachability.jl. All the code is freely available online [1]. We performed the evaluation presented in this section on a Mac notebook with an Intel i5 CPU@3.1 GHz and 16 GB RAM.

3.4.1 Benchmark descriptions

In order to provide a fair and complete comparison, we select commonly used benchmarks in the reachability field from the HyPro models repository [4]. In addition, we evaluate our implementation on a number of models taken from the ARCH-COMP 2019 competition [21], which is a friendly competition and provides a landscape of the current capabilities of verification tools. Finally, we benchmark a scalable model from [84] to compare results for systems with up to 1024 dimensions. To demonstrate the qualitative performance of our approach, we verify a safety property for each benchmark, which requires precise approximations in each step of the algorithm. We briefly describe the benchmarks below.

Linear switching system. This five-dimensional model taken from [4] is a linear hybrid system with five locations of different controlled, randomly-generated continuous dynamics stabilized by an LQR controller. The discrete structure has a ring topology with transitions determined heuristically from simulations. The safety property for this system is $x_1 > -1.2$.

Spacecraft rendezvous. This model with five dimensions represents a spacecraft steering toward a passive target in space [59]. We use a linearized version of this model with three locations. We consider two scenarios, one where the spacecraft successfully approaches the target and another one where a mission abort occurs at $t = 120\,\text{min}$. For the safety properties we refer to [21].

Platooning. This ten-dimensional model with two locations represents a platoon of three vehicles with communication loss at deterministic times [141]. The safety property enforces a minimum distance d between the vehicles: $\bigwedge_{x\in\{x_1,x_4,x_7\}} x \geq -d$. We consider both a time-bounded setting with $d = 42$ and a time-unbounded setting with $d = 50$; note that in the latter setting a fixpoint must be found.

Filtered oscillator. This scalable model consists of a two-dimensional switched oscillator (dimensions x and y) and a parametric number of filters (here: 64–1024) which smooth x [84]. We fixed the maximum number of transitions to five by adding a new variable. The safety property is $y < 0.5$.

3.4.2 Tool descriptions

We compare our implementation in JuliaReach to two other algorithms available in the same tool. All three algorithms use the same decomposition-based continuous-post operator $Post_C^{\square}$ [53], so the main difference between these algorithms is the intersection operation with discrete jumps, which allows for a direct comparison of the approach presented in this work. The existing implementations can be seen as instances of the algorithm in Section 3.2. Furthermore, we compare the implementation to two algorithms available in

Table 3.1. For each benchmark we report the number of dimensions and the number of constrained dimensions (*Dim. (constr.)*), the step size of the time discretization (*Step*) and the run time of the different algorithms as described in Section 3.4.2 (e.g., "Deco" refers to the decomposed algorithm presented in this chapter) in seconds (where the fastest solution is marked in bold face). For benchmark instances with parenthesized computation time, the safety property could not be proven because the overapproximation was too coarse, in which case the computation terminated as soon as the violation was detected. "TO" and "OOM" indicate a timeout of 5×10^4 seconds and an out-of-memory error, respectively.

Benchmark	Dim. (constr.)	Step	Deco	LazySupp	LazyOptim	SpaceEx (LGG)	SpaceEx (STC)
linear (switching)	5 (1)	0.0001	$\mathbf{2.50 \times 10^0}$	1.27×10^1	2.81×10^1	2.60×10^1	2.30×10^1
spacecraft (noabort)	5 (4)	0.04	5.30×10^0	3.42×10^0	2.19×10^2	1.18×10^0	$\mathbf{3.50 \times 10^{-1}}$
spacecraft (120)	5 (5)	0.04	5.30×10^0	2.10×10^0	4.30×10^1	1.91×10^0	$\mathbf{8.10 \times 10^{-1}}$
platoon (bounded)	10 (4)	0.01	$\mathbf{1.30 \times 10^{-1}}$	1.60×10^{-1}	5.69×10^0	5.55×10^0	1.60×10^0
platoon (unbounded)	10 (4)	0.03	$\mathbf{1.08 \times 10^0}$	1.16×10^0	4.96×10^1	3.46×10^1	6.50×10^1
filtered (osc64)	67 (3)	0.01	$\mathbf{2.81 \times 10^0}$	$(7.43 \times 10^0)^\dagger$	5.63×10^2	2.04×10^1	3.25×10^1
filtered (osc128)	131 (3)	0.01	$\mathbf{7.95 \times 10^0}$	$(4.29 \times 10^1)^\dagger$	1.79×10^3	1.69×10^2	4.67×10^3
filtered (osc256)	259 (3)	0.01	$\mathbf{2.80 \times 10^1}$	$(9.19 \times 10^1)^\dagger$	9.99×10^4	8.70×10^3	OOM
filtered (osc512)	515 (3)	0.01	$\mathbf{1.13 \times 10^2}$	$(4.73 \times 10^2)^\dagger$	TO	TO	TO
filtered (osc1024)	1027 (3)	0.01	$\mathbf{5.09 \times 10^2}$	$(5.11 \times 10^3)^\dagger$	TO	TO	TO

†The safety property could not be proven by this tool because the overapproximation was too coarse.

SpaceEx [84], which is an efficient and mature verification tool for linear hybrid systems. We summarize the different approaches below.

Deco. This algorithm implements the decomposed approach presented in this thesis. To compute the (low-dimensional) intersections, we use a polyhedra library [139].

LazySupp. This algorithm uses a (lazy) support-function-based approximation of the intersection operation using the simple heuristic $\rho_{X \cap Y}(\ell) \leq \min(\rho_X(\ell), \rho_Y(\ell))$. This heuristic is fast but not precise enough to verify the safety property of the *filtered oscillator* model.

LazyOptim. In contrast to the coarse intersection in LazySupp, this algorithm uses a more precise implementation of the intersection operation based on line search [83].

SpaceEx LGG. This is an efficient implementation of the algorithm by Le Guernic and Girard [101]. The algorithm uses a support-function representation and can hence check

Table 3.2. Evaluation for different time steps on the *filtered oscillator* model with 64 filters ("filtered_osc64" in Table 3.1) and all algorithms (see Section 3.4.2) that allow varying the time step. The last row shows the relative change of the fourth row over the first row as base line.

Step	Deco	LazySupp	LazyOptim	SpaceEx LGG
0.01	2.8×10^0	7.4×10^0	5.6×10^2	2.0×10^1
0.005	4.1×10^0	1.3×10^1	9.5×10^2	3.6×10^1
0.001	1.7×10^1	6.5×10^1	4.7×10^3	1.6×10^2
0.0005	3.2×10^1	1.3×10^2	8.4×10^3	3.2×10^2
×20.0	×11.4	×17.5	×15.0	×16.0

low-dimensional conditions efficiently, but operations such as intersections work in high dimensions.

SpaceEx STC. The STC algorithm is an extension of the LGG algorithm with automatic time-step adaptation [85].

All tools use a support-function representation of sets. We use template polyhedra with box constraints to overapproximate these sets, which roughly corresponds to a decomposition into one-dimensional blocks. This approximation is fast.

For the algorithms implemented in JuliaReach, we use one-dimensional block structures for all models. For SpaceEx we use the options given in the benchmark sources. For algorithms that use a fixed time step, we fix this parameter to the same value for each benchmark. The SpaceEx STC algorithm does not have such a parameter, so we instead fix the parameter "flowpipe tolerance," which controls the approximation error, to the following values: *Linear switching system*: 0.01, *Spacecraft rendezvous*: 0.2, *Platooning*: 1, *Filtered oscillator*: 0.05.

3.4.3 Evaluation results

The results are presented in Table 3.1. We generally observe a performance boost of the Deco algorithm for models with more than five dimensions, and only a minor overhead for "small" models. This demonstrates the general scalability improvement by performing operations, especially the intersection, in low dimensions. Since all models have a small number of constrained dimensions in their guards and invariants, choosing an appropriate block structure results in very low-dimensional sets for computing the intersections, for which concrete polyhedral computations are very efficient and most precise. We note that such an intersection computation does not scale with the dimension, and so other algorithms must resort to approximation techniques.

Moreover, we found that our approach scales more favorably compared to the high-dimensional approaches when decreasing the time step δ (cf. Section 3.2.1). We demonstrate this observation for the *filtered oscillator* model in Table 3.2 and explain it as follows. Recall that we only compute those sets in the flowpipe in high dimensions for which we have detected an intersection in low dimensions. With a smaller time step, the total

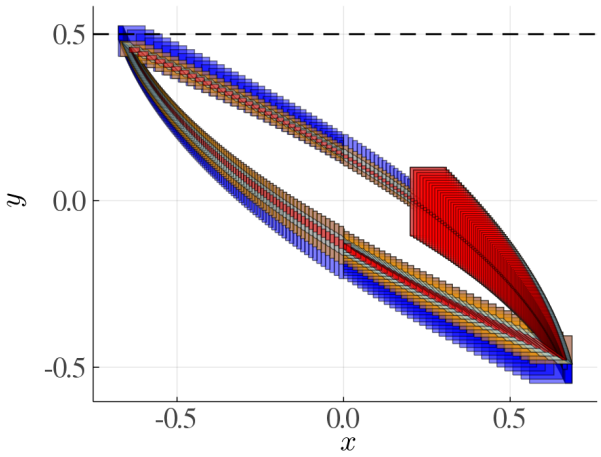


Fig. 3.8. Four flowpipes for the model *filtered oscillator* using a time step of 0.01. The dark blue flowpipe (low-dimensional intersection algorithm) and the orange flowpipe (medium-dimensional intersection algorithm) are obtained for a one-dimensional block structure. The light blue flowpipe is obtained for a two-dimensional block structure and octagon approximation. The red flowpipe is obtained by the tool SpaceEx (which uses high-dimensional analysis). The dashed line corresponds to the safety property $y < 0.5$.

number of sets increases and hence the savings due to our approach become more dominant. To give an example, for the *filtered oscillator* with time step 0.0005, out of the 9,661 sets in total we only computed 1,400 sets in high dimensions. Consequently, making the time step 20 times smaller only makes the run time 11.4 times slower for our algorithm as opposed to at least 15 times slower for other algorithms.

Furthermore, while in general our algorithm may induce an additional approximation error with the block structure, in the evaluation it was always precise enough to prove the safety properties. In Table 3.1 we report the total amount of constrained dimensions in guards, invariants, and safety properties for each benchmark instance. These are the dimensions that determine our low-dimensional flowpipes. For some of the models, the constrained dimensions differ between the invariants/guards and the safety property, but a one-dimensional block structure was still sufficient. Note that we need to compute intersections only with invariants and guards; for safety properties we just need to check inclusion.

In the *linear switching* model, only one dimension is constrained, so our algorithm, together with one-dimensional block structure, is appropriate, especially because a small time step is required to prove the property. In the *platooning* model, the safety property constrains three dimensions but the invariants and guards constrain just one dimension.

The invariants and guards of the models *spacecraft* and *filtered oscillator* constrain two dimensions, so the natural choice for the decomposition is to keep these dimensions in the same block. However, we chose to decompose into one-dimensional blocks for better run-time comparison in Table 3.1. In the implementation, we follow the strategy to perform the intersection in two dimensions and then project back to one dimension (cf. Section 3.3.2). If instead we decide to employ a two-dimensional block structure, we can further gain precision by using different template polyhedra, e.g., with octagon constraints.

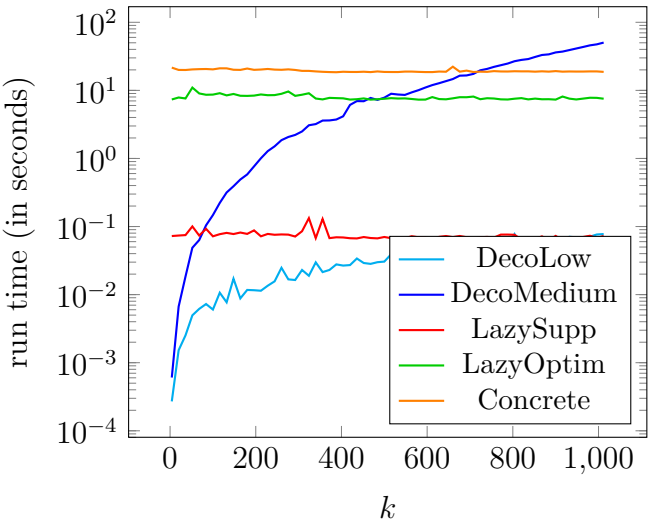


Fig. 3.9. Scaling the number of constrained dimensions k for an intersection $\mathcal{X} \cap \mathcal{G}$ where $\mathcal{X}$ is a hypercube and $\mathcal{G}$ is a half-space.

In Figure 3.8 we visualize this alternative for the *filtered oscillator*. As expected, the one-dimensional analysis is less precise as it corresponds to a box approximation. However, we note that a one-dimensional block structure also inherently reduces the precision of $Post_C^{\square}$, so the additional approximation error does not only stem from the handling of discrete transitions by to our approach. In addition, one can observe that the approximation using a one-dimensional block structure with low-dimensional intersection is coarser compared with the medium-dimensional strategy and we cannot prove the safety property anymore.

3.4.4 Scaling the number of constrained dimensions

We now investigate the effect of increasing the number of constrained dimensions in two tests.

In the first evaluation we focus on the intersection operation. We fix a hypercube $\mathcal{X}$ of dimension 1,024, of radius 4 and centered in the origin, and a half-space $\mathcal{G}$ $x_1 + \cdots + x_k \leq 2$ for parameter k that controls the number of constrained dimensions. The half-space properly cuts the hypercube for any k. In Figure 3.9 we show the run times for different intersection algorithms. We note that these algorithms compute different approximations of the true intersection: the "DecoLow" and "DecoMedium" algorithms implement the respective ideas from Figure 3.6; the "LazySupp" and "LazyOptim" algorithms use the ideas as in the reachability algorithms of the same name (described in Section 3.4.2); the "Concrete" algorithm uses a polyhedra library (note that the H-representation of a hypercube of dimension n has only $2n$ constraints). We can see that the high-dimensional intersection algorithms are not affected by the number of constrained dimensions k whereas the decomposition-based algorithms scale gracefully with k.

In the second evaluation we consider the full reachability setting. For that we modify the *filtered oscillator* benchmark with 128 filters (*filtered_osc128* in Table 3.1) by adding small nonzero entries to k previously unconstrained dimensions in the invariants and guards. We consider all the reachability algorithms that are precise enough to verify the

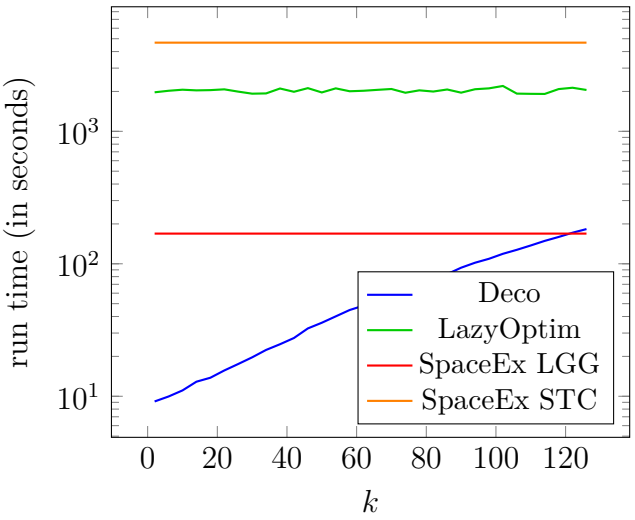


Fig. 3.10. Scaling the number of constrained dimensions k for the modified *filtered_osc128* benchmark.

safety property (which is still satisfied in our modified benchmark instances), i.e., all the algorithms from Table 3.1 except for *LazySupp*. Note that for SpaceEx algorithms we extrapolate the results we report for this benchmark instance in Table 3.1 to a range of values of parameter k as the reachability algorithms implemented in SpaceEx do not depend on the number of constrained dimensions. The run times are plotted in Figure 3.10 as a function of k. Again the high-dimensional algorithms are not affected by the number k whereas the decomposition-based algorithm scales well with k (note the log scale). For $k \approx 120$ (close to $n = 131$) we see a cross-over with the *SpaceEx LGG* run time, which is expected due to the overhead of the decomposition.

3.5 Summary

We have presented a schema that integrates a reachability algorithm based on decomposition for LTI systems in the analysis loop for linear hybrid systems. The key insight is that intersections with polyhedral constraints can be efficiently detected and computed (approximately or often even exactly) in low dimensions. This enables the systematic focus on appropriate subspaces and the potential for bypassing large amounts of flowpipe computations. Moreover, working with sets in low dimensions allows to use precise polyhedral computations that are infeasible in high dimensions. An essential step in our algorithm is the fast computation of a low-dimensional flowpipe for the detection of intersections.

In the presented algorithm, we recompute the flowpipe for the relevant time frames in high dimensions using the same decomposed algorithm with the same time step. However, this is not necessary. We could achieve higher precision by using different algorithmic parameters or even a different, possibly non-decomposed, algorithm (e.g., one that features arbitrary precision [97]). This is particularly promising for LTI systems because one can avoid recomputing the homogeneous (state-based) part of the flowpipe [83].

The algorithm will also benefit from a refinement loop to rearrange the block structure. Currently, we merge different blocks to maintain the precision for hybrid systems where different locations have different contrain dimensions. This results into longer computational time or splitting into original blocks which comes with additional approximation error. Thus, creating heuristics to automatically maintain and change structure between different locations could improve the overall performance of the algorithm.

Chapter 4

Reachability of Linearized Systems and Zonotope Refinements

Reachability analysis for nonlinear systems is challenging, and is often the bottleneck for formal cyber-physical systems (CPS) analysis methods. Despite recent advances, systems described by nonlinear ordinary differential equations are still hard to analyze, control, and verify. On the other hand, a powerful body of methods and theories exists for linear systems making analysis, control, and verification much easier, even for high-dimensional systems.

4.1 Chapter overview and structure

In this work we investigate nonlinear reachability approaches based on *Koopman operator linearization* [144]. Koopman operator linearization is a process where a nonlinear system can be approximated as a linear system with a large number of so-called observable variables, each of which can be a nonlinear function of the original state variables. Koopman operator techniques are also well-suited for data-driven approaches since the Koopman linearized system can be directly created from measurements, bypassing a potentially complex modeling step. The Koopman framework has been successfully applied to many applications, including control [143, 156] and state estimation [170]. This linear system can be computed either symbolically from differential equations or—importantly for black-box systems—from data derived from real-world system executions or simulations [132]. The efficiency of techniques related to reachability analysis for linear systems [94, 52, 36] motivates the use of Koopman operator linearization. However, directly applying existing reachability algorithms is not possible, as Koopman operator linearization approximates nonlinear systems of differential equations with higher-dimensional linear systems. For formal verification using reachability analysis, this is an attractive conversion, as highly scalable methods exist to compute reachable sets for linear systems. However, linear reachability analysis methods must be modified to support nonlinear initial state sets.

In this chapter, we present several algorithms to perform reachability analysis algorithms on Koopman linearized systems. We first show the problem can be solved using a nonlinear

satisfiability modulo theory (SMT) solver to enforce the initial state constraints. This Direct Encoding Algorithm is correct but may be slow in practice and even undecidable in theory. We improve analysis efficiency through zonotope overapproximations of the nonlinear initial sets constructed using interval arithmetic, as well as two abstraction-refinement techniques: (i) Hyperplane Backpropagation and (ii) Zonotope Domain Splitting. Then, we propose combining Taylor models with polynomial zonotope refinement to reason over non-convex initial sets.

The approximation must be also sufficiently accurate for the result to be meaningful, which is controlled by the choice of *observable functions* during Koopman operator linearization. *Ad hoc* approach to choose appropriate observable variables can be time-consuming. In addition, it is difficult to reason about accuracy of such approximation. Random Fourier [165] features can be used as observable functions, to make the process more systematic and provide a higher-accuracy approximation. In contrast with an *ad hoc*, finite-dimensional feature space, random Fourier features leverage the powerful *kernel trick* from machine learning [176, 184] to generate a computationally tractable mapping over an infinite-dimensional feature space.

We provide a comparison of algorithms on the well-known nonlinear system benchmarks to showcase applicability and efficiency of the presented techniques. This chapter is based on the results published in [37] and [44].

The contributions of this chapter are:

- We propose using Koopman operator to shift from nonlinear dynamics to linear.
- We improve the accuracy of the finite Koopman linearization by employing random Fourier features in Section.
- We demonstrate different approaches to to analyse systems with non-convex initial sets.
  - First, we propose the basic direct encoding algorithm.
  - Secondly, we provide an algorithm based on interval arithmetic and numerous refinements to improve efficiency of the algorithm.
  - Finally, we further improve the efficiency of the algorithm by employing Taylor model arithmetic and polinomial zonotopes to handle non-linear initial sets.

The chapter is structured as follows:

- We first desribe the linerization via Koopman operator in Sec. 4.2.
- Then, a systematic way to generate observables using random Fourier features is presented in Sec. 4.3
- We propose new verification algorithms which work with non-convex initial sets in Sec. 4.4

- We demonstrate the superior performance of using Koopman operator and Random Fourier feature observables along with the adjusted reachability algorithm in comparison with existing techniques on various benchmark systems in Sec. 4.5.

4.2 Koopman Operator Linearization

The modern study of dynamical systems is driven by Poincaré's *state space* view of the underlying system. By considering the evolution of points in a state space, this view enables intuitive tools for analyzing, designing, controlling, and verifying dynamical systems. However, it can be ill-suited for certain classes of problems such as uncertain systems [58, 92] and systems without explicit equations describing their evolution (data-driven or black-box models) [115].

An alternative to this state space view is Koopman's *observable space* view of dynamical systems. In contrast to the state space view, the observable space view considers the evolution of observables, or functions, of the given state space [58] instead of the states themselves. This alternative view leads to the notion of the so-called Koopman operator [127]. For dynamical system $S\colon \Omega \to \Omega$ and observable $g\colon \Omega \to \mathbb{R}$, the Koopman operator $\mathcal{K}$ is defined by

$$\mathcal{K}g = g \circ S, \qquad \forall g \in L^{\infty} \tag{4.1}$$

As such, the Koopman operator is an infinite-dimensional linear operator on the space of scalar-valued functions of the state space [127]. The spectral properties of this linear operator describe the evolutionary properties of the underlying dynamical system S, similar to finite-dimensional linear state space models (*e.g.,* eigen decomposition of a state matrix). However, unlike a finite-dimensional linear state matrix which describes the evolution of system states, the Koopman operator describes the evolution of scalar-valued functions as driven by the dynamics of the underlying system [154].

As Eq. 4.1 is equally valid for linear and nonlinear systems, the observable space view enables the linear treatment of full nonlinear dynamics via the Koopman operator. Thus, the Koopman operator has the potential to bridge nonlinear systems and existing linear tools for analysis, design, control, and verification [56, 132] without sacrificing information, like with traditional linearization techniques [58]. Unfortunately, this theory comes with a practical cost because the Koopman operator is infinite-dimensional. In theory, one can equivalently switch between representations of a dynamical system that are nonlinear finite-dimension (state space) or linear infinite-dimensional (observable space). In other words, one can *lift* the nonlinear dynamics into a higher-, possibly infinite-, dimensional space where its dynamics are linear [129].

For certain classes of systems, it is possible to obtain a finite-dimensional Koopman operator that describes the evolution of a system. In particular, these systems posses Koopman-invariant subspaces containing the system state [56].

For example, consider the following continuous-time dynamical system:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \begin{bmatrix} \mu x_1 \\ \lambda\left(x_2 - x_1^4\right) \end{bmatrix} \tag{4.2}$$

where $x^\top = [x_1, x_2]$. If we consider the observables $g_1(x) = x_1$, $g_2(x) = x_2$, $g_3(x) = x_1^4$, the state can be lifted into a three-dimensional space where the system evolves linearly. Using the relationships in Eq. 4.2, the derivatives of the observables are linear, and can be written as

$$\tfrac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{g}(x) = \underbrace{\begin{bmatrix} \mu & 0 & 0 \\ 0 & \lambda & -\lambda \\ 0 & 0 & 4 \end{bmatrix}}_{\tilde{\mathcal{K}}} \boldsymbol{g}(x) \tag{4.3}$$

where $\boldsymbol{g}^\top = [g_1, g_2, g_3]$ and $\tilde{\mathcal{K}}$ is the infinitesimal Koopman operator [134]. The solution to this linear ordinary differential equation is then

$$\begin{aligned} \boldsymbol{g}(x_t) &= e^{\tilde{\mathcal{K}}t}\boldsymbol{g}(x_0) \\ &= \mathcal{K}_t\boldsymbol{g}(x_0) \end{aligned} \tag{4.4}$$

where $\mathcal{K}_t = e^{\tilde{K}t}$ is the Koopman operator as parameterized by t.

As the system states are contained in the set of observables (g_1 and g_2) the states can be recovered from Eq. 4.4 linearly as

$$x_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathcal{K}_t\boldsymbol{g}(x) \tag{4.5}$$

Unfortunately, identifying the observables that form a Koopman-invariant subspace of a system is a difficult problem and an active area of research. However, simple algorithms exist for computing finite *approximations* of the Koopman operator given time-series data. These algorithms are primarily based on the Dynamic Mode Decomposition (DMD) algorithm [175], which performs regression of data to derive linear dynamics [132].

To fit a discrete-time linear system to data, the DMD algorithm starts by concatenating temporal snapshots of the system states x as columns of two data matrices, X and X'. Given m time instances of an n-dimensional system, these two $n \times m - 1$ matrices are formed as

$$X = \begin{bmatrix} x_1 & x_2 & \dots & x_{m-1} \end{bmatrix} \tag{4.6}$$

$$X' = \begin{bmatrix} x_2 & x_3 & \dots & x_m \end{bmatrix} \tag{4.7}$$

where x_k indicates the k^{th} temporal snapshot of x. The best-fit least squares state transition matrix A such that $X' \approx AX$ is then given by

$$A = X'X^{\dagger} \tag{4.8}$$

where $X^{\dagger}$ is the Moore-Penrose pseudoinverse of X. Because the DMD algorithm assumes linear dynamics, one can naturally extend this algorithm to approximate the linear Koopman operator from data by lifting the states in Eqs. 4.6 and 4.7 using a finite set of observables called a *dictionary*. In other words, given some dictionary of ℓ pre-defined observables, where $\boldsymbol{g}(x) = \left[g_1(x), g_2(x), \ldots, g_\ell(x)\right]^\top$, lift Eqs. 4.6 and 4.7 as

$$G = \begin{bmatrix} \boldsymbol{g}(x_1) & \boldsymbol{g}(x_2) & \ldots & \boldsymbol{g}(x_{m-1}) \end{bmatrix} \tag{4.9}$$

$$G' = \begin{bmatrix} \boldsymbol{g}(x_2) & \boldsymbol{g}(x_3) & \ldots & \boldsymbol{g}(x_m) \end{bmatrix} \tag{4.10}$$

and solve in the same manner as with DMD, leading to an approximation of the Koopman operator [197].

Similarly, one can approximate the infinitesimal Koopman operator by substituting data matrix G' by $\frac{\mathrm{d}}{\mathrm{d}t}G$. Depending on the application at hand, these derivatives may be directly available via rate sensors or be computable when synthetic data is leveraged. Alternatively, finite difference methods may be used.

In practice it may be impractical to solve Eq. 4.8 as presented, as the data matrices can be large and/or the data may be noisy. In such cases, one can adjust the algorithm to use low-rank approximations of the data matrix G. This is typically achieved by truncating the non-dominant modes of the singular value decomposition (SVD) of G. Koopman operator linearization for analysis and control are active areas of research in the applied mathematics community [143, 132].

We strive to use the Koopman observable space view to perform reachability analysis for nonlinear systems.

4.3 Linearization via Fourier Features

We now present the automated generation of observables using random Fourier features [73]. Let us first motivate why Fourier features are a good choice for observables. For Koopman linearization, the observables $g(x)$ define a transformation to a high-dimensional space. One commonly used approach to handle such high-dimensional spaces efficiently is the *kernel trick*. In many algorithms the data points $x, y \in \mathbb{R}^n$ only appear in the form of inner products $g(x)^T g(y)$. In this case it suffices to define a kernel function $k(x, y)$ that represents the similarity measure $g(x)^T g(y)$ between data points in the high-dimensional feature space, rather than explicitly defining a transformation $g(x)$ to this space. Kernel functions can also represent more general features that are not vectors and even infinite dimensional features, which motivates their application in the Koopman framework. The kernel trick is mainly applied for machine learning techniques [176], such as regression [184], clustering [122], and

classification [191]. However, the extended dynamic mode decomposition algorithm [195] can also be formulated in terms of inner-products [196], so that the kernel trick can be applied for Koopman linearization. Rather than explicitly choosing observables $g(x)$ we can therefore select a kernel function instead, which implicitly defines the observable function $g(x)$ through the kernel's relation to an inner product space. Commonly used kernels are radial basis function kernels, polynomial kernels, and spline kernels.

The kernel trick cannot be applied directly to our reachability technique since we require an explicit formulation of the observables $g(x)$. We therefore first select a kernel function $k(x, y)$, and then determine observables $g(x)$ that yield a good approximation of the kernel function $k(x, y) \approx g(x)^T g(y)$. Random Fourier features are a common technique to approximate kernel functions [165, 73]. They are based on Bochner's theorem [172], which links a weakly stationary kernel function to a Fourier transform:

$$k(x, y) = \int_{\mathbb{R}^n} e^{j\,\omega^T (x-y)}\, d\mu(\omega) = \mathbb{E}_\omega \left(e^{j\,\omega^T x}\, \overline{e^{j\,\omega^T y}} \right), \tag{4.11}$$

where the function $\mu : \mathbb{R}^n \to [0, 1]$ defines a probability distribution, $\mathbb{E}_\omega(\cdot)$ denotes the expected value with respect to ω, j is the imaginary unit, and $\overline{a}$ denotes the complex conjugate for a complex number $a \in \mathbb{C}$. The distribution $\mu(\omega)$ associated with a specific kernel can be obtained by taking the inverse Fourier transform of $k(x, y)$ [165]. We can collect m samples from the distribution $\mu(\omega)$ to approximate the expected value in (4.11), which finally yields

$$k(x, y) = \mathbb{E}_\omega \left(e^{j\,\omega^T x}\, \overline{e^{j\,\omega^T y}} \right) \approx \frac{1}{m} \sum_{i=1}^{m} \underbrace{e^{j\,\omega_i^T x}}_{g_i(x)}\, \underbrace{e^{j\,\omega_i^T y}}_{g_i(y)}.$$

The random Fourier features are the resulting observables $g_i(x)$ that approximate the kernel function. Note that we can omit the constant factor $\frac{1}{m}$ since extended dynamic mode decomposition will automatically scale the observables accordingly. We consider real-valued kernels only, so we use Euler's formula $e^{j\,x} = \cos(x) + j\,\sin(x)$ to simplify the random Fourier features to

$$g_i(x) = \sqrt{2}\,\cos\!\left(\omega_i^T x + b_i\right), \quad i = 1, \ldots, m, \tag{4.12}$$

where the shift b_i is selected uniformly from the interval $[0, 2\pi]$ and ω_i is drawn randomly from the probability distribution $\mu(\omega)$ corresponding to the kernel that is used. While this random selection might appear to be a disadvantage at first sight, it is guaranteed that the random Fourier feature approximation converges to the exact kernel function when increasing the number of observables [165]. Moreover, we observed from our tests that changes in the values for b_i and ω_i do not significantly influence the accuracy of the resulting linear approximation.

In summary, the random Fourier features presented above represent a systematic method for selecting a finite set of accurate observables which requires only few hyperparameters. These hyperparameters include the type of kernel that is used, the kernel parameters, and

the number of observables. In this thesis, for the evaluation we use a radial basis function kernel

$$k(x, y) = e^{-\frac{\|x-y\|_2^2}{2\,\ell^2}},$$

which contains the lengthscale ℓ as the only parameter. The probability distribution $\mu(\omega)$ for this kernel is the multivariate normal distribution with covariance matrix $\ell^2 \cdot I_n$ centered at the origin [165].

4.4 Reachability analysis of linear systems with nonlinear initial sets

Koopman operator linearization creates approximations to nonlinear, possibly black-box, systems. The resulting systems have linear dynamics in the space of observables. Reachability analysis of linear systems is a well-studied topic, and existing methods are efficient even with thousands or more state variables [41]. However, the initial and unsafe constraints in the original problem are defined on the original system variables, not on the nonlinear observable variables. This nonlinear relationship creates two additional tasks: (1) projection of initial set from state space variables to the space of observables, and (2) projection of reachable set in the space of observables back into the state space. Problem (2) can be resolved by including the original state variables within the dictionary used during Koopman linearization. This means that the projection from the space of observables to the original state variable can be written as a simple linear transformation, $x = Mg(x)$. Problem (1), however, is more difficult to handle, and the main focus of the rest of this section.

For simplicity we assume that the specification we aim to verify is described by a single unsafe set $\mathcal{X}_u$, but the extension to multiple unsafe sets is straightforward.

4.4.1 Reachability analysis using zonotope refinements

4.4.1.1 Direct Encoding of Nonlinear Constraints with SMT Solvers

Let the state space of the system to be $\boldsymbol{x}$ and the observables be $g(x)$. Furthermore, the evolution of observables is determined by a linear differential equations, $g(x_t) = \mathcal{K}g(x_0)$. The state after time t, denoted as x_t is $Mg(x_t)$. Given an initial set $\mathcal{X}_0$, and unsafe set $\mathcal{X}_u$, the safety verification can be formulated as satisfiability of constraints in Equation 4.13.

$$\begin{aligned} x_0 \in \mathcal{X}_0, y = g(x_0), \\ y_t = \mathcal{K}_t y, \\ x_t = My_t, x_t \in \mathcal{X}_u. \end{aligned} \tag{4.13}$$

Given step size $h > 0$, for each time instant $t = i \times h$, an SMT solver can be invoked with the constraints in Equation 4.13. Often, the initial set $\mathcal{X}_0$ and unsafe set $\mathcal{X}_u$ are specified

as conjunctions of linear constraints. For such cases, observe that the only nonlinear constraint in Equation 4.13 is $y = g(x_0)$. The complexity of finding an assignment of variables that satisfies these linear constaints depend on the number and functions used for the observables.

For example, if observables in the dictionary are polynomials, SMT solvers can use algorithms like cylindrical algebraic decomposition [27] that are theoretically guaranteed to terminate with a correct result.

In practice, this algorithm is doubly exponential in the number of variables, so a result may not be produced in a reasonable time.

If the dictionary includes transcendental functions like sin or cos, the problem in general may not even be decidable.

In our implementation, we use the δ-decidability SMT solver dReal [75, 55, 90], which theoretically always terminates but may produce an unknown result if the constraints are on the boundary of satisfiability (within a tolerance δ), which we can then still flag as potentially unsafe.

Although the direct encoding works, for the reasons stated above it can be slow, so we next focus on optimizations and efficiency improvements.

4.4.1.2 Overapproximating Nonlinear Constraints with Intervals

Suppose that the initial set $\mathcal{X}_0$ is given as hyperrectangles, as is often the case. That is, $\mathcal{I} = [\underline{\mathcal{X}_0^1}; \overline{\mathcal{X}_0^1}] \times \ldots \times [\underline{\mathcal{X}_0^n}; \overline{\mathcal{X}_0^n}]$. Over such a domain, it is possible to compute conservative approximation of $y = g(\mathcal{X}_0)$ where $\mathcal{X}_0 \in \mathcal{I}$ using interval arithmetic. Alternatively, when $\mathcal{X}_0$ is defined with linear constraints, we can compute upper and lower bounds on each of the variables using linear programming (LP) to construct box bounds, and then use those to construct the conservative approximation of $y = g(\mathcal{X}_0)$. Let the bounds we obtain on y be such that $y \in [\underline{y^1}, \overline{y^1}] \times \ldots \times [\underline{y^k}, \overline{y^k}]$. Substituting these in Equation 4.13 results in the following constraints.

$$\begin{aligned} y \in [\underline{y^1}^1, \overline{y^1}] \times \ldots \times [\underline{y^1}^k, \overline{y^1}^k], \\ y_t = \mathcal{K}_t y, \\ x_t = M y_t, x_t \in \mathcal{X}_u. \end{aligned} \tag{4.14}$$

This set of constraints can be solved efficiently using LP, as is done in linear systems reachability analysis using zonotopes [94] or linear star sets [35].

While this method can be very efficient, the overapproximation of $g(x)$ using interval arithmetic might yield a very coarse overapproximation. This would mean that spurious unsafe executions of the system may be found, due to the overapproximation. To overcome this the *Interval Encoding Algorithm* uses a hybrid approach, where an LP is first solved at each step according to Equation 4.14, and only if the LP is feasible will we then call the dReal SMT solver with the nonlinear constraints from Equation 4.13.

4.4.1.3 Hyperplane Backpropagation

One of the building blocks that helps us improve efficiency is the notion of **hyperplane backpropagation**. Consider the unsafe set given as $\mathcal{X}_u = [\underline{\mathcal{X}_u^1}; \overline{\mathcal{X}_u^1}] \times \ldots \times [\underline{\mathcal{X}_u^n}; \overline{\mathcal{X}_u^n}]$, observables $g(x)$, and $x = Mg(x)$. Since we assumed the Koopman dictionary contained the original state variables, we can directly encode $\mathcal{X}_u$ as constraints in the observable space, which we write as $g(\mathcal{X}_u)$. If $q^T y \leq r$ is a halfplane constraint in $g(\mathcal{X}_u)$, then the corresponding constraint for propagating this constraint back by time t is obtained by $(\mathcal{K}^T q)^T y \leq r$. We perform hyperplane backpropagation for all the constraints in $g(\mathcal{X}_u)$. In the Koopman linerization literature, this operation is called *pull-back* operation [148]. Any state that satisfies all the constraints obtained by performing the hyperplane backpropagation will end up in $g(\mathcal{X}_u)$ after time t. We label these constraints as $\mathsf{PropCons}(\mathcal{X}_u)$. While we have explicitly specified this only for unsafe sets that are intervals, this can be easily extended to unsafe sets specified as conjunction of half-spaces. In the remainder of this section, we present two main enhancements, namely, *domain contraction*, and *interval refinement* that use constraint propagation.

We use the above process to backpropagate each of the unsafe set constraints into the initial set, i.e., $\mathsf{PropCons}(\mathcal{X}_u)$. We then compute the overlap between the projection of initial set into the observables ($g(\mathcal{X}_0)$) and compute its overlap with propagated constraints, i.e., $g(\mathcal{X}_0) \cap \mathsf{PropCons}(\mathcal{X}_u)$, using interval arithmetic. We then project this set back into the initial set as $M(g(\mathcal{X}_0) \cap \mathsf{PropCons}(U))$. If this projected set $\mathcal{I}'$ is a strict subset of initial set $\mathcal{I}$, then, one can specify the possible violation of safety with higher precision. We then repeat the process with the new set $\mathcal{I}'$. If, during this process $\mathcal{I}'$ is empty, then, none of the trajectories go into the unsafe set; we can declare the system to be safe. If $\mathcal{I}'$ is same as the set $\mathcal{I}$, then this approach does not further improve precision. In this case, the *Hyperplane Backpropagation Algorithm* would invoke dReal and instantiate the constraints given in Equation 4.13 with $\mathcal{I}'$ instead of $\mathcal{I}$. The pseudocode of the core step of the algorithm is provided in Algorithm 2.

Algorithm 2 Hyperplane Backpropagation Algorithm

function BACKPROPAGATE($\mathcal{X}_0, \mathcal{X}_u, t, x, g(x), \mathcal{K}$)
  Output: A smaller initial set $\mathcal{X}_0'$ after backpropagation, or $\emptyset$ if safe.
  $\mathcal{X}_{uo} = g(\mathcal{X}_u)$
  $\mathsf{PropCons} = \mathcal{K}^{-1}\mathcal{X}_{uo}$
  $\mathcal{I}_o = g(\mathcal{X}_0)$
  $\mathcal{I}' = M(\mathcal{I}_o \cap \mathsf{PropCons})$
  while $\mathcal{I}' \subset \mathcal{I}$ **do**
    if $\mathcal{I}' = \emptyset$ **then**
      return `safe`.
    end if
    $\mathcal{I} = g(\mathcal{I}')$
    $I' = M(\mathcal{I}_o \cap \mathsf{PropCons})$
  end while
  return $\mathcal{I}'$
end function

4.4.1.4 Zonotope Domain Splitting

The reachable states at each time step encoded by Equation 4.14 can be represented using a zonotope (see Definition 8), where the box domain of the zonotope is the initial set $[\underline{y^1}, \overline{y^1}] \times \ldots \times [\underline{y^k}, \overline{y^k}]$. The accuracy of this zonotope overapproximation can be improved by splitting each dimension's interval domain into several smaller subintervals and computing the new interval overapproximation in each of the subintervals.

For the proposed *Zonotope Domain Splitting Algorithm*, we use a heuristic that always splits the variable with the maximum range. After splitting, we then perform hyperplane backpropagation on each of the smaller intervals. This process can be repeated until either the overapproximation is safe for the current step, or some predetermined upperLimit is reached on the number of splits. Upon reaching the limit, we then invoke dReal using the nonlinear constraints from Equation 4.13. As a result, dReal is invoked with smaller initial sets, thus helping the numerical procedure to terminate faster. However, as we show later in the evaluation, performing refinement too often can harm the performance of the verification procedure, as splitting can increase the number of queries needed. The pseudocode for the core of the zonotope domain splitting algorithm is given in Algorithm 3

Algorithm 3 Zonotope Domain Splitting Algorithm

function ZonoSplitting($\mathcal{X}_0, \mathcal{X}_u, x, g(x), \mathcal{K}_t, level$)
  if $level \leq max\_level$ **then**
    $\mathcal{X}_{0l}, \mathcal{X}_{0r} = split(\mathcal{X}_0)$
    $\mathcal{X}'_{0l} = Backpropagate(\mathcal{X}_{0l}, \mathcal{X}_u, x, g(x), \mathcal{K}_t)$
    $\mathcal{X}'_{0r} = Backpropagate(\mathcal{X}_{0r}, \mathcal{X}_u, x, g(x), \mathcal{K}_t)$
    `safe1` $= (\mathcal{X}'_{0l} = \emptyset)$ **or** ZonoSplitting$(\mathcal{X}'_{0l}, \mathcal{X}_u, x, g(x), \mathcal{K}_t, level + 1)$
    `safe2` $= (\mathcal{X}'_{0r} = \emptyset)$ **or** ZonoSplitting$(\mathcal{X}'_{0r}, \mathcal{X}_u, x, g(x), \mathcal{K}_t, level + 1)$
    if `safe1` **and** `safe2` **then**
      return `UNSAT`
    else
      return `SAT`
    end if
  else
    return dReal$(\mathcal{X}_0, \mathcal{X}_u, x, g(x), \mathcal{K}_t)$
  end if
end function

Notice that the order of applying hyperplane backpropagation and zonotope domain contraction can alter the performance of the verification procedure. While we prefer performing hyperplane backpropagation at each iteration, delaying the process until the interval under consideration becomes smaller could be a useful heuristic for some examples. In our evaluation, we consider various possible combinations of these two methods. In our experience, invoking dReal with the smaller initial sets succeeds to either prove safety or generate counterexample quicker than larger initial sets.

Whole algorithm Finally, we combine all the ideas presented above into one full algorithm. First, we apply back refinement. Then, once we cannot further refine we call

Algorithm 4 Verification of Koopman linearized systems

Require: Koopman linearized system $\dot{g}(x) = A\,g(x)$, initial set $\mathcal{X}_0$, final time t_F, specification given as an unsafe set $\mathcal{X}_u$, time step size Δt, initial Taylor order κ_0.
Ensure: System is safe (res $= \top$) or unsafe (res $= \bot$).

```
1: res ←⊥, κ ← κ_0                                                    (initialization)
2: repeat
3:     T(x) ← {g(x) | x ∈ X_0}        (comp. using Taylor model arithmetic with order κ)
4:     PZ ← T(x)                  (convert Taylor model to polynomial zonotope, see [124])
5:     R_0, ..., R_{t_F/Δt} ← reachability analysis of ġ(x) = A g(x) for initial set PZ
6:     L ← (R_0, ..., R_{t_F/Δt})                  (initialize queue of not yet verified sets)
7:     repeat
8:         PZ ← L_(1),   L ← (L_(2), ..., L_(|L|))                  (pop first element from queue)
9:         Z ← zonotope enclosure of PZ                                              (see [124])
10:        if Z ∩ X_u ≠ ∅ then    (check if specification is satisfied, see (4.15) and (4.16))
11:            x_0, t ← most critical initial state and corresponding time
12:            if [I_n 0] e^{At} g(x_0) ∈ X_u then
13:                return                (specification falsified ⇒ system is unsafe)
14:            else
15:                PZ_1, PZ_2 ← split PZ                                  (see Prop. 8 and (4.19))
16:                L ← (L, PZ_1, PZ_2)                                    (add new sets to queue)
17:            end if
18:        end if
19:    until L = ( ) or splitting does not yield any further improvement
20:    κ ← κ + 1                                                                (increase Taylor order)
21: until L = ( )                                  (queue empty ⇒ no intersection with X_u)
22: res ← ⊤          (if this line is reached no reach. set intersects X_u ⇒ system is safe)
```

binary refinement up to defined *max_level*. If even after binary refinement we cannot verify whether the system is safe, we call the nonlinear SMT solver (dReal in our case).

4.4.2 Reachability analysis using polynomial zonotope refinement

We now present a polynomial zonotope based verification algorithm for Koopman linearized systems, which is summarized in Alg. 4. We first apply Taylor model arithmetic (see Definition 9) to compute a tight non-convex enclosure for the image of the initial set $\mathcal{X}_0$ through the observable function $g(x)$ in Line 3. Since it simplifies the computation of the zonotope enclosures required later on, we then convert the resulting Taylor model to a polynomial zonotope in Line 4. This polynomial zonotope is used as the initial set for the computation of the reachable set for the Koopman linearized system as performed in Line 5, for which we can use any reachability algorithm for linear systems. For simplicity we assume here that the obtained reachable sets are exact. In the general case where the exact reachable set cannot be computed one can for example incorporate the error measures from [84] and [194] into the verification algorithm.

The problem we are facing now is that the reachable sets $\mathcal{R}_0, \ldots, \mathcal{R}_{t_F/\Delta t}$ are represented by polynomial zonotopes, a set representation for which exact collision checks with the

unsafe set $\mathcal{X}_u$ are computationally demanding. We resolve this issue by applying a novel polynomial zonotope refinement procedure in lines 6-19, where we recursively split the polynomial zonotopes until we can either verify or falsify the specification using zonotope enclosures of the split sets. In particular, we first enclose each polynomial zonotope in the queue **L** with a zonotope in Line 9. For a zonotope $\mathcal{Z} = \langle c, G \rangle_Z$ collision checks with an unsafe set as performed in Line 10 are very efficient: If the unsafe set is a halfspace $\mathcal{X}_u = \langle h, d \rangle_H$, we have according to [94]

$$(\mathcal{Z} \cap \mathcal{X}_u \neq \emptyset) \Leftrightarrow \left(h^T c - \sum_{i=1}^{p} |h^T G_{(\cdot,i)})| \leq d \right) \tag{4.15}$$

For general polytopes $\mathcal{X}_u = \langle H, d \rangle_P$ collision checks can be realized using linear programming:

$$(\mathcal{Z} \cap \mathcal{X}_u \neq \emptyset) \Leftrightarrow (\delta = 0), \tag{4.16}$$

where

$$\delta = \min_{\alpha, x} \|c + G\alpha - x\|_1 \;\text{ s.t. }\; \alpha \in [-\mathbf{1}, \mathbf{1}],\; Hx \leq d. \tag{4.17}$$

If the specification cannot be verified, we next try to falsify it in lines 11-13 by extracting the initial point x_0 that is expected to violate the specification the most from $\mathcal{Z}$. For a halfspace $\mathcal{X}_u = \langle h, d \rangle_H$ the vector of zonotope factors $\alpha = [\alpha_1 \;\ldots\; \alpha_p]^T$ resulting in the largest violation is given as $\alpha = -\text{sign}(h^T G)$, where the signum function is interpreted elementwise. Since the factors α of the zonotope enclosure are related to the dependent factors of the original polynomial zonotope and since polynomial zonotopes preserve dependencies during reachability analysis [125], we can then directly extract the initial point x_0 corresponding to α from the polynomial zonotope. For general polytopes we can use the optimal α from the linear program in (4.17) to estimate the most critical initial point. If we can neither verify nor falsify the specification we have a so called spurious counterexample that arises due to the over-approximation introduced by the zonotope enclosure. We therefore split the polynomial zonotope in this case in Line 15 since splitting reduces the over-approximation in the zonotope enclosure (see Fig. 4.1). The split sets are then added to the queue in Line 16, where we use a first-in, first-out scheme for the queue to detect easy falsifications fast before excessively splitting the sets.

One remaining issue we are facing is that Taylor model arithmetic is not exact. Due to the over-approximation in the initial set it can therefore happen that we can neither verify nor falsify the specification by splitting the polynomial zonotope. To solve this issue we embed our whole algorithm into a repeat-until-loop that iteratively increases the order κ used for Taylor model arithmetic (see Line 20). Since Taylor model arithmetic converges to the exact result if the order goes to infinity, we obtain a complete algorithm that is guaranteed to terminate. In practice we can often prevent computational expensive iterations of the outer loop by choosing the initial order κ_0 large enough. It remains to decide when to stop splitting the polynomial zonotopes and increase the Taylor order instead (see Line 19). The simplest method is to just use an upper bound for the number of recursive splits that are performed. A more sophisticated approach is to abort splitting

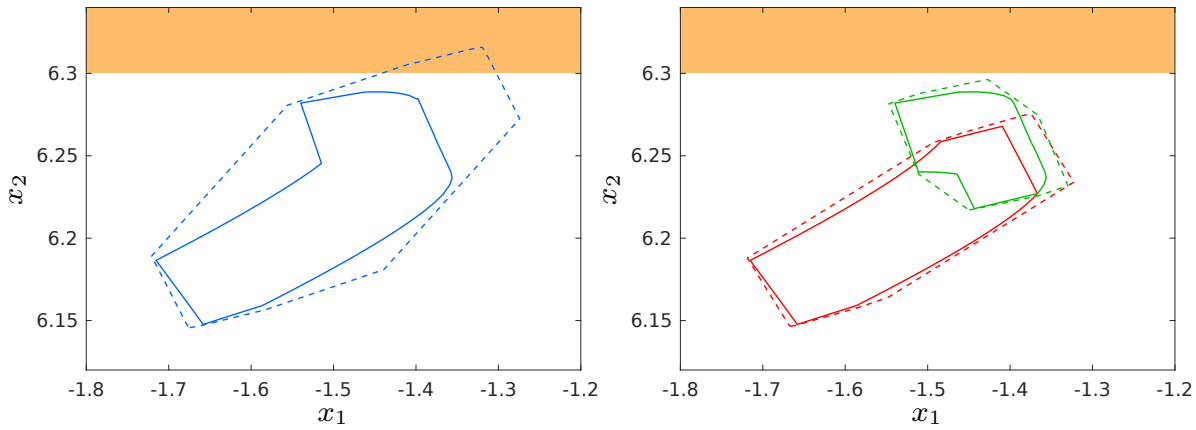


Fig. 4.1. Reachable set for the Roessler system (see Sec. 4.5.1) at time $t = 2.95$, where polynomial zonotopes are depicted by solid lines, the corresponding zonotope enclosures are depicted by dashed lines, and the unsafe set is shown in orange. While the zonotope enclosure of the original polynomial zonotope is too conservative to verify the specification (left), splitting the polynomial zonotope once reduces the over-approximation enough for verification to succeed (right).

if the distance between the most critical point $[I_n \ \mathbf{0}]\, e^{At} g(x_0)$ and the unsafe set $\mathcal{X}_u$ is smaller than the over-approximation in the polynomial zonotope $\mathcal{PZ}$, which is given by the independent generators.

Finally, we provide a closed-form expression for splitting a polynomial zonotope since this operation is not specified in the original work [124]:

Proposition 8. *(Split) Given a polynomial zonotope* $\mathcal{PZ} = \langle c, G, G_I, E \rangle_{PZ} \subset \mathbb{R}^n$ *and the index* $r \in \{1, \dots, p\}$ *of one dependent factor, the operation* $\texttt{split}(\mathcal{PZ}, r)$ *returns two polynomial zonotopes* $\mathcal{PZ}_1$,$\mathcal{PZ}_2$ *satisfying* $\mathcal{PZ}_1 \cup \mathcal{PZ}_2 = \mathcal{PZ}$*:*

$$\mathcal{PZ}_1 = \left\langle c, \begin{bmatrix} \widehat{G}_1^{(1)} & \dots & \widehat{G}_h^{(1)} \end{bmatrix}, G_I, \begin{bmatrix} \widehat{E}_1 & \dots & \widehat{E}_h \end{bmatrix} \right\rangle_{PZ}$$
$$\mathcal{PZ}_2 = \left\langle c, \begin{bmatrix} \widehat{G}_1^{(2)} & \dots & \widehat{G}_h^{(2)} \end{bmatrix}, G_I, \begin{bmatrix} \widehat{E}_1 & \dots & \widehat{E}_h \end{bmatrix} \right\rangle_{PZ}$$

with

$$\widehat{E}_i = \begin{bmatrix} E_{(\{1,\dots,r-1\},i)} & E_{(\{1,\dots,r-1\},i)} & \dots & E_{(\{1,\dots,r-1\},i)} & E_{(\{1,\dots,r-1\},i)} \\ 0 & 1 & \dots & E_{(r,i)} - 1 & E_{(r,i)} \\ E_{(\{r+1,\dots,p\},i)} & E_{(\{r+1,\dots,p\},i)} & \dots & E_{(\{r+1,\dots,p\},i)} & E_{(\{r+1,\dots,p\},i)} \end{bmatrix},$$

$$\widehat{G}_i^{(k)} = \begin{bmatrix} b_{i,0}^{(k)} \cdot G_{(\cdot,i)} & \dots & b_{i,E_{(r,i)}}^{(k)} \cdot G_{(\cdot,i)} \end{bmatrix},$$

$$b_{i,j}^{(1)} = 0.5^{E_{(r,i)}} \binom{E_{(r,i)}}{j}, \quad b_{i,j}^{(2)} = -0.5^{E_{(r,i)}} \left(2(E_{(r,i)} \bmod 2) - 1\right) \binom{E_{(r,i)}}{j},$$

where $x \bmod y$, $x, y \in \mathbb{N}_0$ *is the modulo operation and* $\binom{w}{z}$, $w, z \in \mathbb{N}_0$ *denotes the binomial coefficient. To remove redundancies we subsequently apply the compact operation as defined in [124] to* $\mathcal{PZ}_1$ *and* $\mathcal{PZ}_2$*.*

Proof. The `split` operation is based on the substitution of the selected dependent factor α_r with two new dependent factors $\alpha_{r,1}$ and $\alpha_{r,2}$:

$$\begin{aligned}\Big\{\alpha_r \mid \alpha_r \in [-1,1]\Big\} = \Big\{0.5(1+\alpha_{r,1}) - 0.5(1+\alpha_{r,2}) \mid \alpha_{r,1}, \alpha_{r,2} \in [-1,1]\Big\} \\ \Big\{0.5(1+\alpha_{r,1}) \mid \alpha_{r,1} \in [-1,1]\Big\} \cup \Big\{-0.5(1+\alpha_{r,2}) \mid \alpha_{r,2} \in [-1,1]\Big\}.\end{aligned} \tag{4.18}$$

Inserting this substitution into the definition of polynomial zonotopes in Def. 10 yields

$$\mathcal{PZ} = \left\{ c + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,i)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \alpha_k, \beta_j \in [\text{-}1, 1] \right\} \overset{(4.18)}{=}$$

$$\underbrace{\left\{ c + \sum_{i=1}^{h} \left(\prod_{\substack{k=1 \\ k \neq r}}^{p} \alpha_k^{E_{(k,i)}} \right) \left(\frac{1+\alpha_{r,1}}{2} \right)^{E_{(r,i)}} G_{(\cdot,i)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \alpha_k, \beta_j, \alpha_{r,1} \in [\text{-}1, 1] \right\}}_{=\mathcal{PZ}_1}$$

$$\cup \underbrace{\left\{ c + \sum_{i=1}^{h} \left(\prod_{\substack{k=1 \\ k \neq r}}^{p} \alpha_k^{E_{(k,i)}} \right) \left(\frac{1+\alpha_{r,2}}{-2} \right)^{E_{(r,i)}} G_{(\cdot,i)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,j)} \;\middle|\; \alpha_k, \beta_j, \alpha_{r,2} \in [\text{-}1, 1] \right\}}_{=\mathcal{PZ}_2}.$$

Finally, with

$$\begin{aligned}\left(\frac{1+\alpha_{r,1}}{2} \right)^{E_{(r,i)}} &= b_{i,0}^{(1)} + b_{i,1}^{(1)} \alpha_{r,1} + b_{i,2}^{(1)} \alpha_{r,1}^2 + \cdots + b_{i,E_{(r,i)}}^{(1)} \alpha_{r,1}^{E_{(r,i)}} \\ \left(\frac{1+\alpha_{r,2}}{-2} \right)^{E_{(r,i)}} &= b_{i,0}^{(2)} + b_{i,1}^{(2)} \alpha_{r,2} + b_{i,2}^{(2)} \alpha_{r,2}^2 + \cdots + b_{i,E_{(r,i)}}^{(2)} \alpha_{r,2}^{E_{(r,i)}}\end{aligned}$$

we obtain the equations above. □

The split operation for polynomial zonotopes is not exact, meaning that the resulting sets usually overlap (see Fig. 4.1). To minimize the size of the overlapping region we split the dependent factor with index r that maximizes the following heuristic:

$$\max_{r \in \{1,\ldots,p\}} \sum_{\substack{i=1 \\ E_{(r,i)} > 1}}^{h} \left(1 - 0.5^{E_{(r,i)}} \right) \|G_{(\cdot,i)}\|_2, \tag{4.19}$$

where $G \in \mathbb{R}^{n \times h}$ and $E \in \mathbb{N}_0^{p \times h}$ are the generator and exponent matrix of the polynomial zonotope. Moreover, since the goal of splitting in Alg. 4 is to verify a certain specification, it is advisable to first project the polynomial zonotope onto the halfspace normal directions of the unsafe set $\mathcal{X}_u$ before evaluating the heuristic (4.19) in order to direct the splitting process towards directions that are beneficial for verification.

Note that the polynomial zonotope refinement technique presented in this section is not restricted to verification of Koopman linearized systems, but can equally be applied for collision checks of polynomial zonotopes or Taylor models with halfspaces and polytopes in general. Moreover, by inverting the inequality constraints polynomial zonotope refinement

can also be applied to check if a Taylor model or polynomial zonotope is contained in a halfspace or polytope.

4.5 Evaluation Results

In this section, we present evaluation results of using the Koopman operator and random Fourier feature observables in reachability analysis. We implemented our algorithms in Julia, using the LazySets package of JuliaReach [53], the package TaylorModels.jl[1] for Taylor model arithmetic, the PyCall.jl[2] package to call dReal [90], the DataDrivenDiffEq.jl[3] package for Koopman operator linearization via Extended DMD and the DifferentialEquations.jl [164] package to generate numerical simulations. A Sobol sequence in the set of initial conditions is used to determine the initial conditions for the simulations. The resulting data matrices for each simulation are then combined via column-wise concatenation as in [190].

For each system, the dictionary of observables for Koopman linearization included multivariate polynomial basis functions up to a fixed order for the original state variables, $\sin t$ and $\cos t$, and combinations of these (*e.g.,* $x \sin^a t \cos^b t$). Lastly, SVD truncation was performed as described in [132] to remove any non-dominant modes.

Note that although we know the nonlinear differential equations for each system, we only used simulation data to perform Koopman linearization, treating each system as a black box.

4.5.1 Benchmarks

We evaluate our algorithms on four benchmark nonlinear systems: roessler model, biological model, steam model and coupled Van der Pol oscillator. This set of benchmarks contains models from 3 to 7-dimensional systems. In addition, some systems are considered more complex in terms of nonlinearity, which allows us to evaluate performance of both the linearization and reachability algorithms. As our algorithms are sensitive to the distance between the reachable set and the unsafe region, we consider parameterized unsafe regions, where a parameter i controls the distance of the reachable set to the unsafe region (if the value of i is big enough then the system is unsafe). The nonlinear differential equations for each system are available on HyPro [178] benchmark website [3].

Roessler attractor: The dynamic equations for the Roessler attractor [171] are

$$\begin{aligned}\dot{x}_1 &= -x_2 - x_3\\ \dot{x}_2 &= x_1 + 0.2\,x_2\\ \dot{x}_3 &= 0.2 + x_3\,(x_1 - 5.7),\end{aligned}$$

[1]https://github.com/JuliaIntervals/TaylorModels.jl
[2]https://github.com/JuliaPy/PyCall.jl
[3]https://datadriven.sciml.ai/

and we consider the initial set $\mathcal{X}_0 = [-0.05, 0.05] \times [-8.45, -8.35] \times [-0.05, 0.05]$, the final time $t_F = 6$, and the unsafe region $x_2 \geq 6.375 - 0.025 \cdot i$ parameterized by $i \in [0, 20]$.

Steam governor: The dynamic equations for the steam governor [182] are

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3^2 \sin(x_1) \cos(x_1) - \sin(x_1) - 3\, x_2 \\ \dot{x}_3 &= \cos(x_1) - 1,\end{aligned}$$

and we consider the initial set $\mathcal{X}_0 = [0.95, 1.05] \times [-0.05, 0.05] \times [0.95, 1.05]$, the final time $t_F = 3$, and the unsafe set $x_2 \leq -0.25 + 0.01 \cdot i$ parameterized by $i \in [0, 10]$.

Coupled Van-der-Pol oscillator: The dynamic equations for the coupled Van-der-Pol oscillator [166] are

$$\begin{aligned}\dot{x_1} &= x_2 & \dot{x_3} &= x_4 \\ \dot{x_2} &= (1 - x_1^2)\, x_2 - x_1 + (x_3 - x_1) & \dot{x_4} &= (1 - x_3^2)\, x_4 - x_3 + (x_1 - x_3),\end{aligned}$$

and we consider the initial set $\mathcal{X}_0 = [-0.025, 0.025] \times [0.475, 0.525] \times [-0.025, 0.025] \times [0.475, 0.525]$, the final time $t_F = 2$, and the unsafe set $x_1 \geq 1.25 - 0.05 \cdot i$ parameterized by $i \in [1, 16]$.

Biological system: The dynamic equations for the biological system [123] are

$$\begin{aligned}\dot{x_1} &= -0.4\, x_1 + 5\, x_3\, x_4 & \dot{x_5} &= -5\, x_5\, x_6 + 5\, x_3\, x_4 \\ \dot{x_2} &= 0.4\, x_1 - x_2 & \dot{x_6} &= 0.5\, x_7 - 5\, x_5\, x_6 \\ \dot{x_3} &= x_2 - 5\, x_3\, x_4 & \dot{x_7} &= -0.5\, x_7 + 5\, x_5\, x_6, \\ \dot{x_4} &= 5\, x_5\, x_6 - 5\, x_3\, x_4\end{aligned}$$

and we consider the initial set $\mathcal{X}_0 = [0.99, 1.01] \times \cdots \times [0.99, 1.01]$, the final time $t_F = 2$, and the unsafe set $x_4 \leq 0.883 + 0.002 \cdot i$ parameterized by $i \in [1, 10]$.

4.5.2 Approximation Error

We first investigate the accuracy of the Koopman linearized system with respect to the original nonlinear dynamics, where we compare the random Fourier feature observables against the ad hoc observables. These ad hoc observables consist of multi-variate polynomials of the system state x up to a fixed order, trigonometric functions of the time t, and combinations of these (*e.g.,* $x_1\, x_2 \sin^2(t) \cos(t)$). To obtain the data traces required for extended dynamic mode decomposition we simulate the original nonlinear systems for 500 points sampled from the corresponding initial set, where a Sobol sequence is used for sampling. For the generation of the random Fourier feature observables according to (4.12) we use the parameter $\ell = 0.3$ and $m = 71$ for the Roessler attractor, $\ell = 1.62$ and $m = 72$ for the steam governor, $\ell = 1.24$ and $m = 132$ for the coupled Van-der-Pol oscillator, and

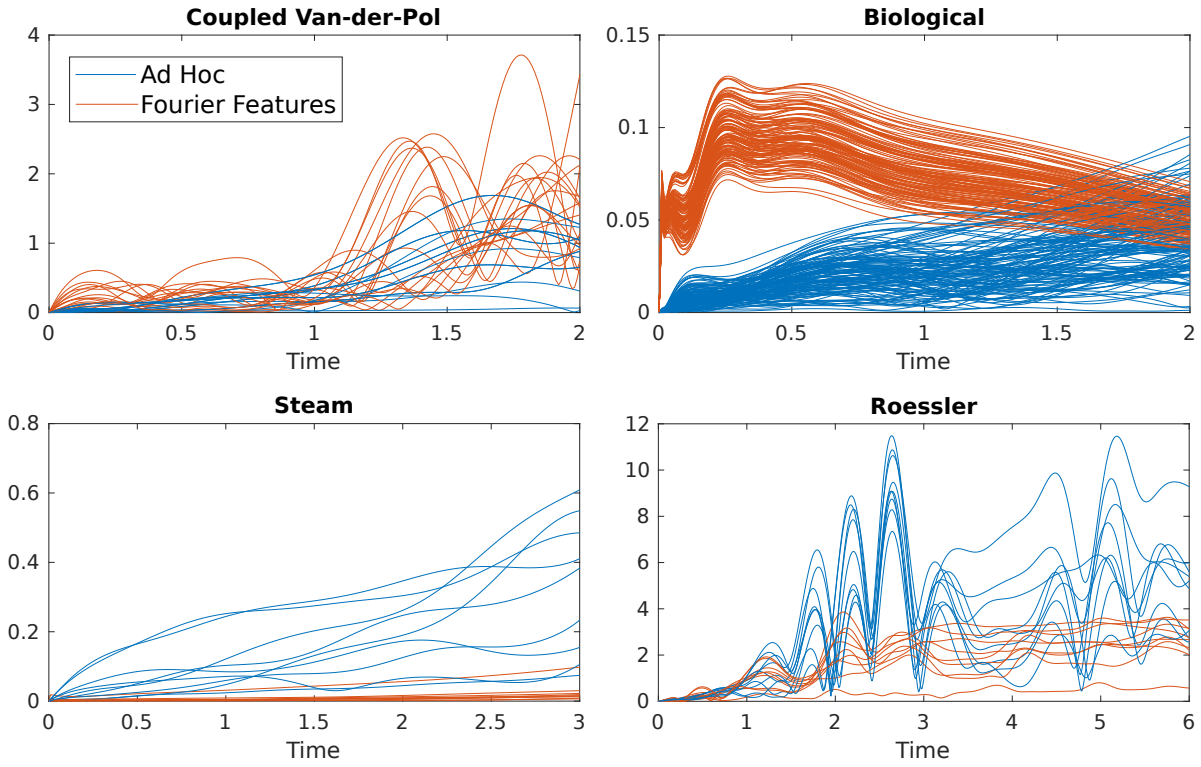


Fig. 4.2. Relative simulation error between Koopman linearized systems and the original nonlinear system in percent.

$\ell = 1.81$ and $m = 105$ for the biological system, where ℓ is the lengthscale parameter of the kernel and the number of observables m is chosen identical to the one used for the ad hoc observables [37]. As a measure for the accuracy we use the Euclidean distance between simulated trajectories for the original nonlinear system and the Koopman linearized system. The initial points for these trajectories are the center and the vertices of the initial set. According to Fig. 4.2 random Fourier feature observables are for the steam governor and the Roessler attractor more accurate than than the ad hoc observables. Moreover, while for the short time horizons considered in Fig.4.2 it seems that the ad hoc observables are more precise for the coupled Van-der-Pol oscillator and the biological system, over longer time horizons the error of the ad hoc observables is exploding. This is visualized in Fig. 4.3, where the trajectory corresponding to the ad hoc observables progresses into a completely different direction than the original system, while random Fourier features stay accurate. In this way, random Fourier features are not only a more systematic approach for choosing observables, but also improve the precision of the resulting Koopman linearized system.

4.5.3 Verification using Reachability Analysis

We now compare novel verification algorithms for Koopman linearized systems against each other, as well as different sets of observables. This evaluation highlights advantages and disadvantages of both verification methods, as well as different linearization techniques for various dynamical systems and unsafe regions. In addition, we compare against verification of the original nonlinear system using Flow* [61], a state-of-the-art tool for reachability analysis of nonlinear systems which consistently demonstrates one of the best results in

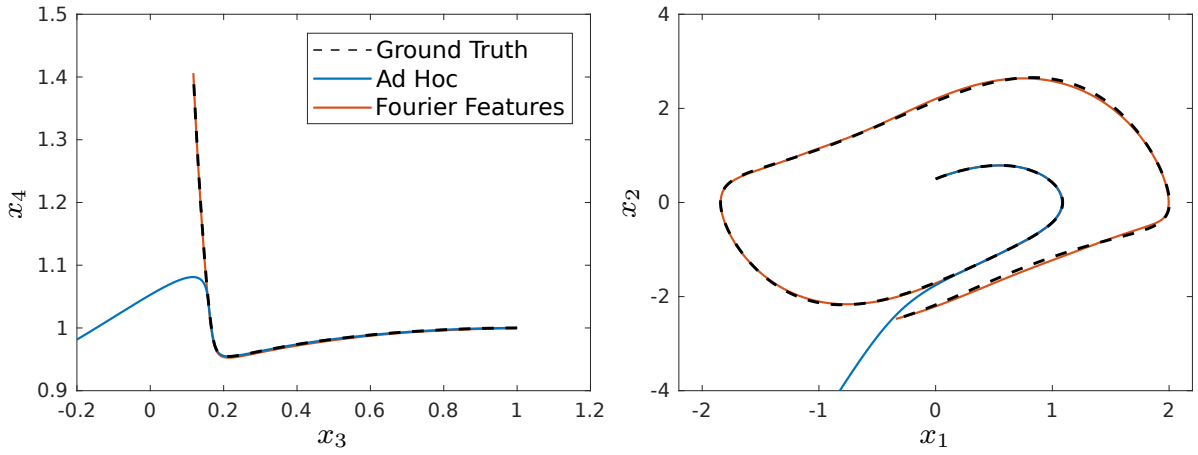


Fig. 4.3. Comparison of simulations for Koopman linearized systems with the ground truth from the original nonlinear system for a time horizon of $t_F = 10$, where the biological system is shown on the left and the coupled Van-der-Pol oscillator is shown on the right.

ARCH competition for wide set of models. In this work, we consider only discrete-time safety. Note that for discrete-time safety the reachable set computation in Line 5 of Alg. 4 simplifies to $\mathcal{R}_i = [I_n \ \mathbf{0}]\, e^{Ai\Delta t}\, \mathcal{X}_0$, $i = 0, \ldots, t_F/\Delta t$. We consider both, the ad hoc observables as well as the random Fourier feature observables.

The resulting computation times for verification are summarized in Tab. 4.2.

4.5.3.1 Performance of Hyperplane Backpropagation

We compare the performance of the Hyperplane Backpropagation (Section 4.4.1.3) against the basic Interval Encoding algorithm (Section 4.4.1.2). We show in Figure 4.4 a comparison only on the Roessler model, while similar observation can be observed on the rest of the models as well. We observe that the algorithm with backpropagation is around two times faster than the Interval Encoding algorithm for all problem instances. In addition, the computational time gets smaller as i is increased. The main reason can be that we have a smaller time horizon when i is large, because an unsafe state is reachable. We need to compute up to $t = 2.93$ for $i = 1$ and up to $t = 2.81$ for $i = 21$. We also call dReal less when i is large for the same reason. For $i = 1$, with the Interval Encoding Algorithm we call dReal 14 times, and only 8 times for Hyperplane Backpropagation. For the last instance, $i = 21$, we call dReal 7 and 3 times for the two algorithms respectively, which leads to performance improvement.

4.5.3.2 Performance of Zonotope Domain Splitting

We next evaluate Zonotope Domain Splitting on the Coupled Van der Pol oscillator to analyse the effect of different unsafe regions. We demonstrate the performance of the algorithm on two instances of the model: a safe case where $i = 4$ and an unsafe case where $i = 12$. We further evaluate using different values of max_level. We can observe that when the system is safe, Zonotope Domain Splitting with a large value of max_level generally benefits performance, whereas for $i = 12$ we see the opposite. The explanation is that for the safe instance we can save calls to dReal with backpropagation and splitting together.

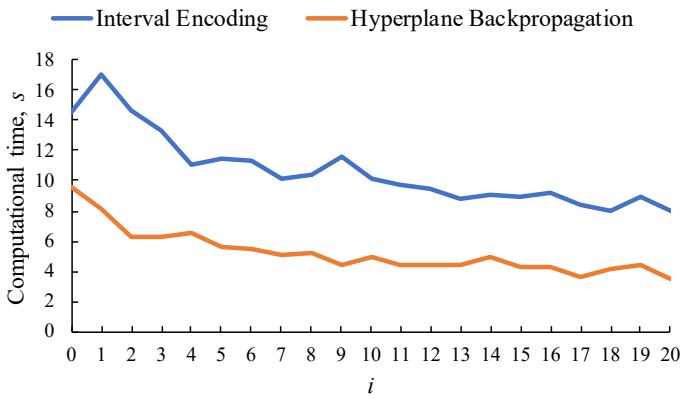


Fig. 4.4. Evaluation results of Interval Encoding and Hyperplane Backpropagation algorithms for the Roessler model for different values of i. Hyperplane Backpropagation is generally twice as fast for this problem.

Table 4.1. Evaluation results of the Zonotope Domain Splitting algorithm for the Coupled Van der Pol oscillator with $max\_level \in [0, 5]$. Safe corresponds to the $i = 4$ case and unsafe to the $i = 12$ case.

max_level	0	1	2	3	4	5
Safe	190.56	200.94	136.78	135.29	133.68	117.24
Unsafe	37.77	49.45	50.73	54.69	62.16	74.34

For $i = 12$, on the other hand, the system is unsafe and there are fewer steps where we can avoid calls to dReal. Performing splitting in this case is futile, as the overapproximation cannot prove safety of the system.

Yet another observation is that direct encoding and zonotope domain splitting are not able to verify or falsify the high-dimensional biological model at all if random Fourier feature observables are used. Both techniques eventually fallback to SMT solvers when symbolically safe guarantees cannot be provided. SMT solvers, in turn, usually are not well-suited for handling the trigonometric functions as well as the high coupling between variables used for random Fourier feature observables. Similarly to the previous comparison in Section 4.5.3.1, the results are consistent across different models and therefore evaded in this work.

4.5.3.3 Performance of Polynomial Zonotopes based algorithm

For all benchmark instances the verification algorithm based on polynomial zonotopes has the lowest computation time, and is often even magnitudes faster than the other verification approaches. The main reason for this is that with the polynomial refinement strategy we can completely avoid the computationally expensive calls to SMT solvers used by the other methods. Moreover, while the computation time for the other approaches often depends on how difficult it is to verify or falsify the specification, this algorithm exhibits roughly equal runtimes for all specifications. The explanation for this is that the polynomial zonotope refinement approach that we use for the collision checks with unsafe

Table 4.2. Computation time in seconds for verification or falsification of the benchmark systems from Sec. 4.5.1 using different approaches, where the symbol − indicates that the computation timed-out after 2 hours. The parameter i specified in the second column changes the specification, and the third column shows weather the specification can be verified or falsified.

	i	Safe?	Flow*	Direct Enc.		Zono. Split.		**Poly. Zono.**	
				ad hoc	fourier	ad hoc	fourier	ad hoc	fourier
Coupled VP	1	✓	251	788	398	0.57	171	**0.20**	**3.00**
	8	×	497	680	120	53	232	**0.79**	**3.77**
	16	×	1665	557	373	18	38	**0.20**	**2.99**
Biological	1	✓	260	470	−	0.59	−	**0.44**	**1.95**
	5	✓	250	426	−	49	−	**0.44**	**1.73**
	10	✓	238	427	−	179	−	**0.46**	**1.76**
Steam	0	✓	61	197	149	182	42	**0.12**	**0.25**
	5	×	285	59	40	37	38	**0.38**	**0.56**
	10	×	77	29	20	18	27	**0.12**	**0.26**
Roessler	0	✓	55	181	291	9.53	117	**0.55**	**0.35**
	10	×	78	177	385	5.01	241	**0.22**	**0.75**
	20	×	55	174	158	3.5	86	**0.21**	**0.34**

sets is very efficient, so that the majority of the runtime is spent on the computation of the image through the observable function using Taylor model arithmetic, a task which is independent from the specification. Interestingly, using random Fourier features instead of ad hoc observables can either prolong or accelerate the verification process, depending on the benchmark instance and verification approach used. The reasoning behind this is that random Fourier features provide a better approximation to the original system which could lead to rather straightforward verification of the safety property in some cases, but additional complexity of observables, on the other hand, require higher computational time. It should be noted, even if Fourier features prolong the time required for verification in some cases, the usage of random Fourier feature observables can be justified by their superior accuracy demonstrated in Sec. 4.5.2.

4.6 Summary

Accurate reachability and verification of nonlinear dynamical systems is a grand challenge. Many methods have been proposed for this problem, and this work has provided a new avenue to verification based on Koopman operator linearization. This process outputs a system of linear dynamics with nonlinear constraints on the initial state set.

As demonstrated on several nonlinear system benchmarks, the combination of these two techniques is both extremely accurate and extremely fast.

The main trade-off with Koopman linearized systems is that the guarantees are on the system approximation, not the original system, which is the main limitation of the proposed approach. Note, that if the bounds of the approximation error can be computed,

then this restriction can be dropped, as polynomial zonotopes can take the error into consideration during the analysis without any substantial changes to the whole algorithm or its performance. Despite this, we believe the method could still be useful for verification in systems engineering, where the goal is to produce evidence that the system meets its requirements. Koopman linearization process uses data to create linear approximations in the space of observable functions. Thus, we can compute reachable states for black-box systems, while most traditional reachability methods cannot be applied to this class of systems.

It could also be effective for finding unsafe counterexamples—falsification—or to analyze systems where only simulation code is provided, or even real-world systems where sensor measurements could be used to create a Koopman linearized model for analysis.

Chapter 5

Online Reachability Analysis using Barrier Certificates

Inductive methods for reachability offer an alternative approach in which one looks for a so-called *barrier certificate* [188, 161], a function of the system state whose zero-level set separates the "unsafe" region from the system trajectories that start from a given initial set. The existence of a barrier certificate entails that the system is safe. Three advantages of barrier certificates are that they can establish safety: 1) over infinite time horizons, 2) for systems with nonlinear dynamics, and 3) uncertain inputs or parameters. Reachability methods that offer some or all of these advantages do exist, but are usually computationally expensive. Checking that a candidate function is indeed a barrier certificate can be done in many cases efficiently and automatically. However, finding barrier certificates is much harder, and this is a disadvantage of the approach.

5.1 Chapter overview and structure

In this chapter, we present a novel technique for online safety verification of autonomous systems which performs bounded/unbounded horizon reachability analysis efficiently by neural barrier certificates. In particular, we provide a novel approach to synthesise, via barrier certificates, sound over-approximations of reach sets for dynamical systems, whether linear or nonlinear, possibly under control inputs. We go beyond establishing safety by computing valid over-approximations *without* explicitly solving the system dynamics.

We interpret the search of such over-approximations as a synthesis of barrier certificates, which we tackle with a sound algorithm. We devise a two-level technique which utilises neural networks as efficient emulators of both the system dynamics and barrier certificates. Since barrier certificate candidates are efficiently generated via neural networks, our approach enables online computation of reach sets after appropriate offline training. Key to our approach is a MetaNN that acts as a generalisation of example barrier certificates that depend on initial and unsafe sets. We use FOSSIL [8] as the underlying generator of example barrier certificates with correctness guarantees. The FOSSIL tool can formally synthesise neural network-based barrier certificates. FOSSIL is attractive for this work due

to its ability to provide *training data* (*i.e.,* example barrier certificates) with correctness guarantees. Furthermore, neural network templates allow for the generation of barrier certificates (and consequently reach sets) that are more general in structure and shape, which are especially useful for handling nonlinear, non-polynomial vector fields. We note that FOSSIL does not scale to on-the-fly use due to the intrinsic complexity of the algorithms implemented – this motivates the introduction of our meta-neural network. The overall soundness of our approach is guaranteed by utilising SMT solvers for checking candidate barrier certificates.

Our approach uses barrier certificates given by parameterised neural networks that depend on a given initial set, on unsafe sets, and on the time horizon – when these parameters are fixed, the networks generate (putative) barrier certificates for the system. Such networks are trained efficiently offline using system simulations over sampled regions of the state space. A meta-neural network (MetaNN) is then employed to generalise the barrier certificates to state space regions beyond the training set. These certificates are generated and validated on-the-fly as sound over-approximations of the reachable states, thus either ensuring system safety or activating appropriate alternative actions in unsafe scenarios. Finally, we demonstrate our technique on case studies from linear models to nonlinear control-dependent models for autonomous driving.

We show that our approach successfully generates sound over-approximations of reach sets of linear and nonlinear systems. The efficiency of reach-sets generation is demonstrated in an autonomous driving scenario. Our approach not only extends the capabilities of barrier certificate verification to the synthesis of parameterised controllers, but also increases the success rate of online barrier certificate generation from 78% to 99% compared to direct application of the underlying barrier generator FOSSIL (at a cost of 10 hours of offline training on a standard laptop).

To summarise, the contributions of this work are:

- We show that level sets of barrier certificates over-approximate reach sets.
- Founded on continuity properties of system dynamics, we consider barrier certificates as functions of initial and unsafe sets and use them for reach set computations.
- We provide a computational framework that fixes parameterised templates for initial and unsafe sets and computes a MetaNN as parameterised barrier certificates.
- We employ offline training and fast online validation and execution that increase the success rate of generating barrier certificates for safe autonomous driving.

The chapter is structured as follows:

- Section 5.2 elaborates on how we plan to use barrier certificates for computations of reachable sets.
- Section 5.3 describes the main algorithm proposed in the work, detailing the training of a neural network that generalises the construction of reach sets.

- Finally, evaluation results using the proposed methodology are presented in Section 5.4; the concept is demonstrated using some toy case studies, before its efficacy is demonstrated using an online vehicle path planning case study.

Our proposed framework is applicable to any tool that generates certified barrier certificates. This chapter serves as a basis for our paper, which is currently being prepared for a submission [7].

5.2 Barrier Certificates for Reach Sets

Barrier certificate theory and the third condition of Eq. (2.13) leverages Nagumo's theorem [26] to demonstrate the existence of an invariant set in the vector field f. Meanwhile the first two conditions ensure that our initial set lies within this invariant set and that our unsafe set does not. Notably, any invariant set containing our initial set must be a *true* reachable set from the initial set. This is our first contribution and is stated formally next as a corollary of Theorem 1.

Corollary 1.1. *Define $\mathcal{R}$ to be the set of states reachable from the initial set $\mathcal{X}_0$ under the dynamics Eq. (2.2). Assume there is barrier certificate $B(x)$ satisfying the conditions of Definition 11. Then, the set $\mathcal{R}_o = \{x \in \mathcal{X} \mid B(x) \leq 0\}$ gives an over-approximation of $\mathcal{R}$,* i.e., $\mathcal{R} \subseteq \mathcal{R}_o$.

The intuition behind this over-approximation is the fact that the trajectories starting from $\mathcal{X}_0$ cannot escape the set $\mathcal{R}_o$ since $B(x(\cdot)) : \mathbb{R}_{\geq 0} \to \mathbb{R}$ does not change the sign and is always non-positive for trajectories starting from $x_0 \in \mathcal{X}_0$.

Proof of Corollary 1.1 Take any trajectory $x(t)$ that starts at some $x_0 \in \mathcal{X}_0$, and consider the evolution of $B(x(t))$ along this trajectory. The first condition in (2.13) implies $B(x_0) \leq 0$. Together with the third condition in (2.13) we get that the flow of the system $B(x(t))$ cannot become positive. Consequently, the set $\mathcal{R}_o = \{x \in \mathcal{X} \mid B(x) \leq 0\}$ contains the flow of the system and over-approximates the reach set $\mathcal{R}$.

Barrier certificates can also be used to study the safety of dynamical systems when their trajectories are restricted to a subset of the state space. This is formally stated in the next theorem and will be relevant in our case study on safe autonomy.

Theorem 2. *Consider a working region $\mathcal{X}_b \subseteq \mathcal{X}$, the initial set $\mathcal{X}_0 \subseteq \mathcal{X}_b$, and unsafe set $\mathcal{X}_u \subseteq \mathcal{X}$. If a function $B : \mathcal{X} \to \mathbb{R}$ satisfies the following conditions:*

$$\begin{aligned} &B(x) \leq 0 \ \forall x \in \mathcal{X}_0, \ B(x) > 0 \ \forall x \in \mathcal{X}_u \cap \mathcal{X}_b, \\ &\frac{\partial B(x)}{\partial x} f(x) < 0, \ \forall x \in \mathcal{X}_b \ s.t. \ B(x) = 0, \end{aligned} \tag{5.1}$$

then the trajectories of Eq. (2.2) evolve inside $\mathcal{R}_b := \{x \in \mathcal{X}_b \mid B(x) \leq 0\}$ before leaving the working region $\mathcal{X}_b$.

Theorem 2 means that we can use $\mathcal{R}_b$ as an over-approximation of the reach set when trajectories are restricted to $\mathcal{X}_b$.

Proof of Theorem 2. Define $\tau^*(x_0)$ as the first time a trajectory leaves $\mathcal{X}_b$ when the trajectory starts from the initial state $x_0 \in \mathcal{X}_0$. Note that $\tau^* \in \mathbb{R}_{\geq 0} \cup \{+\infty\}$, where $\tau^* = +\infty$ indicates that the trajectory remains inside $\mathcal{X}_b$ for all $t \geq 0$. Let us define the partition $\mathcal{X}_0 = \mathcal{X}_{01} \cup \mathcal{X}_{02}$, where

$$\mathcal{X}_{01} = \{x_0 \in \mathcal{X}_0 \,|\, \tau^*(x_0) = +\infty\},$$
$$\mathcal{X}_{02} = \{x_0 \in \mathcal{X}_0 \,|\, \tau^*(x_0) < +\infty\}.$$

For the initial set $\mathcal{X}_{01}$, we simply restrict the space of the system to X_b and the result hold by applying Corollary 1.1 to the initial set $\mathcal{X}_{01}$ and unsafe set $\mathcal{X}_u \cap \mathcal{X}_b$. For the initial set $\mathcal{X}_{02}$, the trajectory $x(t)$ satisfies both conditions $B(x(t)) \leq 0$ and $\frac{\partial B(x)}{\partial x} f(x(t)) < 0,\ \forall x(t) \in \mathcal{X}_b$ with $B(x) = 0$ before leaving $\mathcal{X}_b$. This implies that $B(x(t))$ is non-positive for all t before $x(t)$ leaving $\mathcal{X}_b$. This proves that $\mathcal{R}_b$ is an over-approximation of the reachable states of trajectories before the trajectory leaves $\mathcal{X}_b$.

5.3 Computation of Safe Reach Sets

This work utilises several neural networks throughout. Here, we formalise the notation used when referring to these networks. Feed-forward neural networks are such as that depicted in Fig. 5.1. Consider a network $\mathcal{N}$ with h_0 input neurons, k hidden layers each with $h_1, \ldots, h_k$ neurons and an output layer of h_{k+1} neurons. We consider each layer to be fully connected and denote the corresponding weight matrix for the i-th layer as W_i and

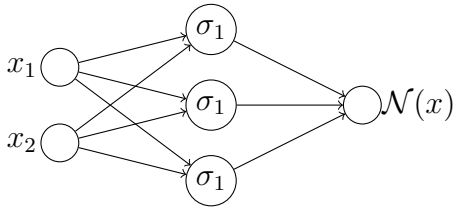


Fig. 5.1. A feed-forward neural network with two input neurons, one hidden layer and a single output neuron.

bias vector as b_i for $i = 0, \ldots, k+1$, where $i = 0$ denotes the input layer and $i = k+1$ the output layer. Each hidden layer has a corresponding activation function $\sigma_i : \mathbb{R} \to \mathbb{R}$ such that the output of the i-th layer is given by $z_i = \sigma_i(W_i z_{i-1} + b_i)$.

5.3.1 High-level overview of the framework for online verification

We now describe the first key contribution of this work: a methodology to produce sound reachable sets for dynamical systems, through the synthesis of sound barrier certificates. However, verification using barrier certificates can be limited by performance issues that are intrinsic to the use of SMT solvers and gradient descent based synthesis approaches, with procedures either being slow to return solutions or not returning one at all (in view

of incompleteness – see [185] for several results on soundness and relative completeness of inductive verification techniques). This can be particularly restrictive in practical settings or in applications where on-the-fly verification is required, as we shall see later. We alleviate this issue by generalising over the space of barrier certificates for some fixed problem settings.

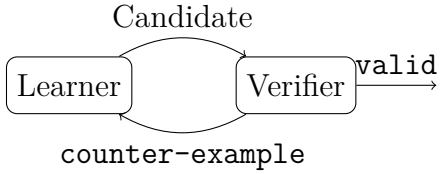


Fig. 5.2. The Counter-Example Guided Inductive Synthesis (CEGIS) loop.

Our proposed framework is applicable to any tool that generates sound barrier certificates. We employ FOSSIL [8], which uses internally a CEGIS loop. The CEGIS procedure is represented in Figure 5.2, which illustrates the two fundamental complementary components: the learner and the verifier. The learner is a neural network. The structure and characteristics of the neural network (*i.e.,* layers and activation functions) can be selected freely within the tool. This includes structure of the network, *i.e.,* the number of hidden layers and the number of neurons within these layers, and secondly the activation functions present within these layers. The selected activation functions may be simple polynomial functions, resulting in polynomial-based barrier certificates, or may be other nonlinear activation functions used in machine learning. This flexibility allows for the construction of reach sets of varying structural complexity.

We utilise machine learning techniques and barrier certificate properties to efficiently compute sound reachable set for the system modelled by Eq. (2.2). Given an initial region $\mathcal{X}_0$ as an **input** we seek a barrier certificate as an **output**. This barrier certificate either covers the reachable set $\mathcal{R}$ or along with working region $\mathcal{X}_b$ result into an over-approximation $\mathcal{R}_o$ of $\mathcal{R}$.

The verifier is an SMT-solver that can check the satisfiability of formulae over the real numbers. Its role is to provide correctness guarantees of the candidate barrier certificates proposed by the learner, or alternatively return to the learner counter-examples of points within the state space where the candidate barrier certificate is invalid. This is achieved by checking the satisfiability of the negation of the conditions in Eq. (2.13). FOSSIL leverages two choices of SMT-solver: Z3 [72] and dReal [90]. Z3 is limited to reasoning over polynomial functions, whereas dReal can handle any nonlinear function. However, dReal may return spurious counter-examples, since checking satisfiability of general nonlinear formulae is an undecidable problem.

We present schematically our algorithm in Fig. 5.3. Our approach has two main phases. During the first (offline) phase we generate a collection of initial sets $\mathcal{X}_0$, which are used by the simulation engine to build working regions $\mathcal{X}_b$ for the system. Then we feed the initial sets and their corresponding working regions to FOSSIL to generate barrier certificates for the reachable sets. We use the initial sets, the working regions, and the

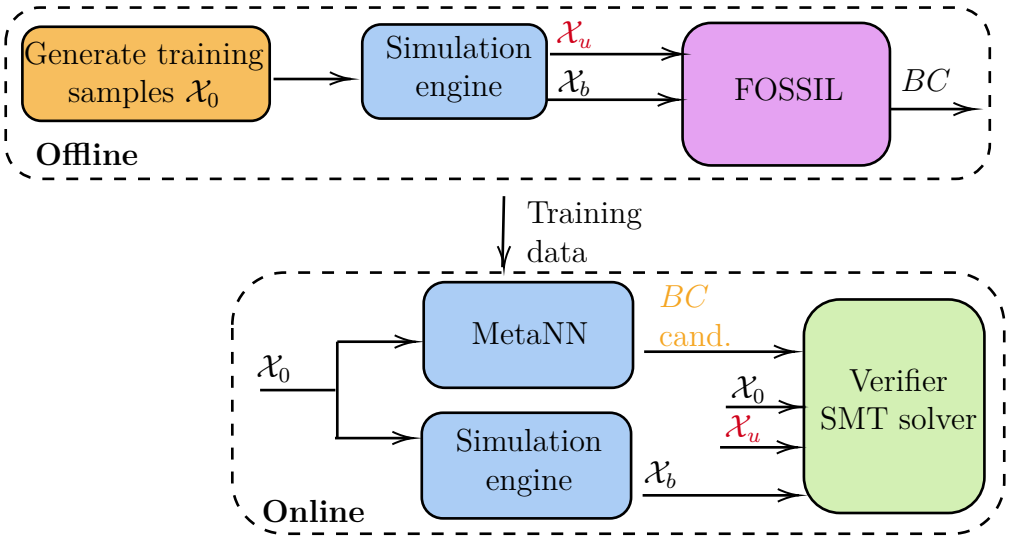


Fig. 5.3. The proposed framework has two phases: (i) an offline phase (top) where the training data is prepared and the MetaNN is trained; and (ii) an online phase (bottom) where sound reach sets are generated via neural barrier certificates. $\mathcal{X}_0$ = initial set; $\mathcal{X}_u$ = unsafe set; $\mathcal{X}_b$ = working region; BC= barrier certificate.

barrier certificates to train MetaNN, whose task is to generalise the computation of barrier certificates. Essentially, MetaNN is an (efficient) emulator of FOSSIL.

We now illustrate phase two (online phase) of our approach. We pass an initial set $\mathcal{X}_0$ to both the simulation engine and MetaNN to compute a working region $\mathcal{X}_b$ and a barrier certificate candidate, respectively. Next, the validity of the candidate with respect to $\mathcal{X}_0, \mathcal{X}_u$ and $\mathcal{X}_b$ is checked by an SMT solver. If the candidate barrier certificate is valid, then we have attained our aim of computing a safe reach set. Otherwise, we can use the counter-example generated by the SMT solver to refine the training of MetaNN.

Remark 2.1. *The framework presented in Fig. 5.3 is founded on continuity properties of the system trajectories with respect to its initial state. The barrier certificate can be seen not only as a function of state x but also a function of a parametrisation of the initial set $\mathcal{X}_0$. The aim of MetaNN is to give such a parameterised barrier certificate, and it is obtained by training MetaNN on a dataset of barrier certificates.*

Online safe planning Our approach can be easily adapted to implement online, safe planning over bounded time horizons, as depicted in Fig. 5.4. Briefly, the online phase of our approach is embedded in a loop that provides control inputs and iteratively checks for system safety. The loop uses two controllers: a 'base' controller for normal use and a 'backup' controller which takes over in emergency situations. The idea is that the base controller provides inputs to control the system over a short time horizon; such inputs and the current system state are passed to the simulation engine and to MetaNN to compute the next working region and a candidate barrier certificate (reach set), respectively. These are in turn passed to the SMT solver, which decides whether the system is safe (for the current time horizon), *i.e.,* that the computed reach set/barrier certificate and the unsafe set do not intersect. If the SMT solver cannot guarantee an empty intersection, then the

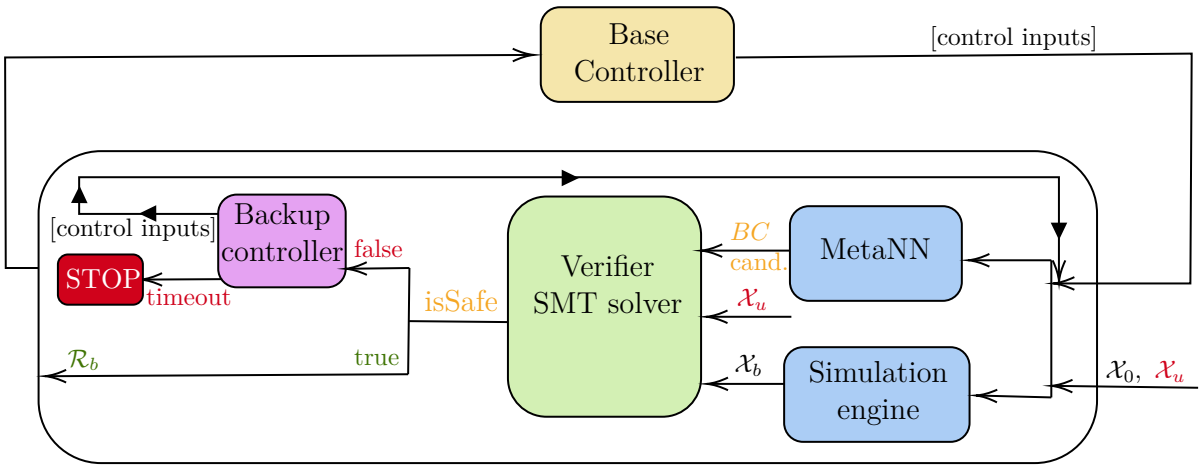


Fig. 5.4. Structure of our online safe planning approach. The base controller (yellow) operates in normal conditions, while the backup controller (purple) is activated to avoid unsafe regions in case of potential collision. The MetaNN has been already trained. Verifier checks the barrier certificate (BC) candidate and calls backup controller in case the result is unsafe. Then the process of generating a BC candidate and running the simulation engine repeats and new BC candidate is checked. If the system is unsafe or the MetaNN does not generate valid BC within the time bound, then the system stops.

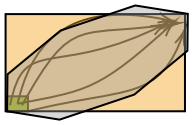
backup controller is activated to divert from unsafe set and will provide input controls for the current time horizon. However, if the verifier cannot guarantee the system is safe even for the input from the backup controller, then the system stops. Otherwise, the inputs provided by the base controller are applied to the actual system, and the loop repeats for the next time horizon. In the current implementation, to generate an initial set for the next time horizon we use a center trajectory as the dynamics of the system. In reality any trajectory can be used instead and it does not affect the soundness of the algorithm. In addition, we bloat around the last state of the trajectory to account for sensing errors (e.g. time lag or silent errors) of real-world models.

We now discuss each part of our framework in more detail.

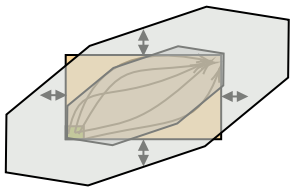
5.3.2 Simulation Engine

Given an initial set $\mathcal{X}_0$ (an hyperbox), the simulation engine generates a working region for it. For this purpose we run simulations from the vertices of the initial set and a random sample of points inside of it. Then we take the convex hull of the generated trajectories. This set allows us to compute a working region $\mathcal{X}_b$ for the system starting in $\mathcal{X}_0$, which is basically a box over-approximation of the generated convex hull. Note that for stable systems we can use unbounded sets as working regions. Then, we also generate an octagonal over-approximation of the convex hull. The resulting set can encompass the unsafe region, if one is not provided. In this case the unsafe set $\mathcal{X}_u$ is the complement of the octagonal over-approximation of the convex hull. However, the approximation could be tight thus making it harder for FOSSIL to find a valid barrier certificate. Thus, if after a specified number of CEGIS loops FOSSIL cannot provide a certificate, then we bloat this over-approximation, *i.e.,* push back the unsafe region.

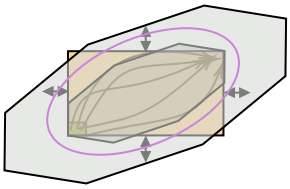
(a) Step 1.
Running simulations from the initial set X_0.



(b) Step 2.
Construction of a working region X_b and octagonal over-approximation.



(c) Step 3.
Bloating of the octagonal over-approximation.



(d) Step 4.
Computation of reachable set R_b as an intersection of a constructed barrier certificate (purple ellipsoid) and bloated octagonal

Fig. 5.5. Four-step construction of a Safe Reach Set R_b.

5.3.3 Training MetaNN

Let $\mathcal{X}_0$ denote initial sets represented as parameterised boxes and $a \in \mathbb{R}^d$ the coefficients of a barrier certificate for a given template with d terms. For each initial set, we compute a corresponding unsafe set $\mathcal{X}_u$ as described previously, which represents the non-reachable region of the state space. We seek to learn a mapping from $\mathcal{X}_0$ to a, *i.e.,* MetaNN should be able to generate barrier certificates that represent the reach sets of the system.

Computation of Safe Reach Sets We demonstrate how to compute safe reachable region $\mathcal{R}_b$: this is done in four separate steps. Figure 5.5(a) illustrates the first step: here we run multiple simulations from the initial set $\mathcal{X}_0$. Figure 5.5(b) shows how we construct from simulation traces the working region $\mathcal{X}_b$ and octagon approximation. Then, we bloat the octagon region by a factor ε, and pass its complement $\mathcal{X}_u$ to FOSSIL, along with $\mathcal{X}_0$. Once we obtain a valid barrier certificate from FOSSIL, we intersect it with the working region X_b. Whenever FOSSIL cannot produce a valid barrier certificate for the provided X_b, we further bloat the working region and call FOSSIL again. We repeat this until a certificate is found or until a specified maximal bloating level is reached, in which case the system is deemed unsafe.

Training Data Generation The first step is to generate sample data on which MetaNN can be trained. The procedure for this is as follows. We start by generating with FOSSIL a barrier certificate $B(a, \cdot)$ for each pair $(\mathcal{X}_0, \mathcal{X}_u)$, and by storing the coefficients a of the barrier certificate, which will be our training data. Once trained, MetaNN should map an initial set to a set of coefficients for which the barrier certificate results in a set $\mathcal{R}_0$ such that $\mathcal{R} \subseteq \mathcal{R}_0$, *i.e.,* an over-approximation of the true reach set.

The barrier certificate generation in FOSSIL uses neural network templates. FOSSIL initialises weights and biases of the neural network template randomly, which are then guided to a solution using gradient descent. However, since the training can be done by

sequentially moving the initial set gradually and iteratively through the state space, we can utilise this to find more useful training data. Specifically, we 'seed' the parameters of the next initialisation using the parameters of the most recent solution. In other words, we begin searching for the next barrier certificate from the previous one. The advantage of this seeding is twofold. First, it reduces the amount of computation required to generate the training data by improving the performance of FOSSIL since we start closer to a valid certificate. Second, it allows us to take advantage of potential continuity results when generalising over a function space. Many barrier certificates may exist for a given initial set and template. While the resulting functions may be structurally similar, the corresponding coefficients may not be, making generalisation more difficult. This seeding approach should allow for more-similar barrier certificates representations a which in turn should enhance the ability of MetaNN to generalise without over-fitting. Note that if at any point we fail to find a barrier certificate for a given initial set, we simply use the most recent solution to 'seed' the next attempt.

For a given data point consisting of an initial set $\mathcal{X}_0$ and a barrier certificate representation a, it is reasonable to assume that other valid barrier certificate representations exist near (in the Euclidean sense) to a. Thus, we can generate additional data points for a fixed initial set by perturbing the representation a with multiplicative noise as $\tilde{a} = a \cdot c$, where $c \sim N(1, \sigma^2)$ and N is the normal distribution with unit mean and variance σ^2. Then, we can check via FOSSIL the *new* barrier certificate constructed from $\tilde{a}$ to verify whether it is also a valid barrier certificate for the given initial set and corresponding unsafe (unreachable) region. If so, the generated barrier certificate can be added to the training dataset as a new data point. If σ is sufficiently large this will provide meaningful additional data to the training procedure.

Loss function and Network Structure. MetaNN is trained using gradient descent using an L1 loss function, which for n_d data points can be expressed as

$$\mathcal{L} = \frac{1}{n_d} \sum_{i=i}^{n_d} |a_i - \mathcal{N}(\mathcal{X}_{0_i})|, \tag{5.2}$$

where $|\cdot|$ denotes the absolute value, $\mathcal{X}_0$ are input sets and a is the output of the network representing the barrier certificate coefficients. ReLUs are used as activation functions during the training phase. The output of MetaNN are coefficients of the barrier certificate for a given barrier template. We use second-order polynomial functions throughout our tests, however alternative templates are also supported.

5.3.4 Validation of MetaNN

Before deploying MetaNN we validate its reliability at generating safe reach sets for a separate *test* dataset. As for training, the test dataset is composed of input-output pairs $(\mathcal{X}_0, a)$, where $\mathcal{X}_0$ are input sets and a are the coefficients of a barrier certificate for a candidate reach set. The validation consists in certifying whether the set $\mathcal{R}_0$ constructed

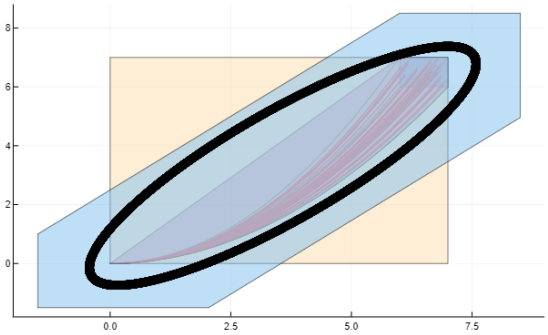


Fig. 5.6. Safe reach set for the linear system with with real eigenvalues. The red traces are simulation trajectories. The inner and outer octagons are, respectively, the convex hull of the generated trajectories, and its over-approximation. The yellow rectangle is a bounding box. The black ellipsoid is the zero level-set of the validated barrier certificate.

using the candidate certificate computed by MetaNN is in fact a sound over-approximation of the true reach set $\mathcal{R}$.

Consider the barrier certificate $B(a, \cdot)$ constructed from the MetaNN output a for a given initial set $\mathcal{X}_0$. Let the corresponding unsafe set $\mathcal{X}_u$ be the empty set, $\mathcal{X}_u = \emptyset$. We obtain the working region set $\mathcal{X}_b$ as described above. Let the certificate conditions in Eq. (5.1) be $\mathcal{F}(x)$. Using the SMT solver dReal [90], we seek a witness to $\neg\mathcal{F}(x)$. If no witness is found, we have certified the function $B(a, \cdot)$ as a barrier certificate for a given initial set and working region, with no unsafe set. The set $\mathcal{R}_o = \{x \in \mathcal{X} \mid B(a, x) \leq 0\}$ is an invariant set containing the initial set, and hence is a sound over-approximation of the true reach set $\mathcal{R}$. We count this as a successful or valid output of MetaNN, and our validation consists of finding the percentage of valid outputs obtained over the test inputs.

In practice, this validation is performed over a large number of data points to obtain a statistical estimate of the reliability of the network's outputs as safe reach sets.

To validate a barrier certificate candidate, we pass computed in MetaNN coefficients with the barrier certificate template into SMT solver to test its properties. We substitute X_0 with the θ_I, X_u with the θ_U.Since our barrier certificate candidates over-approximate reachable sets, then we need to prove the following : Reach $(\theta_I) \subseteq \{x | Barrier(\theta_I, \theta_U) \leq 0\}$.

5.4 Evaluation Results

In this section, we first apply our framework for the computation of safe reach sets on a number of linear and nonlinear autonomous models. Then, we present the application in an autonomous driving setup. We have implemented our algorithms in Julia, using the LazySets package of JuliaReach [53] for set operations and the DifferentialEquations.jl package [164] to perform simulations, the PyCall.jl[1] package to call FOSSIL and dReal. We utilise PyTorch with CUDA support to train the MetaNN.

Stable linear system with real eigenvalues. Consider the dynamics:

[1]https://github.com/JuliaPy/PyCall.jl

$$\begin{cases} \frac{d}{dt}x(t) = -x, & x(0) = x_0 \\ \frac{d}{dt}y(t) = -2 \cdot y, & y(0) = y_0. \end{cases}$$

We have generated valid barrier certificates for different initial sets with a fixed size within a specified domain, and used them to train the MetaNN. A sample of barrier certificates generated for this model is shown in Fig. 5.6. The input to the MetaNN is the initial set, represented as coefficients of its support vectors. We have tested the MetaNN over different initial sets of different size, but within the same specified domain. The correctness of the output of the MetaNN, *i.e.,* the validity of the barrier certificates, is 99.3% over 1000 randomly sampled initial sets of random size.

Stable linear system with complex eigenvalues. Consider the dynamics:

$$\begin{cases} \frac{d}{dt}x(t) = y, & x(0) = x_0 \\ \frac{d}{dt}y(t) = -0.2 \cdot x - 0.2 \cdot y, & y(0) = y_0. \end{cases}$$

A sample of barrier certificates generated for this model is shown in Fig. 5.7. Similarly to the previous model, we have trained the MetaNN, and the correctness of the obtained neural barrier certificates is 99.2% over 1000 randomly sampled initial sets of random size.

Nonlinear jet engine system. The model of a jet engine taken from the HyPro [178] benchmark repository is

$$\begin{cases} \frac{d}{dt}x(t) = -y - 1.5 \cdot x^2 - 0.5 \cdot x^3 - 0.5, & x(0) = x_0 \\ \frac{d}{dt}y(t) = 3 \cdot x - y, & y(0) = y_0. \end{cases}$$

A sample of barrier certificates generated for this model is shown in Fig. 5.8. As the dynamics of this system is qualitatively similar the previous one, we did not train and test the meta-neural network.

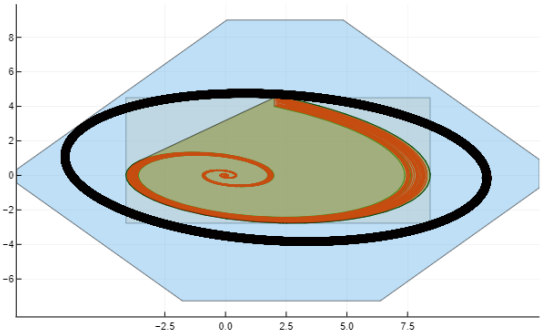


Fig. 5.7. Safe reach set for linear system with spiralling stable dynamics. The red traces are simulation trajectories. The inner and outer octagons are, respectively, the convex hull of the trajectories and its over-approximation. The yellow rectangle is a bounding box. The black ellipsoid is the zero level-set of the validated barrier certificate.

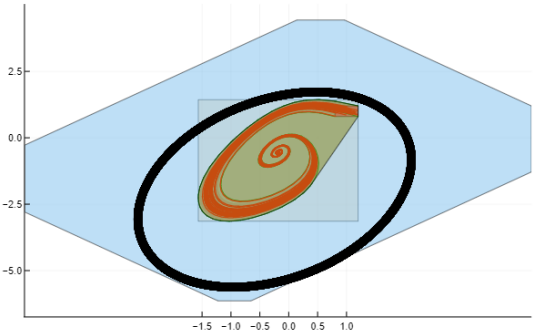


Fig. 5.8. Safe reach set for the nonlinear model of a jet engine system. The red traces are simulation trajectories. The inner and outer octagons are, respectively, the convex hull of the trajectories and its over-approximation. The yellow rectangle is a bounding box. The black ellipsoid is the zero level-set of the validated barrier certificate.

Online safe path planning for vehicle dynamics. Consider the dynamics:

$$\begin{cases} \frac{d}{dt}x(t) = v\cos\theta(t), & x(0) = x_0 \\ \frac{d}{dt}y(t) = v\sin\theta(t), & y(0) = y_0 \\ \frac{d}{dt}\theta(t) = \omega, & \theta(0) = \theta_0, \end{cases} \tag{5.3}$$

where (x, y) indicate the location in two-dimensional space and θ is an angle describing the orientation of the car. We assume that the velocity v and the rotational speed ω are inputs that are selected by a feedback controller.

The goal is to design v and ω such that the car starting from an initial location will reach a target region $\mathcal{X}_r$, while avoiding obstacles that are present over the (x, y) domain. The obstacles considered in this case study are static boxes, however the implementation of our approach can in principle also handle dynamically-changing obstacles. We emphasise that the control inputs ought to be executed in a provably safe manner, with sound certificates.

The literature in path planning is mature but it generally neglects the dynamics of the vehicle, and most often does not come with formal safety guarantees. We show that we can first neglect the obstacles and design a baseline controller for steering the dynamics from the initial set to the target region $\mathcal{X}_r$. We then show how the scheme presented in Sec. 5.3 can be used to provide sound safety guarantees on the executed control inputs.

We have designed the following baseline controller for reaching the target:

$$\begin{cases} v(t) = \alpha_1 \left(1 - \exp\left(-\alpha_2 \|(x(t) - x_r, y(t) - y_r)\|_2\right)\right) \\ \omega(t) = \alpha_3(\theta_r(t) - \theta(t)) \\ \theta_r(t) := \arctan\left(\frac{y(t)-y_r}{x(t)-x_r}\right), \end{cases} \tag{5.4}$$

where $(x_r, y_r) \in \mathcal{X}_r$ is a reference point within the target region and $\theta_r(t)$ is the relative angle between current and target location. The angular speed $\omega(t)$ is designed to steer the car from the current orientation towards the reference angle. The speed is also proportional

to the distance from the reference point passed trough an exponentially decaying function, thus slowing down the vehicle when approaching the reference point. Note that this simple controller can be replaced with any path planning algorithm that can be synthesised over Eq. (5.3).

The controller in Eq. (5.4) does not consider obstacles and does not come with any safety guarantees. These desirable extensions can be addressed by our safe reach set approach. First, let us note that the dynamics in Eq. (5.3) admit two structural properties that are useful in the computation of barrier certificates. While these properties are *not* specifically required by our approach, they can be utilised to reduce the computational burden of our technique.

Property 1. The dynamics are shift invariant with respect to location (x, y) for fixed control inputs (v, ω): if $(x(t), y(t), \theta(t))$ is a solution from an initial condition (x_0, y_0, θ_0), then $(x(t) - x', y(t) - y', \theta(t))$ is a solution from the initial condition $(x_0 - x', y_0 - y', \theta_0)$. This means the computation of barrier certificate can always be done for initial sets that are centred at the origin $(0, 0)$.

Proof of Property 1. The proof is by direct substitution. Since the vector field in the right-hand side of (5.3) is independent of the location variables (x, y), then adding any constant value to these variables also results in a valid solution for these set of differential equations.

Remark 2.2. *If $B(x, y, \theta; v, \omega)$ is a barrier certificate parameterised with (v, ω) for initial and unsafe sets $\mathcal{X}_0, \mathcal{X}_u$, then $B(x, -y, -\theta; v, -\omega)$ is a barrier certificate for initial and unsafe sets $\bar{\mathcal{X}}_0, \bar{\mathcal{X}}_u$, where*

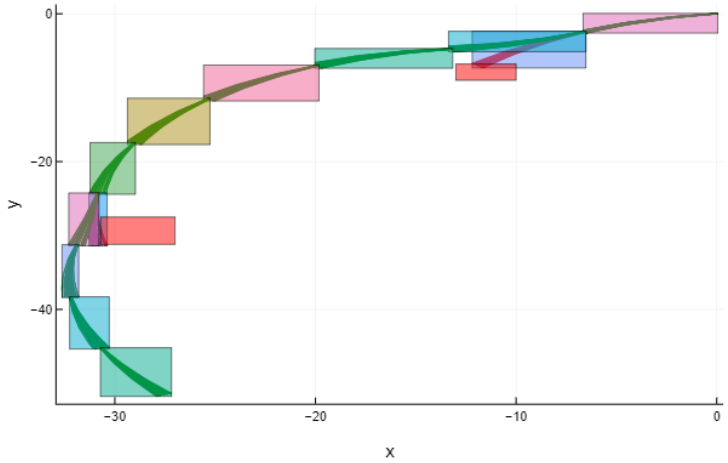
$$\bar{\mathcal{X}}_0 := \{(x, -y, -\theta) \,|\, (x, y, \theta) \in \mathcal{X}_0\}, \text{ and}$$
$$\bar{\mathcal{X}}_u := \{(x, -y, -\theta) \,|\, (x, y, \theta) \in \mathcal{X}_u\}.$$

Property 2. The dynamics are symmetric with respect to the x-axis. More precisely, if $(x(t), y(t), \theta(t))$ is a solution from an initial condition (x_0, y_0, θ_0) with fixed control inputs (v, ω), then $(x(t), -y(t), -\theta(t))$ is a solution from the initial condition $(x_0, -y_0, -\theta_0)$ with fixed control inputs $(v, -\omega)$. This means that the computation of barrier certificates may be restricted to $\omega \geq 0$.

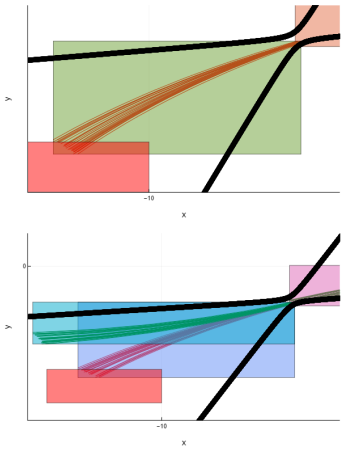
Proof of Property 2 and Remark 2.2. Here we show that if $B(x, y, \theta; v, \omega)$ is a barrier certificate parameterised with (v, ω) for initial and unsafe sets $\mathcal{X}_0, \mathcal{X}_u$, then $B(x, -y, -\theta; v, -\omega)$ is a barrier certificate for initial and unsafe sets $\bar{\mathcal{X}}_0, \bar{\mathcal{X}}_u$, where

$$\bar{\mathcal{X}}_0 := \{(x, -y, -\theta) \,|\, (x, y, \theta) \in \mathcal{X}_0\},$$
$$\bar{\mathcal{X}}_u := \{(x, -y, -\theta) \,|\, (x, y, \theta) \in \mathcal{X}_u\}.$$

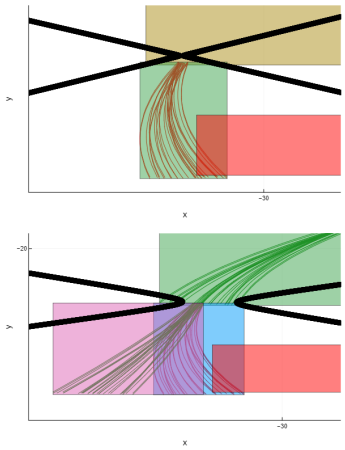
The proof is by direct substitution. We write down the conditions for barrier certificate $B(x, y, \theta; v, \omega)$ for dynamics (5.3) with parameters (θ, ω). We then substitute y with $-y$



(a) Reachable sets for a total time horizon $T = 20$ along with unsafe occurrences at step 2 and step 7; the car starts in the top-right corner.



(b) Original control (top) along with the backup control (bottom) for step 2.



(c) Original control (top) along with the backup control (bottom) for step 7.

Fig. 5.9. Plot of the trajectory of the autonomous driving case study for $T = 20$. Each block corresponds to $T_r = 2$. Red traces correspond to trajectories which lead to unsafe region. Green trajectories are safe. Red sets represent unsafe regions. Black curves are zero-level set of barrier certificates.

and ω with $-\omega$ in all the conditions and use the symmetric properties of the dynamics of the system to retrieve the conditions for the barrier certificate $B(x, -y, -\theta; v, -\omega)$.

We choose the following values for the baseline controller: $\alpha_1 = 0.2$, $\alpha_2 = 3.54$, and $\alpha_3 = 0.06$. We train MetaNN over the domain $v \in [1.5; 10]$, $\omega \in [0; 0.125]$ and $\theta \in [0; 6.5]$ by generating 5414 valid barrier certificates. The trained MetaNN gives correctness of 99.5% over 1000 randomly sampled initial sets of random size. It takes 20 minutes to train the model with CUDA (NVIDIA GeForce RTX 3060 Laptop GPU) for 750000 epochs. Fig. 5.9(a) shows the trajectories of the autonomous car over a time horizon $T = 20$ in which safe reachability is computed every $T_r = 2$, thus corresponding to ten iterations of the procedure presented in Fig. 5.4.

Online Certification of Safe Reach Sets. We now study online correctness guarantees for the safe reach sets constructed by the procedure. We wish to provide a guarantee that for a chosen control action, the vehicle does not collide with an obstacle over some fixed horizon. Note that, as per Theorem 2, any such guarantee is conditional on the vehicle remaining within the working region $\mathcal{X}_b$, and we emphasise that if the vehicle leaves this region within the time horizon the safe action is to stop.

This certification check is equivalent to asserting that the function constructed from the output of MetaNN, which we denote as $B(x)$, is a valid barrier certificate. Define the unsafe set $\mathcal{X}_u$ as the union of all box-shaped obstacles. The initial set $\mathcal{X}_0$ is a box containing the vehicle's true location. The size of this initial set has been selected to account for the uncertainty in the location of obstacles, and to take care of the change in location of the vehicle that occurs during the computation time allocated to the generation of the safe reach sets: this vouches for the online implementation of our scheme. Finally, the working region $\mathcal{X}_b$ is computed as in Sec. 5.2. Due to the trigonometric terms present in the system dynamics Eq. (5.3), dReal is again a suitable choice of SMT solver for checking the validity of the candidate barrier certificates produced by MetaNN.

In Figs. 5.9(b)–5.9(c) we report two examples of control action that would result in unsafe behaviour, and their resolution. Specifically, for cases when the dynamics leads to the unsafe region we use the same controller to avoid obstacles. For this purpose, we exploit the unsafe state as the target region and take a complement of the output angular speed to steer away. We note that each iteration of the algorithm (see Fig. 5.4) takes about 0.03 seconds to execute on a standard laptop for a step $T_r = 2$.

Finally, we note that if dReal is unable to find a counter-example, then we have certified our barrier certificate and hence reach set as being a safe over-approximation of the true reach set, and can safely proceed with the control action over the time horizon. Meanwhile, the existence of a counter-example does not imply that our control action is inherently dangerous, rather just that we cannot assert that it is safe. Our immediate response, as depicted in Figs. 5.9(b)–5.9(c), is to select a different control input to attempt to steer the car safely. However, we could also seek to refine our candidate barrier certificate offline to certify that it is indeed safe. This refinement could be used as feedback to MetaNN for further training.

Comparison with Direct Online Certification of Safe Reach Sets. As demonstrated with the car model case study, our proposed MetaNN enables us to perform controller synthesis together with the online certification. In order to compare the performance of MetaNN *vs.* FOSSIL, we take the same controller synthesised by MetaNN and give it directly to FOSSIL for online certification. Fig. 5.10 demonstrates cumulative distribution of the computational time taken to synthesise 100 barrier certificates directly by FOSSIL. Note, that for the car model under consideration we utilise $T_r = 2$, which means if the barrier certificate is not computed within 2 seconds, online certification is not possible and an auxiliary controller should be employed. We report that only 78% of instances are verified on time. In addition, three cases reach time-out (200 seconds), which means FOSSIL cannot generate certified barrier certificates with the provided

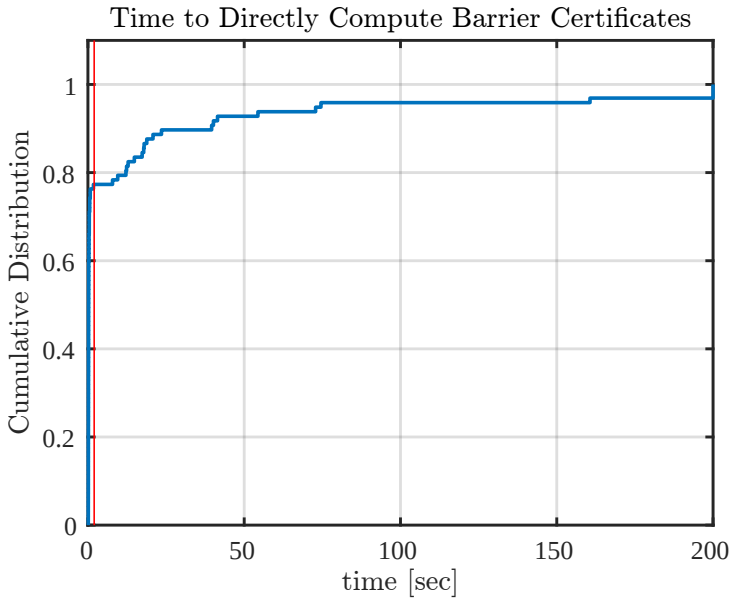


Fig. 5.10. Cumulative distribution of time taken to directly generate barrier certificates via FOSSIL without using MetaNN. The red vertical line shows that FOSSIL synthesised valid barrier certificates only for 78% of instances in the acceptable time bound for the car model (2 seconds).

specifications of the model. In Table 5.1 we report the statistics of the computational times for the certification process directly from FOSSIL. We can observe that FOSSIL demonstrate divergence in computational time for different instances of the model, *e.g.,* maximal computational time for one instance is 160s, while minimal and average just 0.11s and 7.11s respectively.

The computational time for our proposed MetaNN remains constant for each instance. It takes less than 0.7s to extract a barrier certificate from the neural network and verify it in dReal. In addition, we were able to obtain valid barrier certificates for the instances where FOSSIL timed out. Therefore, using MetaNN and offline training, we not only extend the capabilities of barrier certificate verification to synthesis of parameterised controllers, but also increases the success rate of online barrier certificate generation from 78% to 99%. This is at the cost of offline training of MetaNN that takes 10 hours.

	All (s)	Valid (s)
Total	1290.43	690.43
Avg	12.90	7.11
min	0.11	0.11
max	200	160.63

Table 5.1. Statistics of the computational time to synthesise valid barrier certificates for 100 instances on the car model. FOSSIL was not able to find a barrier certificate in 3 instances within the fixed time bound (200s). These three timeout exceptions are excluded in the Valid column.

5.5 Summary

We developed a novel approach for sound computation of reach sets of dynamical systems using level sets of barrier certificates. We developed a MetaNN that is trained offline on samples of barrier certificates and is used online integrated with an SMT solver for ensuring safety. We demonstrate our approach on case studies and an autonomous driving scenario.

In the provided tests we demonstrate the applicability of the presented approach on a wide range of the models. The success rate of using MetaNN for these models is high enough to use in online settings. Online certification of the vehicle dynamics clearly outline the strengths and the potential of using MetaNN in comparison with utilizing FOSSIL directly. We did not compared our framework with other reachability tools as we do not know any publicly available existing tools for online setting.

The algorithm is independent of the selection of controllers or planning algorithm and provide guarantees on the safety of the system. The algorithm is restricted only by (i) scalability of FOSSIL and (ii) training process. In the first case, if FOSSIL can handle larger systems with complex barrier certificates templates, then the presented approach will be stronger too. In the second, it is the utmost importance to learn the MetaNN, so the amount of spurious barrier certificates in online setting will be minimal. Although, such barrier certificates will not lead to unsafe regions due to safety check via SMT-solver, they might increase the time horizon of the dynamics due to use of additional backup or even safe controllers and additional planning.

We showed that the new approach not only extends the capabilities of barrier certificate verification to synthesis of parameterised controllers, but also increases the success rate of online barrier certificate generation from 78% to 99% at the cost of offline training.

Chapter 6

Conclusion

There are multiple challenges in verification of cyber-physical systems. Real-world models require scalable techniques to handle complex non-linearity. In addition, switching between different modes of the system introduces discrete jumps and costly set operations. Such systems may operate in time-constrained settings, thus we need to reason about the safety of such systems in a reasonable time, sometimes instantly. In this thesis, new scalable verification techniques for cyber-physical systems are presented.

6.1 Contribution and current limitations

To begin with, we demonstrated how decompositional reachability can be extended to linear hybrid systems. In addition, we showcased that set operations can be efficiently computed in low dimensions for high-dimensional systems. Decomposed reachability analysis allows us to focus on appropriate subspaces and perform reachability analysis for systems with thousands of variables. The hybrid system should follow a specific structure to get the most out of the proposed decompositional algorithm. In particular, constraints must be defined in low dimensions or be loosely coupled to avoid additional errors due to the wrapping effect. Additionally, the block structure of the system is defined at the very beginning of the hybrid loop algorithm, while for some systems it can be beneficial to change decoupling on-the-fly. The block structure is also defined manually, which means that it is up to the model designer to find the most efficient structure for the analysis.

Next, a new approach to perform reachability analysis of nonlinear systems was presented by utilizing the Koopman operator. It transforms nonlinear models into linear with a systematic way, namely random Fourier features, to generate approximative linear system dynamics. It opened a way to perform reachability analysis on linear systems instead, which is substantially faster. However, due to non-linear observables which are used for a tighter approximation, nonlinear initial sets are introduced. Interval and Taylor model arithmetics as part of zonotope and polynomial zonotope approximation respectively were utilized to efficiently handle nonlinear initial sets created by such linearization. Since some systems can be linearized using Koopman and Fourier without any additional approximation error, it creates a foundation for fast analysis of nonlinear systems. Note,

that the use of SMT solver is a main drawback of this approach, particularly, for systems linearized with Fourier observables due to high nonlinearity. It should be noted that approximation error for such linearization cannot yet be bounded analytically, which means that only statistical safe guarantees on the original system can be provided.

Finally, an algorithm to compute on-the-fly sound reachable sets using barrier certificates was presented. It utilizes two neural networks to produce barrier certificates. These neural networks are connected to SMT solver to produce formal guarantees on the generated zero-level sets. Since the algorithm is independent of the selection of barrier certificates generators for the offline training step, the only restriction is the capabilities of FOSSIL, a toolbox chosen in this work. The online section of the framework relies on a well-trained neural network, which in turn requires sufficient data for the training process. Currently, FOSSIL supports only a limited class of dynamical systems which restricts the class of models our framework supports. In addition, we support only purely continuous systems as discrete jumps bring additional complexity in online settings.

The presented techniques are evaluated on a number of benchmarks and case studies, which are commonly-used for benchmarking in the field. In addition, some of the models are from ARCH competition, since it reports contains results for other tools, so we can pick the best competitors to compare the results against the presented techniques. Thus, we demonstrate the efficiency of the proposed algorithms against state-of-the-art tools.

To sum up, the presented techniques allow verification of wide class of dynamical systems faster in comparison with existing methods. However, each of these methods aims to address a particular subset of systems, while there are systems where other tools might perform better. Thus, the proposed techniques were developed in JuliaReach as it provides all the building pieces for reachability analysis as well as an implementation of multiple state-of-the-art algorithms accessible via a single interface. This could help researchers to easily add or modify these algorithms and evaluate possible improvements, as well as simply run and compare different techniques to analyse the model under consideration, since a single commonly-used framework is yet to be developed for efficient verification of a wide range of dynamical systems. Note, that since ideas presented in this thesis rely on the performance of underlying tools, e.g. SMT solvers and barrier certificate generators. Therefore, improvements to the mentioned tools could significantly improve results for the described techniques.

6.2 Future work

Although the techniques and algorithms presented in this thesis already outperform state-of-the-art tools on a given set of benchmarks, they can be a good basis for future research directions.

For instance, in the benchmarks considered in the evaluation for the decompositional algorithm, it was not necessary to change the block structure when switching between locations. In general, different locations may constrain different dimensions, so tracking the “right” dimensions may be necessary to maintain precision. While it is easy to merge

different blocks, subsequent computations would become more expensive. Hence one may also want to split blocks again for optimal performance. Since this comes with a loss in precision, heuristics for rearranging the block structure, possibly in a refinement loop, are needed. Parallel computations should be considered for the cases when multiple blocks must be computed, as it can result in a faster computing in comparison with the current imperative algorithm.

In addition, we plan to further optimize the reachability analysis algorithm using model transformation via Koopman operator, as well as to create a user-friendly tool in order to participate in the annual nonlinear reachability competition, such as ARCH. Moreover, research is required on how to reduce approximation error and how to measure that error to take it into account during reachability analysis. This will enable the algorithm to provide formal guarantees on the original system. Another interesting direction is how to linearize hybrid systems and how to preserve the mapping between the original and liniarized system in different modes.

We also plan to consider a wider applicability of the approach to generate barrier certificates on-the-fly, particularly in real-time and safety-critical contexts. Additionally, there is a need to test different tools to generate barrier certificates in offline settings to train neural network on more challenging benchmarks. Currently, FOSSIL is the only tool we use in an offline setting and it restricts us, as it does not yet handle a wide range of benchmarks, especially with non-trivial settings. An interesting research direction is to apply this algorithm to the hybrid setting and find an efficient way to perform discrete jumps and interesction with the barrier certificates. This would help to utilize more complex planning and control over the system, which, in turn, will provide more options for model designers. Furthermore, there are more areas of research which are not covered in this thesis. For instance, when we discuss real-life systems we should remember that the industry usually utilizes tools such as MATLAB Simulink/Stateflow to model their systems. Model transformation is required to prepare models to be analyzed via formal verification tools. Currently, researchers from the formal verification field manually transform models to perform formal verification, however, automatic transformation enables everyone with the original, e.g. MATLAB model, to test verification techniques. There are tools, such as SL2SX [151, 121], which could model the basic Simulink system into SpaceEx hybrid automaton representation, however, they lack Stateflow support. An instrument to transform models from industrial tools to verification would boost the use of formal techniques and open new ways of utilizing algorithms with safe guarantees on real-life systems.

Semantically, one of the main differences between industrial modelling tools and formal verification tools, is that the former incorporate “must” semantics, while the latter reason about systems with “may” semantics. In other words, simulation models are deterministic and their transitions must be taken as soon as guard constraints are satisfied, while verification models are nondeterministic and allow the dynamics to flow even with satisfied guard constraints. The problem has been studied in [149] and [150], however, the proposed approach suffer from wrapping effect and could be very convervative in overapproximation.

A promising approach presented in [120] relies on the formulation of Lie derivative and the generation of temporary modes to replicate "must" semantics. However, complex invariants could create a large number of additional modes, which consequently leads to the accumulation of error and increased complexity of the system.

Finally, all the verification methods must be benchmarked. Although HyPro benchmark repository [3] and ARCH competitions [11] are the current main sources of the benchmarks, and Hyst [38] allows to transform models between different verification tools, there is still a lack of a unified suite of models. Given current challenges in verification, such as scalability, verification of non-linear systems, etc., each of the models should clearly outline what each particular model tests and how it is related to real-life problems. Reasonable time frames to perform analysis, as well as safety properties, must be provided to each system.

References

[1] JuliaReach. https://github.com/JuliaReach, 2017, (accessed July 1, 2023).

[2] JuliaReach. https://github.com/JuliaReach, 2017, (accessed July 1, 2023).

[3] Benchmarks of continuous and hybrid systems. https://ths.rwth-aachen.de/research/projects/hypro/benchmarks-of-continuous-and-hybrid-systems/, 2019, (accessed July 1, 2023).

[4] HyPro Benchmark Repository. https://ths.rwth-aachen.de/research/projects/hypro/benchmarks-of-continuous-and-hybrid-systems/, 2020, (accessed July 1, 2023).

[5] Robots for hazardous environments. https://www.bristolroboticslab.com/robots-for-hazardous-environments, 2022, (accessed July 1, 2023).

[6] Robotics and autonomous systems: Defence science and technology capability. https://www.gov.uk/guidance/robotics-and-autonomous-systems-defence-science-and-technology-capability, 2022, (accessed July 1, 2023).

[7] Alessandro Abate, Sergiy Bogomolov, Alec Edwards, Kostiantyn Potomkin, Sadegh Soudjani, and Paolo Zuliani. Safe reach set computation via neural barrier certificates. Submitted for publication. Not yet accessible due to double-blind review rule.

[8] Alessandro Abate, Daniele Ahmed, Alec Edwards, Mirco Giacobbe, and Andrea Peruffo. FOSSIL: A software tool for the formal synthesis of Lyapunov functions and barrier certificates using neural networks. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pages 24:1–24:11. ACM, 2021.

[9] Daniele Ahmed, Andrea Peruffo, and Alessandro Abate. Automated formal synthesis of neural barrier certificates for dynamical models. In *International conference on tools and algorithms for the construction and analysis of systems*.

[10] M. Althoff. Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In *Proceedings of the International Conference on Hybrid Systems: Computation and Control*, pages 173–182, 2013.

[11] M. Althoff, S. Bak, D. Cattaruzza, X. Chen, G. Frehse, R. Ray, and S. Schupp. ARCH-COMP17 category report: Continuous and hybrid systems with linear continuous dynamics. In *Proceedings of the 4th International Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 143–159, 2017.

[12] Matthias Althoff. *Reachability analysis and its application to the safety assessment of autonomous cars*. PhD thesis, Technische Universität München, 2010.

[13] Matthias Althoff. An introduction to CORA 2015. In *Proceedings of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.

[14] Matthias Althoff and John M. Dolan. Online verification of automated road vehicles using reachability analysis. *IEEE Transactions on Robotics*, 30(4):903–918, 2014. doi: 10.1109/TRO.2014.2312453.

[15] Matthias Althoff and Goran Frehse. Combining zonotopes and support functions for efficient reachability analysis of linear systems. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 7439–7446. IEEE, 2016.

[16] Matthias Althoff and Bruce H Krogh. Avoiding geometric intersection operations in reachability analysis of hybrid systems. In *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control*, pages 45–54, 2012.

[17] Matthias Althoff and Bruce H. Krogh. Reachability analysis of nonlinear differential-algebraic systems. *IEEE Trans. Automat. Contr.*, 59(2):371–383, 2014.

[18] Matthias Althoff, Olaf Stursberg, and Martin Buss. Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. *Nonlinear Analysis: Hybrid Systems*, (2):233–249, 2010. URL https://doi.org/10.1016/j.nahs.2009.03.009.

[19] Matthias Althoff, Dmitry Grebenyuk, and Niklas Kochdumper. Implementation of Taylor models in CORA 2018. In *Proceedings of the International Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 145–173, 2018.

[20] Matthias Althoff, Stanley Bak, Marcelo Forets, Goran Frehse, Niklas Kochdumper, Rajarshi Ray, Christian Schilling, and Stefan Schupp. ARCH-COMP19 category report: Continuous and hybrid systems with linear continuous dynamics. In *Proceedings of the International Workshop on Applied Verification of Continuous and Hybrid Systems*, pages 14–40, 2019.

[21] Matthias Althoff, Stanley Bak, Marcelo Forets, Goran Frehse, Niklas Kochdumper, Rajarshi Ray, Christian Schilling, and Stefan Schupp. ARCH-COMP19 category report: Continuous and hybrid systems with linear continuous dynamics. In *ARCH19*, pages 14–40, 2019.

[22] Matthias Althoff, Goran Frehse, and Antoine Girard. Set propagation techniques for reachability analysis. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1), 2021.

[23] Rajeev Alur. Formal verification of hybrid systems. In *2011 Proceedings of the Ninth ACM International Conference on Embedded Software (EMSOFT)*, pages 273–278. IEEE, 2011.

[24] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, volume 736, pages 209–229, 1992.

[25] Aaron D. Ames, Xiangru Xu, Jessy W. Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Trans. Autom. Control.*, 62(8):3861–3876, 2017.

[26] Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control Barrier Functions: Theory and Applications. *arXiv:1903.11199*, March 2019.

[27] Dennis S Arnon, George E Collins, and Scott McCallum. Cylindrical algebraic decomposition i: The basic algorithm. *SIAM Journal on Computing*, 13(4):865–877, 1984.

[28] Eugene Asarin and Thao Dang. Abstraction by projection and application to multi-affine systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 32–47, 2004.

[29] Eugene Asarin, Olivier Bournez, Thao Dang, and Oded Maler. Approximate reachability analysis of piecewise-linear dynamical systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 20–31. Springer, 2000.

[30] Eugene Asarin, Thao Dang, and Antoine Girard. Reachability analysis of nonlinear systems using conservative approximation. In *International Workshop on Hybrid Systems: Computation and Control*, pages 20–35. Springer, 2003.

[31] Eugene Asarin, Thao Dang, Goran Frehse, Antoine Girard, Colas Le Guernic, and Oded Maler. Recent progress in continuous and hybrid reachability analysis. In *2006 IEEE Conference on Computer Aided Control System Design*, pages 1582–1587. IEEE, 2006.

[32] Eugene Asarin, Thao Dang, and Antoine Girard. Hybridization methods for the analysis of nonlinear systems. *Acta Informatica*, 43(7):451–476, 2007.

[33] Radhakisan Baheti and Helen Gill. Cyber-physical systems. volume 12, pages 161–166, 2011.

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

[35] Stanley Bak and Parasara Sridhar Duggirala. Hylaa: A tool for computing simulation-equivalent reachability for linear systems. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, HSCC '17, 2017.

[36] Stanley Bak and et al. Numerical verification of affine systems with up to a billion dimensions. In *Proceedings of the International Conference on Hybrid Systems: Computation and Control*, pages 23–32, 2019.

[37] Stanley Bak and et al. Reachability of black-box nonlinear systems after Koopman operator linearization. In *Proceedings of the International Conference on Analysis and Design of Hybrid Systems*, pages 253–258, 2021.

[38] Stanley Bak, Sergiy Bogomolov, and Taylor T Johnson. HYST: a source transformation and translation tool for hybrid automaton models. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pages 128–133. ACM, 2015.

[39] Stanley Bak, Sergiy Bogomolov, Thomas A Henzinger, Taylor T Johnson, and Pradyot Prakash. Scalable static hybridization methods for analysis of nonlinear systems. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pages 155–164, 2016.

[40] Stanley Bak, Sergiy Bogomolov, and Matthias Althoff. Time-triggered conversion of guards for reachability analysis of hybrid automata. In *Formal Modeling and Analysis of Timed Systems: 15th International Conference, FORMATS 2017, Berlin, Germany, September 5–7, 2017, Proceedings 15*, pages 133–150, 2017.

[41] Stanley Bak, Hoang-Dung Tran, and Taylor T. Johnson. Numerical verification of affine systems with up to a billion dimensions. In *Proceedings of the 22Nd ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '19, pages 23–32, New York, NY, USA, 2019. ACM. doi: 10.1145/3302504.3311792.

[42] Stanley Bak, Hoang-Dung Tran, and Taylor T Johnson. Numerical verification of affine systems with up to a billion dimensions. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 23–32, 2019.

[43] Stanley Bak, Sergiy Bogomolov, Parasara Sridhar Duggirala, Adam R Gerlach, and Kostiantyn Potomkin. Reachability of black-box nonlinear systems after koopman operator linearization. *IFAC-PapersOnLine*, 54(5):253–258, 2021.

[44] Stanley Bak, Sergiy Bogomolov, Brandon Hencey, Niklas Kochdumper, Ethan Lew, and Kostiantyn Potomkin. Reachability of Koopman linearized systems using random Fourier feature observables and polynomial zonotope refinement. In *International Conference on Computer Aided Verification*, pages 490–510. Springer, 2022.

[45] Luis Barba and Stefan Langerman. Optimal detection of intersections between convex polyhedra. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2015.

[46] Clark Barrett and Cesare Tinelli. Cvc3. In *International Conference on Computer Aided Verification*, pages 298–302. Springer, 2007.

[47] Alberto Bemporad and Manfred Morari. Verification of hybrid systems via mathematical programming. In *International Workshop on Hybrid Systems: Computation and Control*, pages 31–45. Springer, 1999.

[48] Johan Bengtsson, Kim Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. UPPAAL — a tool suite for automatic verification of real-time systems. In *International hybrid systems workshop*, pages 232–243. Springer, 1995.

[49] Frédéric Benhamou and Laurent Granvilliers. Continuous and interval constraints. *Handbook of Constraint Programming*, page 569, 2006.

[50] Veronica Biagini, Milos Subasic, Alexandre Oudalov, and Jochen Kreusel. The autonomous grid: Automation, intelligence and the future of power systems. *Energy Research & Social Science*, 65:101460, 2020. ISSN 2214-6296.

[51] Sergiy Bogomolov, Goran Frehse, Mirco Giacobbe, and Thomas A. Henzinger. Counterexample-guided refinement of template polyhedra. In *23rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 589–606. Springer, 2017.

[52] Sergiy Bogomolov, Marcelo Forets, Goran Frehse, Frédéric Viry, Andreas Podelski, and Christian Schilling. Reach set approximation through decomposition with low-dimensional sets and high-dimensional matrices. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week)*, pages 41–50, 2018.

[53] Sergiy Bogomolov, Marcelo Forets, Goran Frehse, Kostiantyn Potomkin, and Christian Schilling. JuliaReach: a toolbox for set-based reachability. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 39–44, 2019.

[54] Sergiy Bogomolov, Marcelo Forets, Goran Frehse, Kostiantyn Potomkin, and Christian Schilling. Reachability analysis of linear hybrid systems via block decomposition. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):4018–4029, 2020.

[55] Christopher W Brown. Qepcad b: a program for computing with semi-algebraic sets using cads. *ACM SIGSAM Bulletin*, 37(4):97–108, 2003.

[56] Steven L. Brunton, Bingni W. Brunton, Joshua L. Proctor, and J. Nathan Kutz. *Koopman Invariant Subspaces and Finite Linear Representations of Nonlinear Dynamical Systems for Control*, volume 11. Public Library of Science. doi: 10.1371/journal.pone.0150171.

[57] Lei Bu and Xuandong Li. Path-oriented bounded reachability analysis of composed linear hybrid systems. *International journal on software tools for technology transfer*, (4):307–317, 2011.

[58] Marko Budišić, Ryan Mohr, and Igor Mezić. Applied Koopmanism. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(4):047510. ISSN 1054-1500, 1089-7682. doi: 10.1063/1.4772195.

[59] Nicole Chan and Sayan Mitra. Verifying safety of an autonomous spacecraft rendezvous mission. In *International Workshop on Applied Verification of Continuous and Hybrid Systems*, 2017.

[60] Mo Chen, Sylvia L. Herbert, and Claire J. Tomlin. Exact and efficient Hamilton-Jacobi guaranteed safety analysis via system decomposition. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 87–92. IEEE, 2017.

[61] X. Chen and et al. Flow*: An analyzer for nonlinear hybrid systems. In *Proceedings of the International Conference on Computer-Aided Verification*, pages 258–263, 2013.

[62] Xin Chen and Sriram Sankaranarayanan. Decomposed reachability analysis for nonlinear systems. In *2016 IEEE Real-Time Systems Symposium (RTSS)*, pages 13–24. IEEE, 2016.

[63] Xin Chen and Sriram Sankaranarayanan. Reachability analysis for cyber-physical systems: Are we there yet? In *NASA Formal Methods Symposium*, pages 109–130. Springer, 2022.

[64] Xin Chen, Erika Abraham, and Sriram Sankaranarayanan. Taylor model flowpipe construction for nonlinear hybrid systems. In *2012 IEEE 33rd Real-Time Systems Symposium*, pages 183–192. IEEE, 2012.

[65] Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. Flow*: An analyzer for nonlinear hybrid systems. In *International Conference on Computer Aided Verification*, pages 258–263. Springer, 2013.

[66] Alongkrit Chutinan and Bruce H Krogh. Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations. In *International workshop on hybrid systems: computation and control*, pages 76–90. Springer, 1999.

[67] Alongkrit Chutinan and Bruce H. Krogh. Computational techniques for hybrid system verification. *IEEE transactions on automatic control*, 48(1):64–75, 2003.

[68] Alessandro Cimatti, Sergio Mover, and Stefano Tonetta. SMT-based verification of hybrid systems. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[69] Edmund M Clarke and Paolo Zuliani. Statistical model checking for cyber-physical systems. In *International Symposium on Automated Technology for Verification and Analysis*, pages 1–12. Springer, 2011.

[70] Liyun Dai, Ting Gan, Bican Xia, and Naijun Zhan. Barrier certificates revisited. *Journal of Symbolic Computation*, 80:62–86, 2017.

[71] Thao Dang, Oded Maler, and Romain Testylier. Accurate hybridization of nonlinear systems. In *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*, pages 11–20, 2010.

[72] Leonardo de Moura and Nikolaj Bjørner. Z3: An Efficient SMT Solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 4963, pages 337–340. 2008. ISBN 978-3-540-78799-0 978-3-540-78800-3.

[73] Anthony M DeGennaro and Nathan M Urban. Scalable extended dynamic mode decomposition using random kernel approximation. *SIAM Journal on Scientific Computing*, 41(3):A1482–A1499, 2019.

[74] Jyotirmoy V Deshmukh and Sriram Sankaranarayanan. Formal techniques for verification and testing of cyber-physical systems. *Design Automation of Cyber-Physical Systems*, pages 69–105, 2019.

[75] Andreas Dolzmann and Thomas Sturm. Redlog: Computer algebra meets computer logic. *ACM Sigsam Bulletin*, 31(2):2–9, 1997.

[76] Asen L. Dontchev. Time-scale decomposition of the reachable set of constrained linear systems. *Mathematics of Control, Signals and Systems*, 5(3):327–340, 1992.

[77] Alexandre Donzé and Oded Maler. Systematic simulation using sensitivity analysis. In *International Workshop on Hybrid Systems: Computation and Control*, pages 174–189. Springer, 2007.

[78] Laurent Doyen, Goran Frehse, George J. Pappas, and André Platzer. *Verification of Hybrid Systems*, pages 1047–1110. Springer International Publishing, Cham, 2018. ISBN 978-3-319-10575-8. doi: 10.1007/978-3-319-10575-8_30.

[79] Parasara Sridhar Duggirala, Sayan Mitra, Mahesh Viswanathan, and Matthew Potok. 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.

[80] Michael Fisher, Viviana Mascardi, Kristin Yvonne Rozier, Bernd-Holger Schlingloff, Michael Winikoff, and Neil Yorke-Smith. Towards a framework for certification of reliable autonomous systems. *Autonomous Agents and Multi-Agent Systems*, 35(1): 1–65, 2021.

[81] Martin Fränzle, Christian Herde, Tino Teige, Stefan Ratschan, and Tobias Schubert. Efficient solving of large nonlinear arithmetic constraint systems with complex boolean structure. *Journal on Satisfiability, Boolean Modeling and Computation*, 1 (3-4):209–236, 2006.

[82] Martin Fränzle, Mingshuai Chen, and Paul Kröger. In memory of Oded Maler: Automatic reachability analysis of hybrid-state automata. *ACM SIGLOG News*, 6 (1):19–39, 2019.

[83] Goran Frehse and Rajarshi Ray. Flowpipe-guard intersection for reachability computations with support functions. In *The IFAC Conference on Analysis and Design of Hybrid Systems*, number 9, pages 94–101, 2012.

[84] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable verification of hybrid systems. In *Proceedings of the International Conference on Computer Aided Verification*, pages 379–395. Springer, 2011.

[85] Goran Frehse, Rajat Kateja, and Colas Le Guernic. Flowpipe approximation and clustering in space-time. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pages 203–212, 2013.

[86] Goran Frehse, Sergiy Bogomolov, Marius Greitschus, Thomas Strump, and Andreas Podelski. Eliminating spurious transitions in reachability with support functions. In *18th International Conference on Hybrid Systems: Computation and Control (HSCC 2015)*, pages 149–158. ACM, 2015.

[87] Goran Fedja Frehse. *Compositional verification of hybrid systems using simulation relations*. PhD thesis, Radboud Universiteit Nijmegen, 2005.

[88] Komei Fukuda, Thomas M. Liebling, and Christine Lütolf. Extended convex hull. *Comput. Geom.*, (1-2):13–23, 2001.

[89] Sicun Gao, Malay Ganai, Franjo Ivančić, Aarti Gupta, Sriram Sankaranarayanan, and Edmund M Clarke. Integrating ICP and LRA solvers for deciding nonlinear real arithmetic problems. In *Formal Methods in Computer Aided Design*, pages 81–89. IEEE, 2010.

[90] Sicun Gao, Soonho Kong, and Edmund M. Clarke. dReal: An SMT Solver for Nonlinear Theories over the Reals. In *Automated Deduction–CADE-24: 24th International Conference on Automated Deduction, Lake Placid, NY, USA, June 9-14, 2013. Proceedings 24*, volume 7898, pages 208–214. Springer, 2013. ISBN 978-3-642-38573-5 978-3-642-38574-2.

[91] Sicun Gao, James Kapinski, Jyotirmoy V. Deshmukh, Nima Roohi, Armando Solar-Lezama, Nikos Aréchiga, and Soonho Kong. Numerically-robust inductive proof rules for continuous dynamical systems. In *CAV*, volume 11562 of *LNCS*, pages 137–154. Springer, 2019.

[92] Adam R. Gerlach, Andrew Leonard, Jonathan Rogers, and Chris Rackauckas. The Koopman Expectation: An Operator Theoretic Method for Efficient Analysis and Optimization of Uncertain Hybrid Dynamical Systems. URL http://arxiv.org/abs/2008.08737.

[93] Vladimír Gerlich, Kateřina Sulovská, and Martin Zálešák. COMSOL multiphysics validation as simulation software for heat transfer calculation in buildings: Building simulation software validation. *Measurement*, 46(6):2003–2012, 2013.

[94] Antoine Girard. Reachability of uncertain linear systems using zonotopes. In *International Workshop on Hybrid Systems: Computation and Control*, pages 291–305. Springer, 2005.

[95] Antoine Girard and Colas Le Guernic. Efficient reachability analysis for linear systems using support functions. *IFAC Proceedings Volumes*, 41(2):8966 – 8971, 2008.

[96] Antoine Girard and Colas Le Guernic. Zonotope/hyperplane intersection for hybrid systems reachability analysis. In *International Workshop on Hybrid Systems: Computation and Control*, pages 215–228, 2008.

[97] Antoine Girard and Colas Le Guernic. Zonotope/hyperplane intersection for hybrid systems reachability analysis. In *International Workshop on Hybrid Systems: Computation and Control*, pages 215–228. Springer, 2008.

[98] Elena V. Goncharova and Alexander I. Ovseevich. Asymptotics for singularly perturbed reachable sets. In *Large-Scale Scientific Computing: 7th International Conference, LSSC 2009, Sozopol, Bulgaria,. Revised Papers 7*, pages 280–285, 2009.

[99] Laurent Granvilliers and Frédéric Benhamou. Algorithm 852: Realpaver: an interval solver using constraint satisfaction techniques. *ACM Transactions on Mathematical Software (TOMS)*, 32(1):138–156, 2006.

[100] Mark R. Greenstreet and Ian Mitchell. Reachability analysis using polygonal projections. In *Hybrid Systems: Computation and Control: Second International Workshop, HSCC'99 Berg en Dal, The Netherlands, March 29–31, 1999 Proceedings 2*, pages 103–116. Springer, 1999.

[101] Colas Le Guernic and Antoine Girard. Reachability analysis of hybrid systems using support functions. In *International Conference on Computer Aided Verification*, pages 540–554, 2009.

[102] Amit Gurung, Rajarshi Ray, Ezio Bartocci, Sergiy Bogomolov, and Radu Grosu. Parallel reachability analysis of hybrid systems in XSpeed. *International Journal on Software Tools for Technology Transfer*, 21(4):401–423, 2019.

[103] H. Guéguen and J. Zaytoon. On the formal verification of hybrid systems. *Control Engineering Practice*, 12(10):1253–1267, 2004. ISSN 0967-0661. Analysis and Design of Hybrid Systems.

[104] Abdullah Omar Hamadeh and Jorge M. Goncalves. Reachability analysis of continuous-time piecewise affine systems. *Automatica*, (12):3189–3194, 2008.

[105] Zhi Han and Bruce H. Krogh. Reachability analysis of large-scale affine systems using low-dimensional polytopes. In *Proceedings of the International Conference on Hybrid Systems: Computation and Control*, pages 287–301, 2006.

[106] Zhi Han and Bruce H Krogh. Reachability analysis of nonlinear systems using trajectory piecewise linearized models. In *2006 American Control Conference*, pages 6–pp. IEEE, 2006.

[107] Byron Heersink, Pape Sylla, and Michael A. Warren. Formal verification of octorotor flight envelope using barrier functions and SMT solving. *arXiv:2107.00612*, July 2021.

[108] T. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57:94–124, 1998.

[109] Thomas A. Henzinger. The theory of hybrid automata. In *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, pages 278–292. IEEE Computer Society, 1996.

[110] Daniel Herr, Moritz Godel, Ryan Perkins, Luke Pate, and Teresa Hall. The economic impact of robotics & autonomous systems across UK sectors. *Robotics and autonomous systems: the economic impact across UK sectors*, 2020.

[111] Dorit S. Hochbaum and Joseph Naor. Simple and fast algorithms for linear and integer programs with two variables per inequality. *SIAM Journal on Computing*, (6):1179–1192, 1994.

[112] Fabian Immler, Matthias Althoff, Luis Benet, Alexandre Chapoutot, Xin Chen, Marcelo Forets, Luca Geretti, Niklas Kochdumper, David P Sanders, and Christian Schilling. ARCH-COMP19 category report: Continuous and hybrid systems with nonlinear dynamics. In *Proceedings of the International Workshop on Applied Verification of Continuous and Hybrid Systems*, pages 41–61, 2019.

[113] Jeff C Jensen, Danica H Chang, and Edward A Lee. A model-based design methodology for cyber-physical systems. In *2011 7th international wireless communications and mobile computing conference*, pages 1666–1671. IEEE, 2011.

[114] Wanxin Jin, Zhaoran Wang, Zhuoran Yang, and Shaoshuai Mou. Neural Certificates for Safe Control Policies. *arXiv preprint arXiv:2006.08465*, 2020.

[115] Christopher K. R. T. Jones. Whither Applied Nonlinear Dynamics? In Björn Engquist and Wilfried Schmid, editors, *Mathematics Unlimited — 2001 and Beyond*, pages 631–645. Springer. ISBN 978-3-642-56478-9.

[116] James Kapinski, Jyotirmoy V. Deshmukh, Sriram Sankaranarayanan, and Nikos Arechiga. Simulation-guided Lyapunov analysis for hybrid dynamical systems. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pages 133–142, 2014.

[117] Akitoshi Kawamura. Lipschitz continuous ordinary differential equations are polynomial-space complete. In *2009 24th Annual IEEE Conference on Computational Complexity*, pages 149–160. IEEE, 2009.

[118] Shahab Kaynama and Meeko Oishi. Overapproximating the reachable sets of LTI systems through a similarity transformation. In *Proceedings of the 2010 American Control Conference*, pages 1874–1879, 2010.

[119] Shahab Kaynama and Meeko Oishi. Complexity reduction through a Schur-based decomposition for reachability analysis of linear time-invariant systems. *International J. Control*, (1):165–179, 2011.

[120] Nikolaos Kekatos. *Formal verification of cyber-physical systems in the industrial model-based design process*. PhD thesis, Université Grenoble Alpes, 2018.

[121] Nikolaos Kekatos, Marcelo Forets, and Goran Frehse. Constructing verification models of nonlinear simulink systems via syntactic hybridization. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1788–1795. IEEE, 2017.

[122] Dae-Won Kim and et al. Evaluation of the performance of clustering algorithms in kernel-induced feature space. *Pattern Recognition*, 38(4):607–611, 2005.

[123] Edda Klipp and et al. *Systems Biology in Practice: Concepts, Implementation and Application*. John Wiley & Sons, 2005.

[124] N. Kochdumper and M. Althoff. Sparse polynomial zonotopes: A novel set representation for reachability analysis. *Transactions on Automatic Control*, 66(9):4043–4058, 2021.

[125] Niklas Kochdumper and et al. Utilizing dependencies to obtain subsets of reachable sets. In *Proceedings of the International Conference on Hybrid Systems: Computation and Control*, 2020.

[126] Soonho Kong, Sicun Gao, Wei Chen, and Edmund Clarke. dReach: δ-reachability analysis for hybrid systems. In *International Conference on TOOLS and Algorithms for the Construction and Analysis of Systems*, pages 200–205. Springer, 2015.

[127] Bernard O Koopman. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.

[128] Anna-Kathrin Kopetzki, Bastian Schürmann, and Matthias Althoff. Methods for order reduction of zonotopes. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 5626–5633. IEEE, 2017.

[129] Milan Korda and Igor Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. 93:149–160. ISSN 0005-1098. doi: 10.1016/j.automatica.2018.03.046.

[130] Brian Korver. The Monte Carlo method and software reliability theory. *Technical Report PSU TR 94–1, Feb*, 1994.

[131] Alexander B Kurzhanski and Pravin Varaiya. Ellipsoidal techniques for reachability analysis. In *International Workshop on Hybrid Systems: Computation and Control*, pages 202–214. Springer, 2000.

[132] J. Nathan Kutz, Steven L. Brunton, Bingni W. Brunton, and Joshua L. Proctor. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. SIAM-Society for Industrial and Applied Mathematics, 2016. ISBN 978-1-61197-449-2.

[133] Rom Langerak, Jan Willem Polderman, and Tomas Krilavičius. Stability analysis for hybrid automata using conservative gains. *IFAC Proceedings Volumes*, 36(6): 337–342, 2003.

[134] Andrzej Lasota and Michael C. Mackey. *Chaos, Fractals, and Noise: Stochastic Aspects of Dynamics*. Springer Science & Business Media. ISBN 978-1-4612-4286-4.

[135] Averill M Law, W David Kelton, and W David Kelton. *Simulation modeling and analysis*, volume 3. McGraw-Hill New York, 2007.

[136] Colas Le Guernic. *Reachability analysis of hybrid systems with linear continuous dynamics*. PhD thesis, Université Grenoble 1 - Joseph Fourier, 2009. URL https://tel.archives-ouvertes.fr/tel-00422569.

[137] Colas Le Guernic and Antoine Girard. Reachability analysis of linear systems using support functions. *Nonlinear Analysis: Hybrid Systems*, 4(2):250–262, 2010.

[138] Huei-Huang Lee. *Finite element simulations with ANSYS Workbench 18*. SDC publications, 2018.

[139] Benoît Legat, Robin Deits, Oliver Evans, Gustavo Goretkin, Twan Koolen, Joey Huchette, Daisuke Oyama, Marcelo Forets, guberger, Robert Schwarz, Henrique Ferrolho, Elliot Saba, and Chase Coleman. JuliaPolyhedra/Polyhedra.jl: v0.6.5, 2020.

[140] Dongxu Li, Stanley Bak, and Sergiy Bogomolov. Reachability analysis of nonlinear systems using hybridization and dynamics scaling. In *18th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS 2020)*, volume 12288 of *LNCS*, pages 265–282. Springer, 2020.

[141] Ibtissem Ben Makhlouf and Stefan Kowalewski. Networked cooperative platoon of vehicles for testing methods and verification tools. In *Proceedings of ARCH14-15. 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems*, pages 37–42, 2014.

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

[143] Alexandre Mauroy, Igor Mezić, and Yoshihiko Susuki, editors. *The Koopman Operator in Systems and Control: Concepts, Methodologies, and Applications*. Springer, 1st ed. 2020 edition.

[144] Alexandre Mauroy, Yoshihiko Susuki, and Igor Mezić. Introduction to the Koopman Operator in dynamical systems and control theory. In *The Koopman Operator in Systems and Control*, pages 3–33. Springer, 2020.

[145] David W McKee, Stephen J Clement, Jaber Almutairi, and Jie Xu. Survey of advances and challenges in intelligent autonomy for distributed cyber-physical systems. *CAAI Transactions on Intelligence Technology*, 3(2):75–82, 2018.

[146] Peter McMullen. The maximal number of faces of a convex polytope. *Mathematika*, pages 179–184, 1970.

[147] Pierre-Jean Meyer, Alex Devonport, and Murat Arcak. *Interval Reachability Analysis: Bounding Trajectories of Uncertain Systems with Boxes for Control and Verification*. Springer Nature, 2021.

[148] Joseph J Meyers, Andrew M Leonard, Jonathan D Rogers, and Adam R Gerlach. Koopman operator approach to optimal control selection under uncertainty. In *2019 American Control Conference (ACC)*, pages 2964–2971. IEEE, 2019.

[149] Stefano Minopoli and Goran Frehse. Non-convex invariants and urgency conditions on linear hybrid automata. In *Formal Modeling and Analysis of Timed Systems: 12th International Conference, FORMATS 2014, Florence, Italy, September 8-10, 2014. Proceedings 12*, pages 176–190. Springer, 2014.

[150] Stefano Minopoli and Goran Frehse. From simulation models to hybrid automata using urgency and relaxation. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pages 287–296, 2016.

[151] Stefano Minopoli and Goran Frehse. Sl2sx translator: from simulink to spaceex models. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pages 93–98, 2016.

[152] Ian M Mitchell. Comparing forward and backward reachability as tools for safety analysis. In *International Workshop on Hybrid Systems: Computation and Control*, pages 428–443. Springer, 2007.

[153] Ian M. Mitchell and Claire Tomlin. Overapproximating reachable sets by Hamilton-Jacobi projections. *J. Sci. Comput.*, 19(1-3):323–346, 2003.

[154] Abhinav Narasingam and Joseph Sang-Il Kwon. Koopman Lyapunov-based model predictive control of nonlinear chemical process systems. 65(11):e16743. ISSN 1547-5905. doi: 10.1002/aic.16743.

[155] Nedialko S. Nedialkov and Martin Von Mohrenschildt. Rigorous simulation of hybrid dynamic systems with symbolic and interval methods. In *Proceedings of the American Control Conference*, pages 140–147, 2002.

[156] Samuel E Otto and Clarence W Rowley. Koopman operators for estimation and control of dynamical systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:59–87, 2021.

[157] A. Papachristodoulou and S. Prajna. On the construction of Lyapunov functions using the sum of squares decomposition. In *On the Construction of Lyapunov Functions Using the Sum of Squares Decomposition*, volume 3, pages 3482–3487, 2002.

[158] Doron Peled, Moshe Y Vardi, and Mihalis Yannakakis. Black box checking. In *Formal Methods for Protocol Engineering and Distributed Systems*, pages 225–240. Springer, 1999.

[159] André Platzer. *Logical Foundations of Cyber-Physical Systems*. Springer, 2018. ISBN 978-3-319-63587-3.

[160] Stephen Prajna. Barrier certificates for nonlinear model validation. *Automatica*, 42 (1):117–126, 2006. ISSN 0005-1098.

[161] Stephen Prajna and Ali Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *International Workshop on Hybrid Systems: Computation and Control*, pages 477–492. Springer, 2004.

[162] Stephen Prajna, Ali Jadbabaie, and George J. Pappas. A Framework for Worst-Case and Stochastic Safety Verification Using Barrier Certificates. *IEEE Transactions on Automatic Control*, 52(8):1415–1428, August 2007. ISSN 0018-9286.

[163] The Associated Press. Nearly 400 car crashes in 11 months involved automated tech, companies tell regulators, Jun 2022, (accessed July 1, 2023). URL https://www.npr.org/2022/06/15/1105252793/nearly-400-car-crashes-in-11-months-involved-automated-tech-companies-tell-regul?t=1656523463105.

[164] Christopher Rackauckas and Qing Nie. DifferentialEquations.jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia. 5(1):15. ISSN 2049-9647. doi: 10.5334/jors.151. URL http://openresearchsoftware.metajnl.com/article/10.5334/jors.151/.

[165] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Proceedings of the International Conference on Neural Information Processing Systems*, pages 1177–1184, 2007.

[166] RH Rand and PJ Holmes. Bifurcation of periodic motions in two weakly coupled Van der Pol oscillators. *International Journal of nonlinear Mechanics*, 15(4-5):387–399, 1980.

[167] Stefan Ratschan. Simulation based computation of certificates for safety of dynamical systems. In *Formal Modeling and Analysis of Timed Systems: 15th International Conference, FORMATS 2017, Berlin, Germany, September 5–7, 2017, Proceedings 15*, volume 10419 of *LNCS*, pages 303–317, 2017.

[168] Stefan Ratschan and Zhikun She. Safety verification of hybrid systems by constraint propagation based abstraction refinement. In *International workshop on hybrid systems: Computation and control*, pages 573–589. Springer, 2005.

[169] Stefan Ratschan and Zhikun She. Safety verification of hybrid systems by constraint propagation-based abstraction refinement. *ACM Transactions on Embedded Computing Systems (TECS)*, 6(1):1–23, 2007.

[170] Andreas Rauh and et al. Carleman linearization for control and for state and disturbance estimation of nonlinear dynamical processes. In *Proceedings of the International Conference on Methods and Models in Automation and Robotics*, pages 455–460, 2009.

[171] Otto E Rössler. An equation for continuous chaos. *Physics Letters A*, 57(5):397–398, 1976.

[172] Walter Rudin. *Fourier Analysis on Groups*. Courier Dover Publications, 2017.

[173] Ricardo Sanfelice, David Copp, and Pablo Nanez. A toolbox for simulation of hybrid systems in Matlab/Simulink: Hybrid Equations (HyEQ) Toolbox. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pages 101–106. ACM, 2013.

[174] Sriram Sankaranarayanan. Change-of-bases abstractions for nonlinear systems. *arXiv preprint arXiv:1204.4347*, 2012.

[175] Peter J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28. ISSN 1469-7645, 0022-1120. doi: 10.1017/S0022112010001217.

[176] Bernhard Schölkopf and Alex J Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2018.

[177] Stefan Schupp, Erika Ábrahám, Xin Chen, Ibtissem Ben Makhlouf, Goran Frehse, Sriram Sankaranarayanan, and Stefan Kowalewski. Current challenges in the verification of hybrid systems. In *International Workshop on Design, Modeling, and Evaluation of Cyber Physical Systems*, pages 8–24. Springer, 2015.

[178] Stefan Schupp, Erika Abraham, Ibtissem Ben Makhlouf, and Stefan Kowalewski. HyPro: A C++ library of state set representations for hybrid systems reachability analysis. In *NASA Formal Methods Symposium*, pages 288–294. Springer, 2017.

[179] Stefan Schupp, Johanna Nellen, and Erika Ábrahám. Divide and conquer: Variable set separation in hybrid systems reachability analysis. In *QAPL@ETAPS*, pages 1–14, 2017.

[180] Yassamine Seladji and Olivier Bouissou. Numerical abstract domain using support functions. In *NASA Formal Methods*, pages 155–169. Springer, 2013.

[181] Christoffer Sloth. *Formal Verification of Continuous Systems*. PhD thesis, Department of Computer Science, The Faculties of Engineering, Science, and Medicine, Aalborg University, 2012.

[182] Jorge Sotomayor and et al. Bifurcation analysis of the Watt governor system. *Computational & Applied Mathematics*, 26(1):19–44, 2007.

[183] Mohit Srinivasan, Amogh Dabholkar, Samuel Coogan, and Patricio A. Vela. Synthesis of control barrier functions using a supervised machine learning approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 7139–7145. IEEE, 2020.

[184] Hiroyuki Takeda and et al. Kernel regression for image processing and reconstruction. *Transactions on Image Processing*, 16(2):349–366, 2007.

[185] Ankur Taly and Ashish Tiwari. Deductive verification of continuous dynamical systems. In Ravi Kannan and K. Narayan Kumar, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2009, December 15-17, 2009, IIT Kanpur, India*, volume 4 of *LIPIcs*, pages 383–394. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2009.

[186] Abenezer G Taye, Josh Bertram, Chuchu Fan, and Peng Wei. Reachability based online safety verification for high-density urban air mobility trajectory planning. In *AIAA AVIATION 2022 Forum*, page 3542, 2022.

[187] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*, pages 303–314, 2018.

[188] Ashish Tiwari. Approximate reachability for linear systems. In *International Workshop on Hybrid Systems: Computation and Control*, volume 2623 of *LNCS*, pages 514–525, 2003.

[189] Claire J Tomlin, Ian Mitchell, Alexandre M Bayen, and Meeko Oishi. Computational techniques for the verification of hybrid systems. *Proceedings of the IEEE*, 91(7): 986–1001, 2003.

[190] Jonathan H. Tu, Clarence W. Rowley, Dirk M. Luchtenburg, Steven L. Brunton, and J. Nathan Kutz. On dynamic mode decomposition: Theory and applications. *Journal of Computational Dynamics*, 1(2):391, 2014. doi: 10.3934/jcd.2014.1.391.

[191] Devis Tuia and et al. Learning relevant image features with multiple-kernel classification. *Transactions on Geoscience and Remote Sensing*, 48(10):3780–3791, 2010.

[192] Cumhur Erkan Tuncali, James Kapinski, Hisahiro Ito, and Jyotirmoy V Deshmukh. Reasoning about safety of learning-enabled components in autonomous cyber-physical systems. In *Proceedings of the 55th Annual Design Automation Conference*, pages 1–6, 2018.

[193] Qiuye Wang, Mingshuai Chen, Bai Xue, Naijun Zhan, and Joost-Pieter Katoen. Synthesizing invariant barrier certificates via difference-of-convex programming. In *International Conference on Computer Aided Verification*, pages 443–466, 2021.

[194] M. Wetzlinger and et al. Adaptive parameter tuning for reachability analysis of linear systems. In *Proceedings of the International Conference on Decision and Control*, pages 5145–5152, 2020.

[195] Matthew O Williams and et al. A data–driven approximation of the Koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.

[196] Matthew O Williams and et al. A kernel-based method for data-driven Koopman spectral analysis. *Journal of Computational Dynamics*, 2(2):247–265, 2015.

[197] Matthew O. Williams, Ioannis G. Kevrekidis, and Clarence W. Rowley. A Data–Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346. ISSN 1432-1467. doi: 10.1007/s00332-015-9258-5.

[198] Weiming Xiang, Hoang-Dung Tran, Xiaodong Yang, and Taylor T Johnson. Reachable set estimation for neural network control systems: A simulation-guided approach. *IEEE Transactions on Neural Networks and Learning Systems*, 32(5):1821–1830, 2020.

[199] Chao Yan and Mark R. Greenstreet. Faster projection based methods for circuit level verification. In *2008 Asia and South Pacific Design Automation Conference*, pages 410–415. IEEE, 2008.

[200] Hengjun Zhao, Xia Zeng, Taolue Chen, and Zhiming Liu. Synthesizing barrier certificates using neural networks. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, pages 1–11, April 2020. ISBN 978-1-4503-7018-9.

[201] Hengjun Zhao, Xia Zeng, Taolue Chen, Zhiming Liu, and Jim Woodcock. Learning safe neural network controllers with barrier certificates. *Formal Aspects of Computing*, 33(3):437–455, 2021. ISSN 0934-5043, 1433-299X.

[202] Qingye Zhao, Xin Chen, Yifan Zhang, Meng Sha, Zhengfeng Yang, Wang Lin, Enyi Tang, Qiguang Chen, and Xuandong Li. Synthesizing ReLU neural networks with two hidden layers as barrier certificates for hybrid systems. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pages 1–11. ACM, 2021.