Formal Safety Net Control Using Backward Reachability Analysis

Bastian Schürmann, Moritz Klischat, Niklas Kochdumper, and Matthias Althoff

***Abstract*— Ensuring safety is crucial for the successful deployment of autonomous systems, such as self-driving vehicles, unmanned aerial vehicles, and robots acting close to humans. While there exist many controllers which optimize certain criteria, such as energy consumption, comfort, or low wear, they are usually not able to guarantee safety at all times for constrained nonlinear systems affected by disturbances. Many controllers providing safety guarantees, however, have no optimal performance. The idea of this paper is therefore to synthesize a formally correct controller that serves as a safety net for an unverified, optimal controller. This way, most of the time, the optimal controller is in charge and leads to a desired, optimal control performance. The safety controller constantly monitors the actions of the optimal controller and takes over if the system would become unsafe. The safety controller utilizes a novel concept of backward reachable set computation, where we avoid the need of computing under-approximations of reachable sets. We have further developed a new approach that analytically describes reachable sets, making it possible to efficiently maximize the size of the backward reachable set. We demonstrate our approach by a numerical example from autonomous driving.**

***Index Terms*— Set-based control, reachability analysis, backward reachable sets, safety controller, safety net, nonlinear systems, disturbed systems, constrained systems, and optimization.**

I. Introduction

As modern control applications become more autonomous and are used in closer interaction with humans, ensuring safety becomes more and more difficult. For instance, autonomous vehicles should never cause an accident to achieve broad public acceptance. The same holds true in human-robot collaboration scenarios. Both examples have in common that the system has to avoid unsafe sets at all times – a difficult task, since the controlled systems often are complex, nonlinear, affected by disturbances as well as noisy sensors, and are subject to state and input constraints.

Even though safety is undeniably the most important feature for a successful application of autonomous systems, there are also other important aspects which determine a successful application in practice, e.g., comfort, energy consumption, and long-term wear. There exists a wide variety of classical controllers which focus on these performance goals and are often used in practice, e.g., standard MPC [1]. However, due to the way they are designed, they cannot be formally verified to prove safety in all situations. Formally correct control algorithms, on the other hand, focus on ensuring safety under all possible situations, which often requires large input values and fast input changes opposing comfort, energy consumption, and long-term wear goals. Before we propose a solution to these problems, we first summarize the state of the art for all aspects related to our paper.

a) Safe Controllers: There is a great interest in the design of controllers which provide formal safety guarantees. However, it is hard to find efficient algorithms which solve constrained nonlinear control problems affected by disturbances. Examples from the literature include tube-based MPC, where an auxiliary controller is used to keep the system in a tube around a trajectory which is iteratively optimized over a moving horizon [2]–[7]. An alternative is to use explicit MPC, where the optimization problem is solved offline and stored [8]–[11]. Other approaches which control all states in tubes or funnels are LQR trees [12], [13], controllers which use so-called trajectory robustness [14], and approaches which concatenate invariant sets of different controllers in trees [15]. Abstraction-based controllers [16]–[24], are a large class of formal approaches which discretize the state and input spaces and use methods from automata theory to provide guarantees for the satisfaction of complex specifications.

b) Under-Approximative Reachable Sets: When using approximations of reachable sets to ensure safety, one has to compute over-approximations for forward reachable sets and under-approximations for backward reachable sets [25]. While the idea is simple, the computation of under-approximative backward reachable sets is not straightforward. There exist fewer methods than for forward reachable set computation like [26] and they are often more conservative than the forward algorithms. They exist for linear systems [27]–[29], piecewise affine systems [30], and polynomial systems [31]–[33]. Only a few consider general nonlinear systems, e.g., [34]–[39], of which most consider systems without inputs [34]–[38]. The combination of backward reachable sets and controller design is done in [40], however only for discrete-time, linear systems without disturbances.

Since we consider nonlinear systems, let us review the approaches for these systems in more detail. Several approaches, mostly for polynomial systems, use the Hamilton-Jacobi framework to find under-approximations of reachable sets [31], [33], [34]. While they obtain quite good results for low-dimensional polynomial systems, they do not scale well with higher dimensions or polynomial degrees, which

The authors are with the Department of Informatics, Technische Universität München, Boltzmannstr. 3, 85748 Garching, Germany (email: bastian.schuermann@tum.de, moritz.klischat@tum.de, niklas.kochdumper@tum.de, althoff@tum.de).

restricts their application to around seven dimensions with a low polynomial degree. Recent approaches use decomposition to deal with the unfavorable computational complexity of their methods [41], at least for suitable classes of nonlinear systems, where the dynamics are loosely coupled. Other approaches use Taylor models in combination with polytopic level sets to under-approximate the reachable set [37]. They have the disadvantage that they might result in unconnected sets, where only one is the actual under-approximation, which then requires computationally expensive checks to find the proper set. Computational challenges also arise when trying to solve practical problems with the approach in [35], which uses so-called polynomial level-set functions to represent the under-approximative reachable sets and is too hard to compute for larger state spaces. Another method is proposed in [38], which has been extended for systems with inputs in [39]. Their results are promising, however, they only compute under-approximations of projections on the coordinate axes. To the best of our knowledge, there exists no approach which computes actual under-approximative reachable sets for nonlinear systems with inputs. Therefore, and since we need a special preservation of input effects for our controller synthesis, which is not provided by any of the existing techniques, we develop a new approach in this work.

c) Safety Nets: The idea of a controller safeguarding other controllers has been previously proposed, e.g., by [42]. There, stability is proven for certain regions with a simple controller, which takes over if a more complex controller fails to satisfy certain safety constraints. This framework is extended in [43], [44] by including the online computation of reachable sets to identify the regions from which the verified controller can recover the system to a stable region. Computing a safe invariant region for a safety controller and ensuring that a second controller cannot leave this region is done in invariance control [45], [46]. Combining the satisfaction of safety constraints with performance controllers can also be achieved using control barrier functions and has been applied to different types of systems, such as autonomous cars and robotic applications [47], [48]. A related method is that of reference governors [49]–[51], which monitor the system and modify the inputs when constraints would be violated. The idea of using reachability analysis together with a safety controller is also used for motion planning in [52] and for safe reinforcement learning in [53]. Another type of safety net controller named shielding is presented in [54], which can also be used for safe reinforcement learning [55].

d) Previous Work: In earlier works, we obtained safe controllers by combining optimal control with reachability analysis. In [56], [57], we obtained a piecewise-constant control law by interpolating optimal open-loop trajectories of extreme states and generators, respectively. In [58], we optimized continuous feedback controllers, but only for disturbed linear systems. In [59], we combine the ideas of [57] and [58] to benefit from the advantages of both approaches.

e) Proposed Approach: To benefit from the advantages of optimal and safe control, we pursue the following idea in this paper: We combine a formal controller serving as a safety net with an optimal, unverified controller which minimizes some cost function. While worst-case behavior has to be considered for safety, in most situations, such behavior does not occur and an optimal controller can achieve better performance. To ensure safety, the formal safety controller monitors the optimal controller by computing its reachable set for the near future. As long as this set is safe, the optimal controller is applied. Otherwise, the safety controller takes over to keep the system safe. Thus, we combine the optimality of classical control with the safety of formal controller synthesis. To compute the formal safety net, we use a set-based control method similar to [57] which combines controller synthesis with reachability analysis.

In our earlier works [56]–[59], we aim at minimizing the reachable set for a given initial set as fast as possible. When using these approaches as safety net controllers, the sets in which the optimal controller has to stay would be small so that the safety controller would take over most of the time. In this work, we address this problem by computing backward reachable sets. By fixing the final reachable set, we compute backwards in time the maximum set for which we can find a safety controller, thereby maximizing the set in which the optimal controller can safely operate and thus minimizing interventions from the safety controller.

f) Contributions: The paper has five main contributions:

(i) We present a novel way of computing under-approximative reachable sets for nonlinear systems with disturbances. While there exists no approach in the literature which is capable of doing this, it is required for synthesizing set-based controllers with safety guarantees. In addition, our novel reachability algorithm allows us to obtain analytical dependencies between the reachable set and its initial set and inputs acting on the system. These dependencies make it possible to substantially speed up computations since one obtains a simple cost function for the optimization.

(ii) We use the results from (i) to develop a new control algorithm based on backward reachable sets. This algorithm has the same advantages as our previous set-based control algorithms [56]–[58], as it provides a control law by optimizing over all possible trajectories starting from an initial set under all possible disturbance realizations. While providing formal guarantees for the constraint satisfaction of disturbed nonlinear systems, our approach has the advantage that it does not require discretizing the state space. Similarly, we do not need to find invariant sets or Lyapunov functions, which together with the required computations, such as sums-of-squares programming suffer from bad scalability for higher dimensional systems. At the same time, our approach results in an optimized, time-dependent feedback law, which has a better performance than a fixed feedback controller, as often used in most existing approaches including tube-based MPC. All complex computations can be performed offline and the solutions can be stored in a maneuver automaton (see Sec. III), thereby allowing a fast online application.

(iii) In addition to these advantages, our new approach computes the largest initial set for which we can control all trajectories to the desired final set despite the presence of disturbances. This makes it very suitable for the application as a safety net controller.

(iv) We present a novel way of set-based safety net control which uses reachable set computation. Our reachable set algorithm scales polynomially and therefore better than existing approaches. As a result, this algorithm can even be used for checking the safety of the optimal controllers online.
(v) In contrast to most other methods, we explicitly consider the computation time of our approach during the online verification. We are even able to guarantee safety without requiring a worst-case execution time, since we can safely switch to the safety controller if the execution time exceeds the allocated time.

g) Outline: The remainder of this paper is organized as follows. We begin with a formal problem statement in Sec. II and some background in Sec. III. In Sec. IV, we present the online application of our combined approach. Afterwards, we explain how backward reachable sets are used to obtain the safety controller: first for linear systems in Sec. V and then for disturbed, nonlinear systems in Sec. VI. We illustrate the applicability of our approach with numerical examples in Sec. VII, before we conclude with a discussion in Sec. VIII and a summary in Sec. IX.

II. Problem Formulation

We consider a disturbed, nonlinear, time-continuous system of the form

$$\dot{x}(t) = f(x(t), u(t), w(t)), \tag{1}$$

with states $x(t) \in \mathbb{R}^n$, inputs $u(t) \in \mathbb{R}^m$, and disturbances $w(t) \in \mathcal{W} \subset \mathbb{R}^d$ ($\mathcal{W}$ is compact, i.e., closed and bounded). The disturbance does not need to be additive and we do not require any stochastic properties for $w(\cdot)$; we only assume that any possible disturbance trajectory is bounded at any point in time in the compact set $\mathcal{W}$. We denote this by $w(\cdot) \in \mathcal{W}$, which is a shorthand for $w(t) \in \mathcal{W}, \forall t \in [0, t_f]$, where $t_f \in \mathbb{R}_0^+$ is the final time. The same shorthand is also used for state and input trajectories throughout the paper. We denote the solution of (1) with initial state x_0, input $u(\cdot)$, and disturbance $w(\cdot)$ at time t by $\xi(x_0, u(\cdot), w(\cdot), t)$. The solution satisfies the following two properties:

$$\begin{aligned}&\xi(x_0, u(\cdot), w(\cdot), 0) = x_0,\\ &\dot{\xi}(x_0, u(\cdot), w(\cdot), t) = f\Big(\xi\big(x_0, u(\cdot), w(\cdot), t\big), u(t), w(t)\Big),\\ &\forall t \in \mathbb{R}_0^+.\end{aligned}$$

If we consider an undisturbed system, we use $\xi(x_0, u(\cdot), 0, t)$ to denote the solution without disturbances, i.e., $\mathcal{W} = \{0\}$.

The task is to find an initial set $\mathcal{X}_0 \subset \mathbb{R}^n$ with maximum volume around a desired state x_0 and a corresponding verified control law $u_{ver}(x, t)$ such that for system (1), all solutions starting at $t = 0$ in $\mathcal{X}_0$ end in a given final set $\mathcal{X}_f \subset \mathbb{R}^n$ at $t = t_f$ despite the disturbance set $\mathcal{W}$. Furthermore, the controller must ensure that all solutions satisfy convex constraints on the states and inputs, i.e.,

$$\xi(x_0, u(\cdot), w(\cdot), t) \in \mathcal{S}, \quad \forall t \in [0, t_f], \tag{2}$$
$$u(t) \in \mathcal{U}, \quad \forall t \in [0, t_f], \tag{3}$$

where $\mathcal{S}$ and $\mathcal{U}$ are both convex sets with half-space representations

$$\mathcal{S} = \{x \in \mathbb{R}^n | C_\mathcal{S} x \leq d_\mathcal{S}\}, \tag{4}$$
$$\mathcal{U} = \{u \in \mathbb{R}^m | C_\mathcal{U} x \leq d_\mathcal{U}\}, \tag{5}$$

where $C_\mathcal{S} \in \mathbb{R}^{q_\mathcal{S} \times n}, d_\mathcal{S} \in \mathbb{R}^{q_\mathcal{S}}, C_\mathcal{U} \in \mathbb{R}^{q_\mathcal{U} \times m}$, and $d_\mathcal{U} \in \mathbb{R}^{q_\mathcal{U}}$.

As explained in the following section, we solve this problem offline and store its solution in a maneuver automaton. Since the locations of most non-convex constraints, such as other traffic participants in automated driving, are usually not known during offline computation, we only consider convex input constraints, e.g., maximum acceleration or steering, and convex state constraints, e.g., maximum velocity. The non-convex dynamic constraints are handled during the online planning using the maneuver automaton, which is a standard approach and can be done using existing techniques (see e.g., [60]).

III. Background

Before discussing our approach, let us first provide some background. We begin by defining reachable sets:

Definition 1 (Reachable Set): For system (1), the reachable set $\mathcal{R}_{t,\mathcal{U},\mathcal{W}}(\mathcal{S}) \subset \mathbb{R}^n$ for a time t, inputs $u(\cdot) \in \mathcal{U} \subset \mathbb{R}^m$, disturbances $w(\cdot) \in \mathcal{W} \subset \mathbb{R}^d$, and a set of initial states $\mathcal{S} \subset \mathbb{R}^n$ is the set of end states of trajectories starting in $\mathcal{S}$ after time t, i.e.,

$$\begin{aligned}\mathcal{R}_{t,\mathcal{U},\mathcal{W}}(\mathcal{S}) = \{x(t) \in \mathbb{R}^n | \exists x_0 \in \mathcal{S}, u(\cdot) \in \mathcal{U}, w(\cdot) \in \mathcal{W} :\\ \xi(x_0, u(\cdot), w(\cdot), t) = x(t)\}.\end{aligned}$$

The reachable set over a time interval $[t_1, t_2]$ is the union of all reachable sets for these time points, i.e.,

$$\mathcal{R}_{[t_1,t_2],\mathcal{U},\mathcal{W}}(\mathcal{S}) = \bigcup_{t \in [t_1,t_2]} \mathcal{R}_{t,\mathcal{U},\mathcal{W}}(\mathcal{S}).$$

If we consider the reachable set for a system with feedback $u_{fb}(x)$, then we denote by $\mathcal{R}_{t,u_{fb},\mathcal{W}}(\mathcal{S})$ the reachable set obtained if we consider the closed-loop dynamics $\dot{x}(t) = f(x(t), u_{fb}(x(t)), w(t))$. Since it is not possible to compute exact reachable sets for most systems [61], we compute approximations instead.

The efficiency of the computation of reachable sets depends on the chosen set representation. A type of set which has very favorable properties for nonlinear systems are polynomial zonotopes [62], which are closed under quadratic and higher-order maps. In this work, we use a sparse representation of polynomial zonotopes, which are defined as follows (see [63, Def. 1]):

Definition 2 (Polynomial Zonotope): A set is called a polynomial zonotope if it can be written as

$$\begin{aligned}\mathcal{PZ} = \Bigg\{ x \in \mathbb{R}^n \Big| x = c + \sum_{j=1}^{v_I} \mu_j g_I^{(j)}\\ + \sum_{i=1}^{v} \left(\prod_{k=1}^{y} \lambda_k^{E_{k,i}} \right) g^{(i)},\ \lambda_k, \mu_j \in [-1, 1] \Bigg\}.\end{aligned} \tag{6}$$

Here, $c \in \mathbb{R}^n$ is the starting point, $g_I^{(j)} \in \mathbb{R}^n, j \in \{1, \ldots, v_I\}$, are the independent generators, $g^{(i)} \in \mathbb{R}^n, i \in \{1, \ldots, v\}$, are

the dependent generators, and $E \in \mathbb{N}_0^{y \times v}$ is the matrix that stores the polynomial exponents.

A special case of polynomial zonotopes are zonotopes, which have only independent generators:

Definition 3 (Zonotope): A set is called a zonotope if it can be written as

$$\mathcal{Z} = \left\{ x \in \mathbb{R}^n \middle| x = c + \sum_{i=1}^{r} \alpha_i g^{(i)}, \alpha_i \in [-1, 1] \right\}.$$

Here, $c \in \mathbb{R}^n$ defines the center of the zonotope, and $g^{(i)} \in \mathbb{R}^n, i \in \{1, \dots, r\}$, are $r = o\,n$ generators, with o denoting the order of the zonotope. We use $\langle c, g^{(1)}, \dots, g^{(r)} \rangle$ as a more concise notation of $\mathcal{Z}$.

Zonotopes offer a simpler set representation compared to polynomial zonotopes and especially for linear systems, polynomial zonotopes do not have any advantages over zonotopes. Therefore, we use zonotopes for computations with linear or linearized systems and polynomial zonotopes when dealing with nonlinear dynamics.

Since reachability analysis is rather time consuming, the computation of the safety controller can often not be done in real time. A solution to this problem is to construct safe maneuver automata [64] (see also [65]–[67]) as illustrated in Fig. 1: One computes safe controllers for short trajectory pieces (a.k.a. motion primitives), e.g., *drive straight* or *turn left* (see Fig. 1(a)). These motion primitives are stored as states in a maneuver automaton. There is a transition from one state to another, if the final set of the first motion primitive ends in the initial set of the second motion primitive as illustrated in Fig. 1(b). These pre-computed motion primitives can be combined online to safely control the system, as shown in Fig. 1(c)–(f).

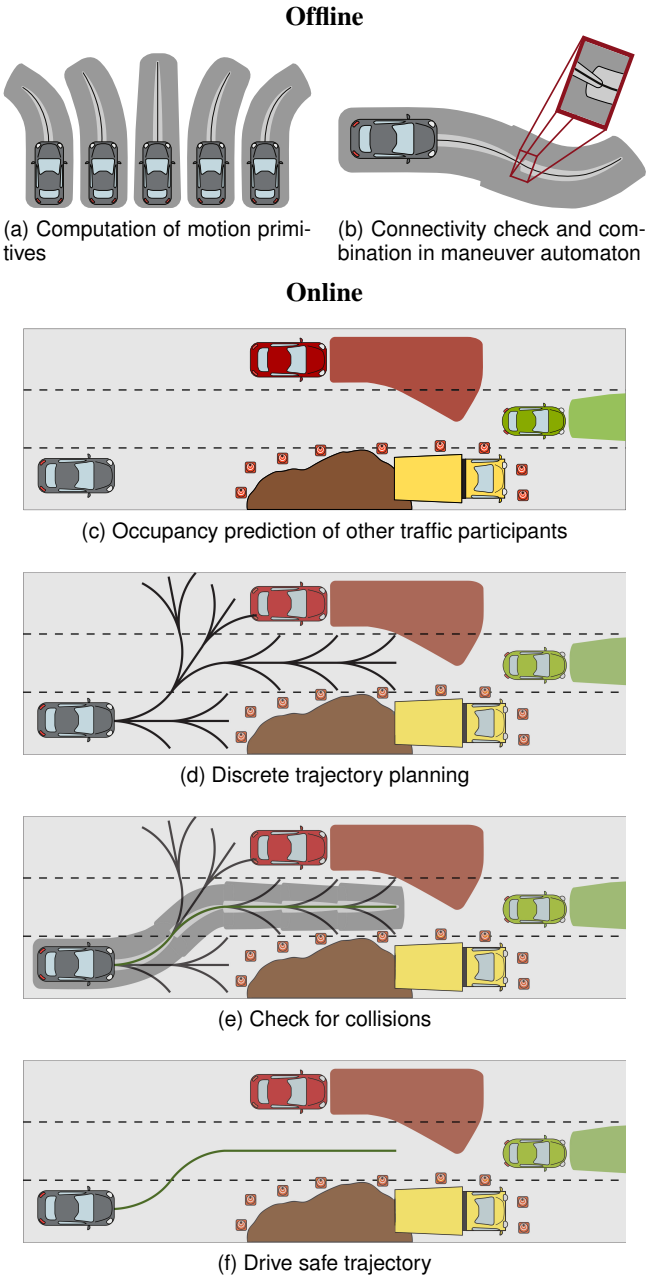
Definition 4 (Maneuver Automaton): A maneuver automaton $\mathbf{MA} = \{\mathcal{M}, \mathcal{D}\}$ is a tuple of a set of motion primitives $\mathcal{M}$ and a set of discrete transitions $\mathcal{D} \subset \mathcal{M} \times \mathcal{M}$ defining which motion primitives can be followed by each other.

Definition 5 (Motion Primitive): A motion primitive

$$\mathbf{MP} = \{x_{ref}(\cdot), u_{ref}(\cdot), t_f, \mathcal{X}_0, \mathcal{X}_f, u_{ver}(\cdot), \mathcal{R}_{[0,t_f],u_{ver},\mathcal{W}}(\mathcal{X}_0)\}$$

is again a tuple, containing a reference trajectory $x_{ref}(\cdot)$, the corresponding reference input $u_{ref}(\cdot)$, a duration t_f, initial and final sets $\mathcal{X}_0$ and $\mathcal{X}_f$, respectively, a verified safety controller $u_{ver}(\cdot)$, and the reachable set of the verified safety controller $\mathcal{R}_{[0,t_f],u_{ver},\mathcal{W}}(\mathcal{X}_0)$.

There exists a transition from one motion primitive $\mathbf{MP}_i$ with final set $\mathcal{X}_f^{(i)}$ to another motion primitive $\mathbf{MP}_j$ with initial set $\mathcal{X}_0^{(j)}$, i.e., $(\mathbf{MP}_i, \mathbf{MP}_j) \in \mathcal{D}$ if and only if $\mathcal{X}_f^{(i)} \subseteq T(\mathcal{X}_0^{(j)})$, where $T(\mathcal{X}_0^{(j)})$ is an allowed transformation depending on the system dynamics. For example, if the system dynamics is independent of the initial state, it is admissible to translate the initial state of the second motion primitive into the final set of the initial motion primitive. In this case, it is only important that the final set is completely contained in the shifted initial set. We give an example for such a transformation in the numerical example in Sec. VII.



(a) Computation of motion primitives

(b) Connectivity check and combination in maneuver automaton

(c) Occupancy prediction of other traffic participants

(d) Discrete trajectory planning

(e) Check for collisions

(f) Drive safe trajectory

Fig. 1. Overview of robust maneuver automata design for an example in automated driving.

IV. Online Control with Safety Net

For the online control, we assume that we have found a safe sequence of motion primitives, which connect the initial state with the goal state. We discuss how to compute such safe motion primitives used in a maneuver automaton offline in Sec. V and Sec. VI. There exist several techniques to find such a safe sequence of motion primitives given a maneuver automaton, e.g.,

- discrete search techniques using sampling, e.g., search-tree based methods [60];
- algorithms which try to match reference trajectories [68];
- moving-horizon planning techniques which can be combined with fail-safe maneuvers [69].

Let us start by describing how our online controller works,

which is summarized in Alg. 1 and illustrated in Fig. 2. We explain this for a single motion primitive to simplify the notation. At the end of each motion primitive, we simply switch to the following one. During the online application, we repeatedly need to decide when to use the unverified, optimal controller or the verified, safety controller. To this end, we use a safety net controller which acts in discrete-time and allows switching to and from the optimal controller at discrete switching times $t_k := k\Delta t,\ k \in \mathbb{N}_0$, where $\Delta t := \frac{t_f}{M},\ M \in \mathbb{N}$. Similarly, we only start a new optimization for the optimal controller once in each time step, so that the control behavior remains predictable. Besides this restriction, we are flexible with the form of the unverified controller and also allow continuous-time controllers.

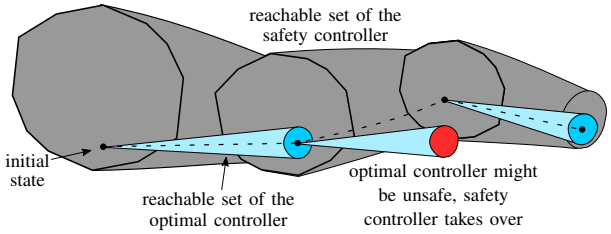


Fig. 2. Overview of the online control: Based on the measurement of the current state, we compute the reachable set for the optimal controller. If it ends inside the reachable set of the safety controller and if it satisfies all constraints, we apply the optimal controller (see first time step). If this is not the case, we switch to the safety controller (see second time step). After the intervention of the safety controller, we check if the optimal controller is safe once again, and switch back to it if safety can be ensured (see third step).

Algorithm 1 Online Control Algorithm

Input: $f(x,u,w)$, $\mathcal{W}$, Δt, t_c, **MP**, $u_{unv}(\cdot)$, $x(\cdot)$
Output: control inputs $u_{appl}(\cdot)$

1: **while** $t < t_f$ **do**
2: $x(t_k - t_c) \leftarrow$ measured state at $t = t_k - t_c$
3: $\hat{\mathcal{X}}(t_k|t_k - t_c) \leftarrow \mathcal{R}_{t_c,u_{appl},\mathcal{W}}(x(t_k - t_c))$
4: $\hat{\mathcal{X}}(t_{k+1}|t_k - t_c) \leftarrow \mathcal{R}_{\Delta t,u_{unv},\mathcal{W}}(\hat{\mathcal{X}}(t_k|t_k - t_c))$
5: **if** computation finished before t_k and satisfaction of (7)-(9) **then**
6: $u_{appl}(\cdot) \leftarrow u_{unv}(\cdot),\ \forall t \in [t_k, t_{k+1})$
7: **else**
8: $u_{appl}(\cdot) \leftarrow u_{ver}(\cdot),\ \forall t \in [t_k, t_{k+1})$
9: **end if**
10: **end while**

For each switching time t_k, we have to decide which controller we want to apply in the next time interval. Let us assume that we are applying a feasible controller $u_{appl}(\cdot)$ for the previous time interval $[t_{k-1}, t_k]$: either the unverified, optimal controller $u_{unv}(\cdot)$ or the verified, safety controller $u_{ver}(\cdot)$. Since the computation of the reachable set takes some time, we need to measure the state $x(t_k - t_c)$ and start the computations at time $t_k - t_c$, to know at t_k which controller to apply during time period $[t_k, t_{k+1}]$. Here, $t_c \leq \Delta t$ denotes an allocated computation time consisting of (i) the computation time of the optimal controller, (ii) the computation time of the verifier, and (iii) a buffer time. It can be chosen by testing how long these computations take on the used machine. If the actual computation time takes longer, we simply apply the safety controller for the next time step to ensure safety.

We assume here that we can measure the current state, possibly subject to some measurement errors. If not all states are measurable, we can also use a set-based observer [70], [71]. For a simpler notation, we present our approach without measurement errors. In the case of measurement errors or set-based observers, we would obtain an initial set instead of a single initial state. Since we use set-based operators, this would not change much.

Based on the measurement of $x(t_k - t_c)$ (see Alg. 1, line 2), we first compute the reachable set at time t_k for the currently applied controller $u_{appl}(\cdot)$ (Alg. 1, line 3):

$$\hat{\mathcal{X}}(t_k|t_k - t_c) := \mathcal{R}_{t_c,u_{appl},\mathcal{W}}(x(t_k - t_c)),$$

where we use the notation $\hat{\mathcal{X}}(t_2|t_1)$ to refer to the reachable set at time t_2 which is computed based on the information at time t_1. Starting from $\hat{\mathcal{X}}(t_k|t_k - t_c)$, we then compute the reachable set at t_{k+1} (Alg. 1, line 4) using the unverified, optimal controller $u_{unv}(\cdot)$:

$$\hat{\mathcal{X}}(t_{k+1}|t_k - t_c) := \mathcal{R}_{\Delta t,u_{unv},\mathcal{W}}(\hat{\mathcal{X}}(t_k|t_k - t_c)).$$

Next, we check if the controller would satisfy the state constraints

$$\mathcal{R}_{[0,\Delta t],u_{unv},\mathcal{W}}(\hat{\mathcal{X}}(t_k|t_k - t_c)) \subseteq \mathcal{S} \tag{7}$$

and the input constraints

$$u_{unv}(x, t_k + \tau) \in \mathcal{U},\ \forall x \in \mathcal{R}_{\tau,u_{unv},\mathcal{W}}(\hat{\mathcal{X}}(t_k|t_k - t_c)), \quad \forall \tau \in [0, \Delta t), \tag{8}$$

during the whole time interval $[t_k, t_{k+1}]$ and if it would end completely inside the reachable set of the safety controller at time t_{k+1} (Alg. 1, line 5):

$$\hat{\mathcal{X}}(t_{k+1}|t_k - t_c) \subseteq \mathcal{R}_{t_{k+1},u_{ver},\mathcal{W}}(\mathcal{X}_0). \tag{9}$$

This means that the solution of the optimal controller can leave the reachable set of the safety controller as long as it satisfies the state and input constraints and returns to the reachable set of the safety controller at t_{k+1}. This ensures that we can switch at t_{k+1} to the safety controller, if no safe solution of the optimal controller exists for $[t_{k+1}, t_{k+2}]$.

If all three constraints are satisfied and the computations are finished before time t_k, we can safely apply the optimal controller during the time interval $[t_k, t_{k+1})$ (Alg. 1, line 6); otherwise, we switch to the safety controller (Alg. 1, line 8).

Theorem 1: Given a motion primitive **MP** from Alg. 2 connecting the initial state x_0 with the desired final set $\mathcal{X}_f \subseteq \mathcal{S}$ and satisfying the state constraints (2), Alg. 1 returns the input trajectory $u_{appl}(\cdot)$. This input trajectory satisfies the input constraints (3) and results in a state trajectory $\xi(x_0, u_{appl}(\cdot), w(\cdot), \cdot)$, which satisfies the state constraints (2) and reaches the final set, i.e., $\xi(x_0, u_{appl}(\cdot), w(\cdot), t_f) \in \mathcal{X}_f$.

Proof: The proof follows by construction of the algorithm and is shown by induction:
Base Case: At time t_0, we know a safe solution from the safety controller for the time interval $[t_0, t_1]$ which satisfies

the state constraints by assumption and the input constraints by construction (as described later in Sec. VI).

Induction Hypothesis: If we know a safe controller for the time interval $[t_{k-1}, t_k]$, we can always get a safe solution for the following time interval $[t_k, t_{k+1}]$.

Induction Step: When we measure the state $x(t_k - t_c)$, we know that we are inside the reachable set of the currently applied controller. By the definition of reachable sets, applying the same controller ensures that the reachable set $\mathcal{R}_{t_c,u_{appl},\mathcal{W}}(x(t_k - t_c))$ ends inside the previously computed reachable set, i.e.,

$$\begin{aligned}\mathcal{R}_{t_c,u_{appl},\mathcal{W}}(x(t_k - t_c)) &\subseteq \mathcal{R}_{\Delta t,u_{appl},\mathcal{W}}(\hat{\mathcal{X}}(t_{k-1}|t_{k-1} - t_c)) \\ &\overset{(9)}{\subseteq} \mathcal{R}_{t_k,u_{ver},\mathcal{W}}(\mathcal{X}_0).\end{aligned}$$

Since $\mathcal{R}_{t_k,u_{ver},\mathcal{W}}(\mathcal{X}_0)$ is the reachable set of the safety controller at time t_k, we can always safely switch at t_k to the safety controller. Therefore, we always check the constraints with the optimal controller for the time interval $[t_k, t_{k+1}]$ at time $t_k - t_c$: If this is safe, we apply the optimal controller for the next time interval; otherwise, we switch to the safety controller at time t_k to obtain a safe solution for the next time interval $[t_k, t_{k+1}]$. ■

In the next two sections, we discuss how to construct the safety controller by using backward reachability analysis.

V. Backward Controller Synthesis for Linear, Discrete-Time Systems

Before we consider nonlinear systems for synthesizing safe controllers, we discuss the case of discrete-time, linear, time-varying systems without disturbances of the form

$$x(t_{k+1}) = A_k x(t_k) + B_k u(t_k), \tag{10}$$

with $A_k \in \mathbb{R}^n$ and $B_k \in \mathbb{R}^m$ denoting the time-varying state and input matrices at time t_k. We assume that the matrices A_k have full rank, i.e., they are invertible. If we obtain them as time-discretized versions of continuous time systems, this assumption is satisfied, as the matrix exponential $e^{A\Delta t}$ is always invertible [72, Ch. 7.2, Thm. 2]. In this case, the backward reachable set can be computed exactly. Afterwards, we extend the idea presented in this section to continuous-time, nonlinear systems with disturbances in Sec. VI. In both sections, we present the algorithms to compute a single motion primitive. For obtaining a full maneuver automaton, these algorithms are simply applied for different initial sets and desired final states.

A. Backward Reachable Set

To obtain the backward reachable set of (10), we express the evolution of a single state $x(t_k)$ after H steps as

$$x(t_{k+H}) = \bar{A}x(t_k) + \sum_{i=k}^{k+H-1} \bar{B}_i u(t_i), \tag{11}$$

where we use the shorthand notation:

$$\begin{aligned}\bar{A} &= A_{k+H-1} \cdot \cdots \cdot A_0, \\ \bar{B}_i &= A_{k+H-1} \cdot \cdots \cdot A_{i+1} B_i, \quad \forall i \in \{k, \cdots, k+H-2\},\end{aligned}$$

and $\bar{B}_{k+H-1} = B_{k+H-1}$. Solving (11) for $x(t_k)$ leads to

$$x(t_k) = \bar{A}^{-1}\left(x(t_{k+H}) + \sum_{i=k}^{k+H-1} (-1)\bar{B}_i u(t_i)\right). \tag{12}$$

After replacing concrete values with sets, we obtain the backward reachable set as:

$$\mathcal{X}(t_k) = \bar{A}^{-1}\left(\mathcal{X}(t_{k+H}) \oplus \bigoplus_{i=k}^{k+H-1} (-1)\bar{B}_i \mathcal{U}\right), \tag{13}$$

where $\oplus$ denotes the Minkowski sum defined as $\mathcal{A} \oplus \mathcal{B} = \{a + b | a \in \mathcal{A}, b \in \mathcal{B}\}$ and $\bigoplus$ the corresponding sum symbol.

This computation can be done easily with zonotopes, as they are closed under linear maps and Minkowski sum. For zonotopic sets $\mathcal{X}(t_{k+H}) = \langle c_{k+H}, g_{k+H}^{(1)}, \ldots, g_{k+H}^{(r_{k+H})} \rangle$ and $\mathcal{U} = \langle c_u, g_u^{(1)}, \ldots, g_u^{(r_u)} \rangle$, the backward reachable set is

$$\begin{aligned}\mathcal{X}(t_k) &= \bar{A}^{-1}\left(\mathcal{X}(t_{k+H}) \oplus \bigoplus_{i=k}^{k+H-1} (-1)\bar{B}_i \mathcal{U}\right) \\ &= \Bigg\{x \in \mathbb{R}^n \Bigg| x = \bar{A}^{-1}c_{k+H} - \sum_{i=k}^{k+H-1} \bar{A}^{-1}\bar{B}_i c_u \\ &\quad + \sum_{i=1}^{r_{k+H}} \alpha_i \bar{A}^{-1} g_{k+H}^{(i)} - \sum_{i=k}^{k+H-1} \sum_{j=1}^{r_u} \beta_{i,j} \bar{A}^{-1}\bar{B}_i g_u^{(j)}, \\ &\quad \alpha_i, \beta_{i,j} \in [-1, 1]\Bigg\} \\ &=: \left\{x \in \mathbb{R}^n \middle| x = \hat{c} + \sum_{i=1}^{\hat{r}} \hat{\alpha}_i \hat{g}^{(i)}, \hat{\alpha}_i \in [-1, 1]\right\},\end{aligned} \tag{14}$$

where

$$\begin{aligned}\hat{c} &:= \bar{A}^{-1}c_{k+H} - \sum_{i=k}^{k+H-1} \bar{A}^{-1}\bar{B}_i c_u, \\ \hat{g}^{(i)} &:= \bar{A}^{-1} g_{k+H}^{(i)}, \ \forall i \in \{1, \ldots, r_{k+H}\}, \\ \hat{g}^{(r_{k+H}+i r_u+j)} &:= -\bar{A}^{-1}\bar{B}_{k+i} g_u^{(j)}, \ \forall i \in \{0, \ldots, H-1\}, \\ &\qquad \forall j \in \{1, \ldots, r_u\}.\end{aligned}$$

Here, r_{k+H} denotes the number of generators of $\mathcal{X}(t_{k+H})$ and $\hat{r} := r_{k+H} + H r_u$. By introducing $\hat{c}, \hat{g}^{(1)}, \ldots, \hat{g}^{(\hat{r})}$, we obtain a simpler notation and are able to handle the generators resulting from the states and inputs in the same way.

B. Resulting Online Control Law

Let us now discuss how to obtain the control law steering the system from a state $x \in \mathcal{X}(t_k)$ to the desired set $\mathcal{X}(t_{k+H})$. Each state $x \in \mathcal{X}(t_k)$ can be expressed as a superposition of the generators $\hat{g}^{(i)}$ by finding the corresponding parameters $\hat{\alpha}_i$ (see (14)). Since, in general, the choice of $\hat{\alpha}_i$ is not unique, we solve the following optimization problem:

$$\min_{\hat{\alpha}} \sum_{i=1}^{\hat{r}} \rho_i |\hat{\alpha}_i| \tag{15}$$

$$\text{s.t. } x = \hat{c} + \sum_{i=1}^{\hat{r}} \hat{\alpha}_i \hat{g}^{(i)}, \tag{16}$$

$$-1 \leq \hat{\alpha}_i \leq 1, \ \forall i \in \{1, \ldots, \hat{r}\}, \tag{17}$$

with weights $\rho_i \in \mathbb{R}_0^+$. We then multiply $G_u = [g_u^{(1)}, \dots, g_u^{(r_u)}]$ with the weights $\hat{\alpha}_i$ corresponding to the input generators at time step $k+j$ to obtain the following piecewise-constant control law $\forall j \in \{0, \dots, H-1\}$:

$$u(x, t_{k+j}) = c_u + G_u[\hat{\alpha}_{r_{k+H}+j\, r_u+1}, \dots, \hat{\alpha}_{r_{k+H}+(j+1)\, r_u}]^T,$$

It follows from the superposition principle of linear systems that if we use the weighted combination of the corresponding inputs, we ensure that any state $x \in \mathcal{X}(t_k)$ is steered in the desired set, too (see, e.g., [56], [57] for details). Since some generators $\hat{g}^{(i)}$ in (14) result from states converging to the desired set without inputs and others from generators which are controlled to the origin by the inputs, the choice of ρ_i balances the applied inputs to the size of the final set: Higher weights for $\rho_1, \dots, \rho_{r_{k+H}}$ in (15) lead to solutions which use higher inputs and end closer to the center of $\mathcal{X}(t_{k+H})$. Higher weights for $\rho_{r_{k+H}+1}, \dots, \rho_{\hat{r}}$, on the other hand, punish large inputs and will lead to solutions which save input capacities on the price of ending closer to the boundaries of $\mathcal{X}(t_{k+H})$.

Since the optimization problem in (15) is a linear program, it can be solved online. For instance, solving the linear problem for the numerical example in Sec. VII only takes around $3\,ms$ for a MATLAB implementation, and even less when using C++. If computation time or effort is a bigger concern, one can also simply apply the first solution which satisfies constraints (16)–(17). If even faster solutions are required, one can express the zonotopes as a convex combination of their extreme states for which closed-form expressions exist [73].

VI. Backward Controller Synthesis for Nonlinear, Continuous-Time Systems

Let us now extend the presented ideas to nonlinear, continuous time systems, for which the superposition principle no longer holds and where the backward reachable sets cannot be computed exactly or efficiently in an under-approximative way.

A. Overview

To obtain an estimate of the backward reachable set, we linearize the dynamics without disturbances for a short time horizon, so that we can use the techniques from Sec. V. In the next step, we compute the forward reachable set considering the actual disturbed, nonlinear dynamics of the controlled system and scale it until it ends in the original final set $\mathcal{X}_f$. We iterate these steps until the initial state x_0 is closest to the center of the reachable set, as presented in Alg. 2 and Fig. 3. Note that the same approach could also be used without a reference trajectory for computing regions of attraction or control invariant sets.

B. Reference Trajectory

We start by computing a reference trajectory from the center of the final set $\mathrm{center}(\mathcal{X}_f)$ to the desired initial state x_0 (see Fig. 3(a)), where $\mathrm{center}(\cdot)$ returns the volumetric center of a set. This is done in function compute_reference_trajectory in

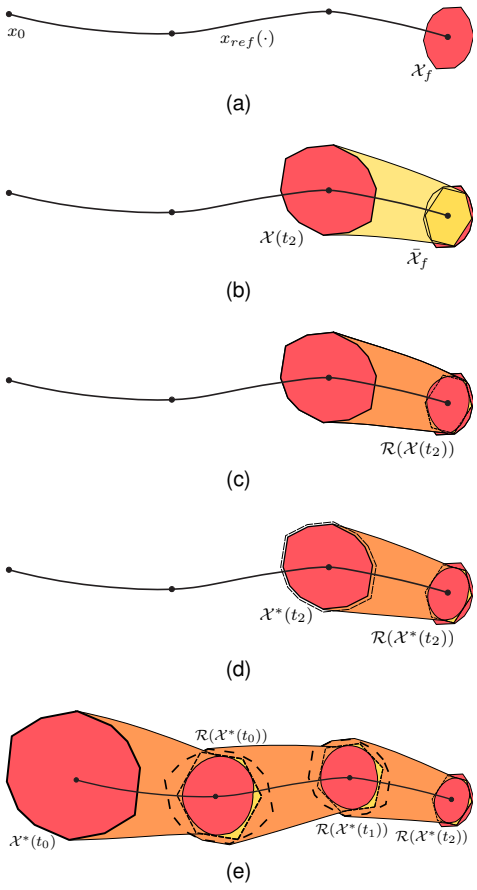


Fig. 3. Illustration of the backward controller synthesis for nonlinear systems: (a) We compute a reference trajectory $\boldsymbol{x_{ref}(\cdot)}$ from the center of the final set $\boldsymbol{\mathcal{X}_f}$ to the desired initial state $\boldsymbol{x_0}$ and divide it into $\boldsymbol{M}$ parts (here, $\boldsymbol{M = 3}$). (b) For computational reasons, we can under-approximate the final set by a zonotope $\boldsymbol{\bar{\mathcal{X}}_f}$ (yellow) with fewer generators. We then compute the backward reachable set $\boldsymbol{\mathcal{X}(t_2)}$ for the linearized dynamics for a short time horizon. (c) Starting from this set, we compute the forward reachable set $\boldsymbol{\mathcal{R}(\mathcal{X}(t_2)) := \mathcal{R}_{\Delta t, u_{ver}, \mathcal{W}}(\mathcal{X}(t_2))}$ based on the disturbed, nonlinear dynamics. (d) If the forward reachable set $\boldsymbol{\mathcal{R}(\mathcal{X}(t_2))}$ does not end inside the under-approximated final set $\boldsymbol{\bar{\mathcal{X}}_f}$ (see (c)), we scale the backward reachable set $\boldsymbol{\mathcal{X}^*(t_2)}$ until $\boldsymbol{\mathcal{R}(\mathcal{X}^*(t_2))}$ ends inside the desired set. (e) We iteratively repeat these steps until we obtain the desired initial set $\boldsymbol{\mathcal{X}^*(t_0)}$.

line 1 of Alg. 2, where we solve the following optimization problem:

$$\begin{aligned} \min_{u_{ref}(\cdot)} \; & \|\xi(x_0, u_{ref}(\cdot), 0, t_f) - \mathrm{center}(\mathcal{X}_f)\| \\ & + \gamma \sum_{k=0}^{N-1} \|u_{ref}(\tau_k)\| \qquad (18) \\ \text{s.t. } & \xi(x_0, u_{ref}(\cdot), 0, t) \in \bar{\mathcal{S}}, \quad \forall t \in [0, t_f], \\ & u_{ref}(\tau_k) \in \bar{\mathcal{U}}, \quad \forall k \in \{0, \cdots, N-1\}, \end{aligned}$$

where $N = M\,H,\ H \in \mathbb{N},\ \tau_k := k\Delta\tau,\ k \in \mathbb{N}_0$, with $\Delta\tau := \frac{t_f}{N} = \frac{\Delta t}{H}$, and where $\gamma \in \mathbb{R}_0^+$ is a factor used to weight the input costs in relation to the state costs. We restrict the inputs to be piecewise constant, so that problem (18) can be solved efficiently using multiple-shooting algorithms [74]. The norm $\|\cdot\|$ can be chosen based on the application. To leave capacities for the set-based controller, we use tightened state and input

Algorithm 2 Backward Reachable Set Algorithm

Input: $f(x,u,w)$, $\mathcal{X}_f$, x_0, $\Delta\tau$, t_f,N, M, $\bar{r}$, $\mathcal{S}$,$\mathcal{U}$,$\bar{\mathcal{S}}$,$\bar{\mathcal{U}}$, γ
Output: motion primitive $\mathbf{MP}$

1: $(x_{ref}(\cdot), u_{ref}(\cdot)) \leftarrow$ compute_reference_trajectory$(f(x,u,w), x_0, \mathcal{X}_f, \gamma, N, t_f, \bar{\mathcal{S}}, \bar{\mathcal{U}})$ ▷ see Sec. VI-B
2: $(A_k, B_k) \leftarrow$ linearize_and_discretize$(f(x,u,w), x_{ref}(\cdot), u_{ref}(\cdot), \Delta\tau)$ ▷ see Sec. VI-B
3: $\mathcal{X}(\tau_{MH}) \leftarrow \mathcal{X}_f$
4: **for** $l = M, \dots, 1$ **do**
5: $\quad \bar{\mathcal{X}}(\tau_{lH}) \leftarrow$ reduce_zonotope_order$(\mathcal{X}(\tau_{lH}), \bar{r})$ ▷ see Sec. VI-C
6: $\quad \mathcal{X}(\tau_{(l-1)H}) \leftarrow$ compute_backwards_reachable_set$(\bar{\mathcal{X}}(\tau_{lH}), A_k, B_k, \mathcal{U}, H)$ ▷ see Sec. VI-D and VI-E
7: $\quad \mathcal{X}^*(\tau_{(l-1)H}) \leftarrow$ optimal_rescaling$(\mathcal{X}(\tau_{(l-1)H}), \bar{\mathcal{X}}(\tau_{lH}), f(x,u,w), \mathcal{W}, \mathcal{S}, H)$ ▷ see Sec. VI-F
8: $\quad \mathcal{R}_{[\tau_{(l-1)H}, \tau_{lH}], u_{ver}, \mathcal{W}}(\mathcal{X}^*(0)) \leftarrow$ forward reachable set $\mathcal{R}_{[0, H\,\Delta\tau], u_{ver}, \mathcal{W}}(\mathcal{X}^*(\tau_{(l-1)H}))$
9: **end for**
10: $\mathbf{MP} \leftarrow \{x_{ref}(\cdot), u_{ref}(\cdot), t_f, \mathcal{X}^*(0), \mathcal{X}_f, u_{ver}(\cdot), \mathcal{R}_{[0,t_f], u_{ver}, \mathcal{W}}(\mathcal{X}^*(0))\}$

constraint sets $\bar{\mathcal{S}} \subseteq \mathcal{S}$ and $\bar{\mathcal{U}} \subseteq \mathcal{U}$. We denote the resulting reference trajectory by $x_{ref}(\cdot) = \xi(x_0, u_{ref}(\cdot), 0, \cdot)$.

We then linearize the system along $x_{ref}(\cdot)$ at $\tau_k' = \frac{1}{2}(\tau_{k+1} - \tau_k)$, $k \in \{0, \dots, N-1\}$, (function linearize_and_discretize in line 2 of Alg. 2) to obtain the system matrices

$$A_{c,k} = \left.\frac{\partial f(x,u,0)}{\partial x}\right|_{\substack{x = x_{ref}(\tau_k') \\ u = u_{ref}(\tau_k')}}, \qquad (19)$$

$$B_{c,k} = \left.\frac{\partial f(x,u,0)}{\partial u}\right|_{\substack{x = x_{ref}(\tau_k') \\ u = u_{ref}(\tau_k')}}.$$

Since the inputs are constant in each time interval, we can treat the system as a discrete-time, linear system as in (10) with

$$A_k = e^{A_{c,k}\Delta\tau}, \; B_k = \int_0^{\Delta\tau} e^{A_{c,k}t} \mathrm{d}t \; B_{c,k}. \qquad (20)$$

To improve computational efficiency during the following backward reachable set computation and optimization, we group the N pieces of the reference trajectory into M parts consisting of H time steps $\Delta\tau$ each. We iteratively compute the backward reachable set for all time intervals $[\tau_{(l-1)H}, \tau_{lH}]$, $l \in \{1, \dots, M\}$, starting at time $\tau_{MH} = t_f$.

C. Zonotope Order Reduction

Due to the addition of input generators in the backward reachable set computation, the number of generators and therefore the order (see Def. 3) of the reachable sets increases during each iteration. To limit the computational effort, we restrict the generator number in each iteration by under-approximating the reachable set $\mathcal{X}(\tau_{lH})$ with a zonotope $\bar{\mathcal{X}}(\tau_{lH}) \subseteq \mathcal{X}(\tau_{lH})$ of a maximum number of $\bar{r}$ generators, as illustrated in Fig. 3(b) and done in function reduce_zonotope_order (line 5 of Alg. 2). The maximum number of generators is a design parameter, which is used for a trade-off between approximation error and computational effort. There exists a number of methods to compute an enclosing zonotope with fewer generators, see, e.g., [75]–[77]. However, for our case, we need an under-approximative approach, which is harder to compute.

Given a zonotope $\mathcal{X} = \langle c, g^{(1)}, \dots, g^{(r)} \rangle$ we first sort the generators according to their length:

$$\|g^{(a_1)}\|_2 \geq \dots \geq \|g^{(a_r)}\|_2, \qquad (21)$$

where $a \in \mathbb{N}^r$ is a vector of indices that defines the order. To obtain a reduced-order zonotope with $\bar{r} \geq n$ generators, we keep the $p = \bar{r} - n$ best generators and under-approximate the zonotope defined by the remaining generators with a box:

$$\bar{\mathcal{X}} = \langle c, g^{(a_1)}, \dots, g^{(a_p)}, \delta_1 e^{(1)}, \dots, \delta_n e^{(n)} \rangle, \qquad (22)$$

where $e^{(i)} \in \mathbb{R}^n$ is the i-th unit vector. The scaling factors $\delta_1, \dots, \delta_n$ are determined by the following linear program:

$$\max_{\delta_1, \dots, \delta_n} \sum_{i=1}^{n} \delta_i \qquad (23)$$

$$\text{s.t.} \quad \langle \mathbf{0}, e^{(1)}\delta_1, \dots, e^{(n)}\delta_n \rangle \subseteq \langle \mathbf{0}, g^{(a_{p+1})}, \dots, g^{(a_r)} \rangle, \qquad (24)$$

$$\delta_i \geq 0, \quad \forall i \in \{1, \dots, n\}, \qquad (25)$$

where $\mathbf{0}$ denotes a vector containing only zeros. Zonotope containment in (24) is checked by converting the zonotope $\mathcal{X}$ to a polytope in half-space representation according to [76, Theorem 2.1]:

$$\mathcal{X} = \left\{ x \in \mathbb{R}^n \middle| C_{\mathcal{X}} x \leq d_{\mathcal{X}} \right\}, \qquad (26)$$

with $C_{\mathcal{X}} \in \mathbb{R}^{q_{\mathcal{X}} \times n}$, $d_{\mathcal{X}} \in \mathbb{R}^{q_{\mathcal{X}}}$. To check if $\bar{\mathcal{X}}$ is located inside $\mathcal{X}$, we use the following proposition (see [78, Theorem 2] for a proof):

Proposition 1: The zonotope $\bar{\mathcal{X}} = \langle \bar{c}, \bar{g}^{(1)}, \dots, \bar{g}^{(\bar{r})} \rangle$ is contained in a set $\mathcal{X}$ if the projection of $\bar{\mathcal{X}}$ onto the normal vectors of the half-spaces from the representation (26) satisfy the inequality constraints in (26):

$$C_{\mathcal{X}} \bar{c} + \sum_{i=1}^{\bar{r}} |C_{\mathcal{X}} \bar{g}^{(i)}| \leq d_{\mathcal{X}}. \qquad (27)$$

For high-dimensional zonotopes $\mathcal{X}$ or for sets with many generators, computing a half-space representation can become computationally challenging. In this case, the zonotope containment problem (24) can also be solved using the approaches from [79] or [80].

D. Backward Reachable Set

Let us now describe how to compute an estimation of the backward reachable set based on the linearized dynamics for each of the larger time intervals $[\tau_{(l-1)H}, \tau_{lH}]$, $l \in \{1, \dots, M\}$ (see Sec. VI-B). For each

time interval, we apply (13) to compute the backward reachable set as illustrated in Fig. 3(b) and done in function compute_backwards_reachable_set (line 6 of Alg. 2). Since we use a reference trajectory, we cannot use the whole inputs for the feedback controller, but have to reduce the available inputs by the inputs already used for the reference trajectory. We do this by introducing for each input generator $g_u^{(j)}$ a scaling factor $\phi_j(\tau_i) := 1 - |\theta_j(\tau_i)| \in [0,1]$, where $\theta_j(\tau_i) \in [-1,1]$ are chosen such that $c_u + \sum_{j=1}^{r_u} \theta_j(\tau_i) g_u^{(j)} = u_{ref}(\tau_i)$. By substituting the input generators by their scaled counterparts $\phi_j(\tau_i) g_u^{(j)}$, the backward reachable set $\mathcal{X}(\tau_{(l-1)H})$ starting from the under-approximated previous set $\bar{\mathcal{X}}(\tau_{lH}) = \langle \bar{c}_{lH}, \bar{g}_{lH}^{(1)}, \dots, \bar{g}_{lH}^{(\bar{r})} \rangle$ can be expressed as a zonotope analogous to (14):

$$\begin{aligned}\mathcal{X}(\tau_{(l-1)H}) = \Big\{ x \in \mathbb{R}^n \Big| x = x_{ref}(\tau_{(l-1)H}) + \sum_{i=1}^{\bar{r}} \alpha_i \bar{A}^{-1} \bar{g}_{lH}^{(i)} \\ - \sum_{i=(l-1)H}^{lH-1} \sum_{j=1}^{r_u} \beta_{i,j} \bar{A}^{-1} \bar{B}_i \phi_j(\tau_i) g_u^{(j)}, \\ \alpha_i, \beta_{i,j} \in [-1,1] \Big\} \\ =: \Big\{ x \in \mathbb{R}^n \Big| x = \hat{c} + \sum_{i=1}^{\hat{r}} \hat{\alpha}_i \hat{g}^{(i)},\ \hat{\alpha}_i \in [-1,1] \Big\},\end{aligned}$$

where

$$\begin{aligned}\hat{c} &:= x_{ref}(\tau_{(l-1)H}), \\ \hat{g}^{(i)} &:= \bar{A}^{-1} \bar{g}_{lH}^{(i)},\ \forall i \in \{1, \dots, \bar{r}\}, \\ \hat{g}^{(\bar{r}+i\, r_u + j)} &:= -\bar{A}^{-1} \bar{B}_{(l-1)H+i} \phi_j(\tau_{(l-1)H+i}) g_u^{(j)}, \\ \forall i \in \{0, \dots, H-1\},\ &\forall j \in \{1, \dots, r_u\},\end{aligned}$$

with $\hat{r} = \bar{r} + H r_u$.

We calculate the control law for the nonlinear continuous-time systems analogously to the linear case in Sec. V-B by solving a linear program equivalent to (15) so that $\forall i \in \{0, \dots, H-1\}$:

$$\begin{aligned}u_{ver}(x, \tau_{k+i}) = u_{ref}(\tau_{k+i}) \\ + G_u(\tau_{k+i}) [\hat{\alpha}_{\bar{r}+i\, r_u+1}, \dots, \hat{\alpha}_{\bar{r}+(i+1)\, r_u}]^T,\end{aligned}$$

where we again denote the matrix containing all (scaled) input generators by $G_u(\tau_{k+i}) := [\phi_1(\tau_{k+i}) g_u^{(1)}, \dots, \phi_{r_u}(\tau_{k+i}) g_u^{(r_u)}]$. Due to the time-dependent weighting with $\phi_j(\tau_{k+i})$ which accounts for the reference inputs, the matrix $G_u(\tau_{k+i})$ becomes time dependent.

E. Reachability Analysis

After computing the backward reachable set based on the linearized, time-discretized dynamics without disturbances, we have to check if its forward reachable set with the real, nonlinear dynamics actually ends in the desired final set, see Fig. 3(c). If this is not the case, we have to shrink the initial set obtained from the backward reachability analysis by reducing the length of the generators until the forward reachable set ends up inside the desired final set as illustrated in Fig. 3(d).

With most existing reachability approaches, it is not possible to see how the size of the generators of the reachable set depends on the size of the generators of the initial set and of the applied inputs. This would require us to recompute the whole reachable set for each optimization loop, as is the case in our previous works [58] and [59]. To overcome this problem, we subsequently present a new approach based on the conservative polynomialization approach from [62] to understand the correlation between initial and final set so that we can simply minimize an algebraic function for the backward reachable set. The details of this new approach can be found in [81].

Let us start with $\mathcal{X}_0 := \mathcal{X}(\tau_{(l-1)H})$, which we obtain from the backward reachable set computation, and introduce a scaling factor $s_i \in [0,1]$ for every zonotope factor $\hat{\alpha}_i$. The initial set with scaling factors is defined as

$$\mathcal{X}_0 := \Big\{ x \in \mathbb{R}^n \Big| x = \hat{c} + \sum_{i=1}^{\hat{r}} s_i \hat{\alpha}_i \hat{g}^{(i)}, \hat{\alpha}_i \in [-1,1] \Big\}. \quad (28)$$

We use the notation $\mathcal{X}_0 = \mathcal{X}_0(s)$ to denote that the initial set depends on the vector of scaling factors $s = [s_1, \dots, s_{\hat{r}}]^T \in \mathbb{R}^{+\hat{r}}$. The resulting forward reachable set $\mathcal{R}_f(s) = \mathcal{R}_{H\,\Delta\tau, u_{ver}, \mathcal{W}}(\mathcal{X}_0(s))$, using [62] is represented as a polynomial zonotope:

$$\begin{aligned}\mathcal{R}_f(s) = \Bigg\{ x \in \mathbb{R}^n \Bigg| x = c_f + \sum_{j=1}^{v_I} \mu_j g_{I,f}^{(j)} \\ + \sum_{i=1}^{v} \left(\prod_{k=1}^{r} s_k^{E_{k,i}} \right) \left(\prod_{k=1}^{r} \delta_k^{E_{k,i}} \right) g_f^{(i)},\ \delta_k, \mu_j \in [-1,1] \Bigg\}.\end{aligned} \quad (29)$$

The expression in (29) contains the desired analytical relation describing how changes in the initial set $\mathcal{X}_0(s)$ influence the final reachable set $\mathcal{R}_f(s)$. The independent generators $g_{I,f}$ resulting from disturbances and over-approximation errors are the only part which we cannot influence and which therefore always increase the final reachable set $\mathcal{R}_f(s)$. Since they are usually rather small compared to the dependent generators, they do not affect the results much. If they get too large, we can always refine the reachable set computation, see [62]. Let us demonstrate in the following numerical example how the analytical relations between initial set and reachable set can be used to quickly find subsets which satisfy a given constraint:

Example 1: We consider the example shown in Fig. 4, where the initial set is

$$\begin{aligned}\mathcal{X}_0(s) = \Bigg\{ x \in \mathbb{R}^n \Bigg| x = \begin{bmatrix} -8 \\ -2 \end{bmatrix} + \begin{bmatrix} 1 \\ -0.5 \end{bmatrix} s_1 \delta_1 + \begin{bmatrix} 0 \\ 1 \end{bmatrix} s_2 \delta_2, \\ \delta_1, \delta_2 \in [-1,1] \Bigg\},\end{aligned}$$

resulting in the final forward reachable set (see Fig. 4)

$$\begin{aligned}\mathcal{R}_f(s) = \Bigg\{ x \in \mathbb{R}^n \Bigg| x = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} \mu_1 + \begin{bmatrix} 2 \\ 0 \end{bmatrix} s_1 \delta_1 + \\ \begin{bmatrix} 0 \\ 2 \end{bmatrix} s_2 \delta_2 + \begin{bmatrix} 1 \\ 1 \end{bmatrix} s_1^3 s_2 \delta_1^3 \delta_2,\ \mu_1, \delta_1, \delta_2 \in [-1,1] \Bigg\},\end{aligned} \quad (30)$$

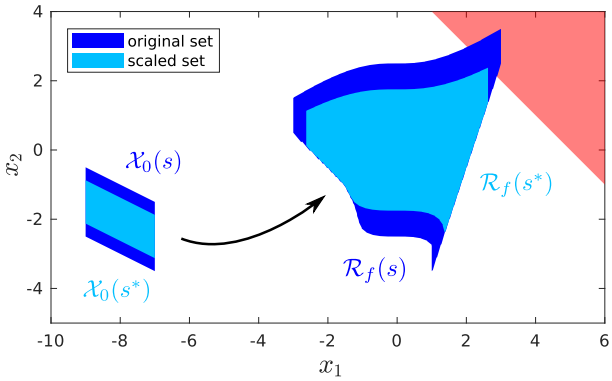


Fig. 4. Visualization of Example 1: The original initial set $\boldsymbol{\mathcal{X}_0(s)}$ and its corresponding final reachable set $\boldsymbol{\mathcal{R}_f(s)}$ with $\boldsymbol{s = [1, 1]^T}$ are depicted in dark blue, where $\boldsymbol{\mathcal{R}_f(s)}$ violates the constraint shown in red. We reduce the scaling factors $\boldsymbol{s^*}$ such that the final set with the new scaling factors satisfies the constraints. The scaled final set $\boldsymbol{\mathcal{R}_f(s^*)}$ and the corresponding scaled initial set $\boldsymbol{\mathcal{X}_0(s^*)}$ are shown in light blue.

Our goal is to find suitable scaling factors $s^* = [s_1^* \ s_2^*]^T$, $s_1^*, s_2^* \in [0, 1]$, such that the final reachable set $\mathcal{R}_f(s^*)$ starting from the scaled initial set $\mathcal{X}_0(s^*)$ satisfies the constraint $[1 \ 1]^T x \leq 5$ that is depicted in red in Fig. 4. Since $\mathcal{X}_0(s)$ and $\mathcal{R}_f(s)$ both depend on the same scaling factors s, we determine s^* such that $\forall x \in \mathcal{R}_f(s^*) : \ [1 \ 1]^T x \leq 5$, which is identical to

$$\max_{x \in \mathcal{R}_f(s^*)} [1 \ 1]^T x \leq 5. \tag{31}$$

Since it holds that

$$\{[1 \ 1]^T x \mid x \in \mathcal{R}_f(s^*)\} \overset{(30)}{=} \{x \in \mathbb{R} \mid 0.5\mu_1 + 2s_1^*\delta_1 + 2s_2^*\delta_2 + 2{s_1^*}^3 s_2^* \delta_1^3 \delta_2, \ \mu_1, \delta_1, \delta_2 \in [-1, 1]\},$$

the constraint in (31) can be equivalently formulated as

$$\max_{\mu_1, \delta_1, \delta_2 \in [-1,1]} 0.5\mu_1 + 2s_1^*\delta_1 + 2s_2^*\delta_2 + 2{s_1^*}^3 s_2^* \delta_1^3 \delta_2 \leq 5 \tag{32}$$

An example for scaling factors satisfying (32) are $s_1^* = 1$ and $s_2^* = 0.625$. We show the reachable set $\mathcal{R}_f(s^*)$ for the scaled initial set $\mathcal{X}_0(s^*)$ in Fig. 4 and see that it in fact satisfies the constraint.

F. Optimization Problem

Using the ideas demonstrated in Example 1, we are now interested in finding scaling factors $s_i \in [0, 1]$ which maximize the volume of the initial set $\mathcal{X}_0(s)$ for which the forward reachable set $\mathcal{R}_f(s)$ still ends up inside the desired under-approximated final set $\mathcal{X}_g := \bar{\mathcal{X}}(\tau_{lH})$. Experiments in Sec. VII have shown that it is advantageous to only scale the generators resulting from the states, i.e., $s_1, \ldots, s_{\bar{r}}$ and keep the scaling factors corresponding to the input generators equal to one. This restriction both ensures that we do not change the resulting control law and reduces the number of optimization variables, which leads to a faster convergence to good solutions.

a) Objective Function: Our goal is to maximize the volume of the initial set $\mathcal{X}_0(s)$, i.e., $\max_s \ V(\mathcal{X}_0(s))$. The volume of a scaled initial set $\mathcal{X}_0(s)$ as defined in (28) can be computed according to [82] as

$$\begin{aligned} V(\mathcal{X}_0(s)) &= 2^n \sum_{i=1}^{n_{comb}} \left|\det\left(\left[s_{\mathcal{G}_1^{(i)}} g^{(\mathcal{G}_1^{(i)})}, \ldots, s_{\mathcal{G}_n^{(i)}} g^{(\mathcal{G}_n^{(i)})}\right]\right)\right| \\ &= 2^n \sum_{i=1}^{n_{comb}} \left|\det(G_i) \det\left(\operatorname{diag}\left(\left[s_{\mathcal{G}_1^{(i)}}, \ldots, s_{\mathcal{G}_n^{(i)}}\right]\right)\right)\right| \\ &= 2^n \sum_{i=1}^{n_{comb}} \left|\det(G_i)\right| \left(\prod_{j=1}^{n} s_{\mathcal{G}_j^{(i)}}\right), \end{aligned} \tag{33}$$

where $\det(\cdot)$ refers to the determinant of a matrix. We denote by $\left\{g^{(\mathcal{G}_1^{(i)})}, \ldots, g^{(\mathcal{G}_n^{(i)})}\right\}$ the n_{comb} possible n-membered subsets of the generator set $\left\{g^{(1)}, \ldots, g^{(\bar{r})}\right\}$ and $G_i = \left[g^{(\mathcal{G}_1^{(i)})}, \ldots, g^{(\mathcal{G}_n^{(i)})}\right]$. Since for high dimensions computing the volume using (33) can become hard, we can use p-radius minimization or segment length minimization [83] as alternatives to approximate the volume in a scalable way.

b) Constraints: We have to consider three constraints: (i) the final forward reachable set $\mathcal{R}_{H\,\Delta\tau, u_{ver}, \mathcal{W}}(\mathcal{X}_0(s))$ must be located inside the desired under-approximated final set $\bar{\mathcal{X}}(\tau_{lH})$, (ii) the reachable set for the whole time interval must be inside the state constraints $\mathcal{S}$, and (iii) the scaling factors s_i for the state generators must be between zero and one, i.e.,

$$s_i \in [0, 1], \ \forall i \in \{1, \ldots, \bar{r}\}. \tag{34}$$

As we see in (32) of Example 1, checking if the polynomial zonotope $\mathcal{R}_f(s)$ is located inside the target set results in an optimization problem which is computationally expensive to solve. Therefore, we compute a zonotope over-approximation $\bar{\mathcal{R}}_f(s)$ of $\mathcal{R}_f(s)$ from (29) according to [63, Prop. 4]:

$$\bar{\mathcal{R}}_f(s) = \left\{x \in \mathbb{R}^n \middle| x = \bar{c}_f(s) + \sum_{i=1}^{v} \bar{\delta}_i \bar{g}_f^{(i)}(s) + \sum_{j=1}^{v_I} \mu_j g_{I,f}^{(j)}, \ \bar{\delta}_i, \mu_j \in [-1, 1]\right\},$$

where $\bar{c}_f(s)$ and $\bar{g}_f^{(i)}(s)$ are polynomial functions in s. Using Proposition 1, the check if $\bar{\mathcal{R}}_f(s)$ is located inside the half-space representation of $\mathcal{X}_g = \{x \in \mathbb{R}^n | C_{\mathcal{X}_g} x \leq d_{\mathcal{X}_g}\}$, with $C_{\mathcal{X}_g} \in \mathbb{R}^{q_{\mathcal{X}_g} \times n}$, $d_{\mathcal{X}_g} \in \mathbb{R}^{q_{\mathcal{X}_g}}$, simplifies to checking the following inequality:

$$C_{\mathcal{X}_g} \bar{c}_f(s) + \sum_{i=1}^{v} |C_{\mathcal{X}_g} \bar{g}_f^{(i)}(s)| + \sum_{j=1}^{v_I} |C_{\mathcal{X}_g} g_{I,f}^{(j)}| \leq d_{\mathcal{X}_g}. \tag{35}$$

If the computation of the half-space representation of $\mathcal{X}_g$ becomes too complex, we can again use the approximations from [79] or [80]. The same constraints are formed for the reachable sets of intermediate time intervals $[\tau_k, \tau_{k+1}]$, $\forall k \in \{0, \ldots, H-1\}$:

$$C_{\mathcal{S}} \bar{c}_k(s) + \sum_{i=1}^{v} |C_{\mathcal{S}} \bar{g}_k^{(i)}(s)| + \sum_{j=1}^{v_I} |C_{\mathcal{S}} g_{I,k}^{(j)}| \leq d_{\mathcal{S}}, \tag{36}$$

where $\bar{c}_k(s)$, $\bar{g}_k^{(i)}(s)$, and $g_{I,k}^{(j)}$ are the center and generators of the zonotope over-approximation of the reachable set of the time interval $[\tau_k, \tau_{k+1}]$.

c) Optimization Problem: With the objective function (33) and the constraints (34)–(36) as derived above, the optimization problem becomes a nonlinear program:

$$\max_{s_1,\dots,s_{\bar{r}}} \; 2^n \sum_{i=1}^{n_{comb}} |\det(G_i)| \left(\prod_{j=1}^{n} s_{\mathcal{G}_j^{(i)}} \right) \tag{37}$$

$$\text{s.t.} \;\; C_{\mathcal{X}_g} \bar{c}_f(s) + \sum_{i=1}^{v} |C_{\mathcal{X}_g} \bar{g}_f^{(i)}(s)| + \sum_{j=1}^{v_I} |C_{\mathcal{X}_g} g_{I,f}^{(j)}| \leq d_{\mathcal{X}_g},$$

$$C_{\mathcal{S}} \bar{c}_k(s) + \sum_{i=1}^{v} |C_{\mathcal{S}} \bar{g}_k^{(i)}(s)| + \sum_{j=1}^{v_I} |C_{\mathcal{S}} g_{I,k}^{(j)}| \leq d_{\mathcal{S}},$$

$$\forall k \in \{0,\dots,H-1\},$$

$$s_i \in [0,1], \; \forall i \in \{1,\dots,\bar{r}\}.$$

Since the computation of the forward reachable set is not required during optimization, (37) can be solved efficiently. The optimization is performed in function optimal_rescaling in line 7 of Alg. 2.

VII. Numerical Example

We demonstrate our approach for an autonomous vehicle. As a model, we choose the kinematic car model from [57], which covers the most important dynamics:

$$\dot{v} = a + w_1, \quad \dot{\Psi} = b + w_2, \quad \dot{x} = v\cos(\Psi), \quad \dot{y} = v\sin(\Psi),$$

where the states v, Ψ, x, and y are the velocity, orientation, and positions in x and in y directions, respectively. The acceleration a and the normalized steering angle b are the inputs, and w_1 and w_2 are additive disturbances. They are constrained by the intervals $a \in [-9.81, 9.81]\frac{m}{s^2}$, $b \in [-0.4, 0.4]\frac{rad}{s}$, $w_1 \in [-0.5, 0.5]\frac{m}{s^2}$, and $w_2 \in [-0.02, 0.02]\frac{rad}{s}$.

A. Safety Controller

For the safety controller, we construct a maneuver automaton consisting of several motion primitives, each computed with our backward-reachable-set-based control approach from Alg. 2. Due to space limitations, we present the details for a single motion primitive and show how multiple motion primitives are used for planning at the end of this section. We choose a *turn left* maneuver based on the one from [57], where we consider the final set $\mathcal{X}_f = [19.8, 20.2]\frac{m}{s} \times [0.18, 0.22]rad \times [19.67, 20.07]m \times [1.79, 2.19]m$. We are looking for a set $\mathcal{X}_0$ around the desired initial state $x_0 = \left[20\,\frac{m}{s}, 0\,rad, 0\,m, 0\,m\right]^T$, which is as large as possible and for which all trajectories can be steered to the final set after one second. We divide the one second time horizon into $N = 40$ time steps, and consider $M = 10$ time intervals of horizon $H = 4$ time steps, which results in $\Delta t = 0.1\,s$ and $\Delta\tau = 0.025\,s$. During the under-approximation of the reachable sets, we reduce the number of generators to $\bar{r} = 12$.

The car dynamics are independent of the absolute orientation and position. Therefore, to concatenate motion primitive $\mathbf{MP}_j$ to $\mathbf{MP}_i$, we can transform the initial set $\mathcal{X}_0^{(j)}$ of $\mathbf{MP}_j$ by

$$T(\mathcal{X}_0^{(j)}) = rot(\Psi_f^{(i)})\mathcal{X}_0^{(j)} \oplus [0, \Psi_f^{(i)}, x_f^{(i)}, y_f^{(i)}]^T,$$

where $\Psi_f^{(i)}, x_f^{(i)}, y_f^{(i)}$ refers to the end states of the reference trajectory of $\mathbf{MP}_i$ and $rot(\Psi_f^{(i)})$ a rotation matrix, which rotates the x and y states by $\Psi_f^{(i)}$. Therefore, for building a maneuver automaton, it suffices if all motion primitives start at the origin for Ψ, x, and y dimensions and only the velocity has to be discretized.

We implement[1] the control approach in MATLAB, where we use CORA [84] for the reachable set computations and ACADO [85] for optimizing the reference trajectory. The computations are performed on a computer with a 3.1 GHz Intel dual-core i7 processor and 16 GB memory. The offline computation of the controller takes around 70 seconds without using parallel computations. When we solve this example without the analytical relations between initial set and final set, the numerical optimization algorithm requires us to compute over fifty reachable sets for each time step, compared to only one when using the analytical relations.

We show the resulting reachable sets for the *turn left* maneuver in Fig. 5. We see that the final set is much smaller than the initial set. Therefore, we would be able to connect the maneuver with itself in a maneuver automaton.

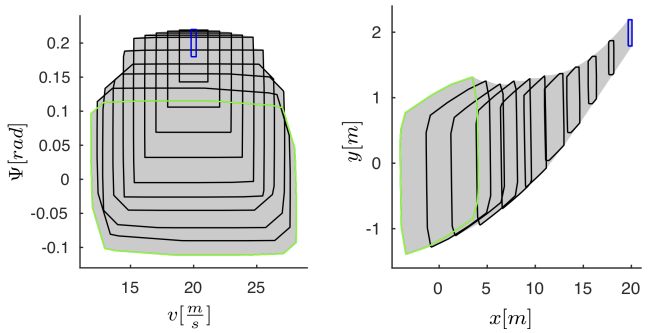


Fig. 5. Reachable sets of the safety controller for a *turn left* maneuver projected onto the (v, Ψ) and (x, y) planes. The initial set is shown in green and the final set in blue.

B. Optimal Controllers

For comparison and a better illustration, we present two different online controllers: first, an LQR-based tracking controller and second, a model predictive controller (MPC).

a) LQR Tracking Controller: Instead of using just a single LQR controller, we compute p of them with different weights $Q_{LQR}^{(i)}, R_{LQR}^{(i)}$, $i \in \{1,\dots,p\}$, and select the best one online. To compute the LQR controllers, we use the reference trajectory $x_{ref}(\cdot)$ and reference input $u_{ref}(\cdot)$ from the current motion primitive and linearize the system along the reference trajectory using (19):

$$u_{LQR}(x(t)) = u_{ref}(t) + K^{(i)}(t)(x(t) - x_{ref}(t)).$$

Based on the linearized dynamics, we compute the LQR matrices $K^{(i)}(t)$, $i \in \{1,\dots,p\}$, by solving the algebraic Riccati equation [86] for the weights $Q_{LQR}^{(i)}, R_{LQR}^{(i)}$.

For our example, we compute $p = 2$ controllers, where without loss of generality we keep the state weights constant as

[1]The implementation will be included in the next version of the AROC toolbox, available at https://aroc.in.tum.de.

$Q_{LQR} = \mathrm{diag}([0.2, 10, 31.2, 1])$ and choose the input weights as $R^{(1)}_{LQR} = \mathrm{diag}([50, 170])$ and $R^{(2)}_{LQR} = \mathrm{diag}([60, 200])$. Lower input weights lead to more aggressive controllers which apply higher inputs to reduce any tracking errors faster and vice versa.

At any point during the online application, we compute the reachable set and check the constraints (7)–(9) for each controller $K^{(i)}(t)$. Afterwards, we apply the one which satisfies the constraints and has the smallest normalized error to the reference trajectory. This allows us to always aim for the best performance, and the different controllers increase the chances to find one which satisfies the state and input constraints. Using multi-core processors allows us to parallelize the computations. If none of the controllers satisfies the constraints, we switch to the safety controller, as described in Sec. IV.

b) Model Predictive Controller: Our second controller is a standard MPC which tracks the reference trajectory by optimizing over a horizon of $P = 6$. It has a time step size $\Delta t_{MPC} = L\Delta\tau$ which, for computational efficiency, is larger than the internal time step size $\Delta\tau$ of the safety controller by a factor of $L = 2$. Its optimization problem is given by

$$\min_{u_{MPC}(\cdot|\tau_k)} \; x_d(\tau_{k+LP}|\tau_k)^T Q_{MPC} x_d(\tau_{k+LP}|\tau_k) + \sum_{i=0}^{P-1} u_{MPC}(\tau_{k+Li}|\tau_k)^T R_{MPC} u_{MPC}(\tau_{k+Li}|\tau_k), \tag{38}$$

$$\text{s.t. } \forall i \in \{0, \dots, P-1\}:$$
$$u_{MPC}(\tau_{k+Li}|\tau_k) \in \mathcal{U},$$
$$\xi(x(\tau_k), u_{MPC}(\cdot|\tau_k), 0, \tau_{L(i+1)}) \in \mathcal{X}^*(\tau_{k+L(i+1)}),$$

with $x_d(\tau_{k+LP}|\tau_k) := \xi(x(\tau_k), u_{MPC}(\cdot|\tau_k), 0, \tau_{LP}) - x_{ref}(\tau_{k+LP})$. In contrast to the LQR tracking approach, we only need a single MPC, as we can include constraints (7)–(9) directly in the optimization problem of the MPC. As weight matrices we choose $Q_{MPC} = \mathrm{diag}([24, 20, 120, 70])$ and $R_{MPC} = \mathrm{diag}([2, 3])$. By restricting $u_{MPC}(\cdot)$ to be piecewise constant, we are able to solve (38) fast with standard optimal control solvers, like direct multiple shooting algorithms [74].

C. Results of Combining the Safety and Optimal Controllers

We illustrate the combination of safety controller and online controller for the *turn left* maneuver. When concatenating two motion primitives, we know that the trajectories for the second motion primitive start inside the final set $\mathcal{X}_f$ of the first motion primitive. Therefore, we choose the set from which our simulated trajectories start to be the final set $\mathcal{X}_f$ shifted around the desired initial state x_0. We compute for both optimal controllers 100 trajectories which start at random states inside this initial set and show them in Fig. 6 for the LQR tracking controller and the MPC. For the 100 trajectories with 10 steps each, the safety controller takes over 2.2% of the time steps for the LQR controller and 0.6% for the MPC. As desired, we are able to apply the optimal controller most of the time; however, there are instances when the optimal controller becomes unsafe and the safety controller takes over. For both controllers, we choose the same maximum computation time of $t_c = 0.05\,s$,

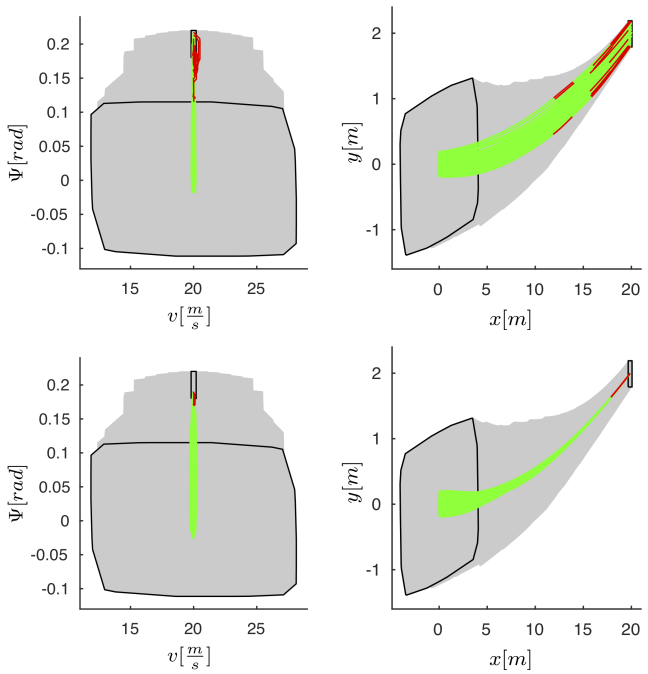


Fig. 6. Illustration of 100 simulations of the LQR tracking controllers (top) and MPC (bottom). All simulations are randomly chosen from the final set shifted to the center of the backwards computed initial set. We show in green when the LQR controllers and MPC are active and in red when the safety controller takes over.

which is large enough that the safety controller never has to take over because the computation takes too long.

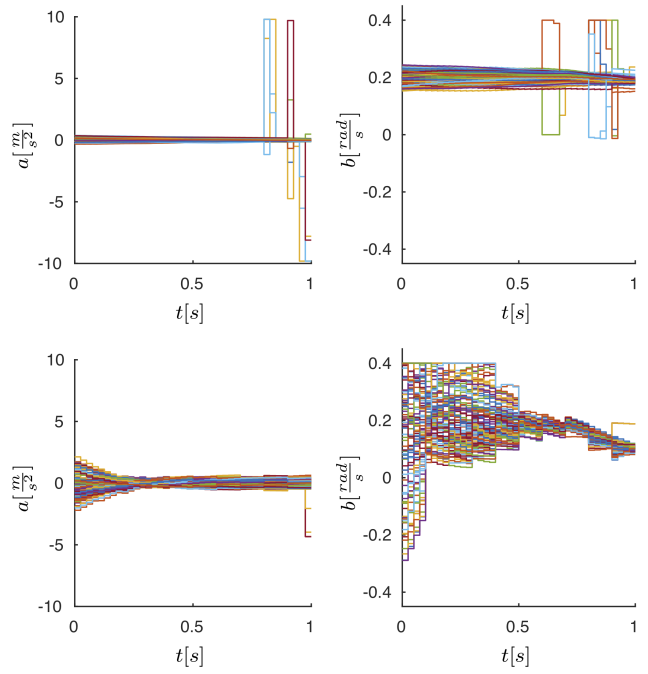


Fig. 7. Plot of the inputs of the 100 simulations of the LQR controllers (top) and MPC (bottom) with safety net. The spikes occur at times when the safety controller takes over.

In Fig. 7, we show the inputs for the safety net controller with the LQR tracking controller and with the MPC, respectively. Aside from the b input for the MPC, the inputs are not very large and do not change drastically most of the time. This

is the desired use of inputs, which of course also depends on the fact that the initial states start in a small set around the initial state of the reference trajectory. The spikes we see in Fig. 7 occur at times when the safety controller takes over. As the safety controller aims to ensure safety for the largest possible set of states, it uses rather large inputs and switches quickly between them. Therefore, if the LQR controllers and MPC are safe, we benefit from their smoother inputs; if they would become unsafe, the safety controller steps in and brings the system back to a safe state.

D. Comparison with Forward Reachable Set Optimizing Controller

We compare the new backward controller with our forward reachable set optimizing controller from [59] in Fig. 8. We consider the same *turn left* maneuver as before, but start from a set with the size of the previous final set around the initial state x_0 (shown in red). We use the controller from [59] to compute the smallest forward reachable set (blue) around the desired final state $\mathrm{center}(\mathcal{X}_f)$. Starting from this final set, we use our new backward algorithm to compute the maximum initial set (green) for which we can steer all states into the final set of the forward algorithm. We see that this backwards computed initial set is much larger than the original initial set. If we use the controller from [59] as a safety controller in the previous subsection and let the trajectories start from its shifted smaller final set, it would have to take over in 79% of the time steps for the LQR controller and in 73% for the MPC. This shows the advantage of the new algorithm, as the larger reachable set allows more freedom for the optimal controller to steer the system without the safety controller having to step in. For a fair comparison, we use the same parameters and algorithms for the reachable set computation for both approaches. If smaller reachable sets from our new approach are desired for easier online planning, we can always limit their size during the controller synthesis or compute subsets (online), based on measurements or final sets of motion primitives, with the methods from [81].

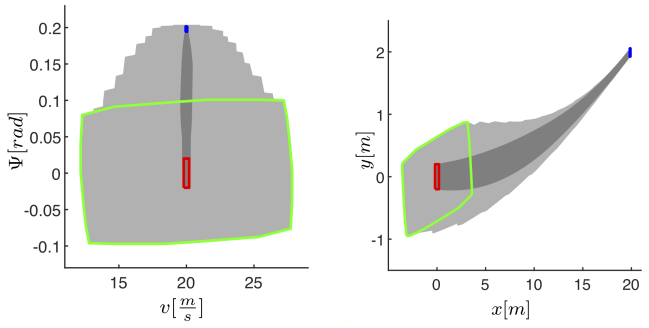


Fig. 8. Comparison with the controller in [59]: Reachable set of controller from [59] shown in dark gray. It starts in the red set and ends in the blue. Our new backward controller starts from the blue set and computes the backward reachable set (light gray), which ends in the green set. This set is much larger than the original initial set, as desired.

E. Application in a Maneuver Automaton

Finally, we illustrate how planning with a maneuver automaton using our motion primitives looks like. We consider the scenario with ID *ZAM_Zip-1_19_T-1* from the CommonRoad database [87]. For better planning, we restrict the lateral size of the motion primitives to the lane width. We compute 25 motion primitives, where each has an average offline computation time of around 80 seconds. Using the motion primitive, we plan for a horizon of $9\,s$ under consideration of the time-dependent occupancies of three other vehicles. The planning takes $1.5\,s$ and we show a simulation of the resulting maneuver at different points in time in Fig. 9. When one gets new measurements, this optimization can be performed in a moving horizon fashion using the updated occupancies of the other traffic participants.

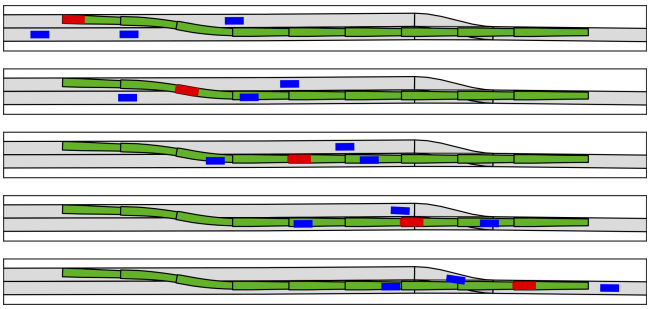


Fig. 9. Visualization of a planned maneuver for an ego vehicle (red) using safe motion primitives (green) under consideration of other vehicles (blue). Simulation of the driven maneuver at times $\mathbf{0}\ s$, $\mathbf{2}\ s$, $\mathbf{4}\ s$, $\mathbf{6}\ s$, $\mathbf{8}\ s$ (top to bottom).

VIII. Discussion

Online Computational Complexity

For the online complexity, we have to distinguish the three parts of our safety net framework: the optimal controller, the safety controller, and the switching logic. The complexity of the online controller depends on the applied control approach and is independent of the presented approach. For example, the LQR approach results in a simple matrix vector multiplication with computational complexity of $\mathcal{O}(n\,m)$, with n denoting the number of states and m denoting the number of inputs. Our safety controller requires the solution of a linear program to obtain the parameters $\hat{\alpha}$ and then some matrix vector multiplications to obtain the input. As shown, e.g., in [88], linear programs can be solved with polynomial time complexity in the number of optimization variables and constraints.

The last part is the switching logic, which requires the computation of the reachable set for the current state. This is only performed for the short time horizon $[t_k - t_c, t_{k+1}]$, i.e., a single controller time step plus the allocated computation time t_c for online controller computation and reachable set computation, and only for a single initial state. Therefore, this can be computed fast and by using the reachability algorithm from [76], which has a computational complexity of $\mathcal{O}(n^3)$ and which has been successfully applied for the online verification of an autonomous vehicle in [89].

This is also demonstrated in our numerical example, where all online computations are performed in the allocated time $t_c = 0.05\,s$. Note that this was done with a standard MATLAB implementation of the reachable set algorithm and the computation time could be significantly reduced with an optimized implementation and faster programming languages like C++.

Offline Computational Complexity

We cannot provide fixed complexity bounds for the offline complexity, as it relies on nonlinear programming for which no complexity bounds exist. Since we cannot bound the number of iterations of the nonlinear programming algorithm, we cannot bound the overall complexity. Let us still give an idea of the complexity of the different parts.

The computation of the reference trajectory is done by solving a single optimal control problem. Even though it is a nonlinear optimization problem, this can be solved fast in practice, especially when restricting the inputs to be piecewise constant. The backward reachable set computation of the linearized dynamics involves only matrix multiplications, whose computational complexity is less than $\mathcal{O}(n^3)$, if $n \geq m$.

The forward reachability algorithm has a computational complexity of $\mathcal{O}(n^5)$ [62]. During the controller optimization, we solve a nonlinear program, where we cannot bound the number of iterations. In each iteration, however, we do not have to recompute the reachable set, only the approximation of the reachable set based on the scaling factors, which has a computational complexity of just $\mathcal{O}(n^2)$ [81].

Optimality

Optimality is ensured by the applied optimal, unverified controller and consequently depends on the used controller type. While we still try to provide optimal solutions with the safety controller, this is not as relevant, as its main objective is to ensure safety. As we rely for the safety controller on nonlinear programming algorithms that do not guarantee convergence to a global optimal solution, we can only expect to obtain a local optimum. In fact, there is no efficient method able to obtain globally optimal controllers for disturbed, nonlinear systems [90], [91]. Most optimal control approaches only consider open-loop dynamics or only undisturbed feedback controllers [92], [93]. With our new approach however, we are able to maximize the size of the initial set for which we can control all states under observance of constraints on states and inputs to the desired final set despite the effect of disturbances. As a result, we optimize over the set of all possible solutions with all possible disturbance realizations and optimize a controller which maximizes the set for which we can provide guarantees. This is not done in comparable methods, as most other formal approaches consider fixed controllers, e.g., tube-based MPC, or fixed control inputs for the whole set, such as in abstraction-based methods.

IX. Conclusion

We present a novel safety net control approach by combining an unverified optimal controller with a verified safety controller. Most of the time, the optimal controller is active and only if its behavior would become unsafe does the safety controller take over. We are thereby able to ensure safety of optimal controllers which cannot guarantee safety on their own. Since optimal controllers have a better control performance than safety controllers, which always have to consider the worst-case behavior, this combination leads to a better control outcome than pure safety controllers, while still maintaining safety guarantees in our numerical experiments.

We also present a novel way for computing the motion primitives of the safety controller, which is based on a) backward reachable sets and b) circumventing the problem of computing under-approximations of reachable sets, and c) using polynomial zonotopes to obtain an analytical correlation between initial sets and reachable sets. As a result, our approach is the first that is able to formally safeguard an optimal controller for disturbed nonlinear systems under state and input constraints with the use of a set-based safety-net controller whose underlying reachable set computation scales polynomially. Since most computations can be performed offline in advance, the online control is fast and efficient. Since the resulting control law is simple to apply and the controller synthesis does not require a deep understanding of control theory or finding Lyapunov functions or such, the control approach is appealing for practical application. We demonstrated the effectiveness of our approach by safeguarding an LQR tracking controller and an MPC for an autonomous vehicle.

Acknowledgment

The author gratefully acknowledges financial support from the European Commission projects UnCoVerCPS under grant number 643921 and justITSELF under grant number 817629, and the Central Innovation Programme of the German Federal Government under grant ZF4086007BZ8.

References

[1] J. B. Rawlings and D. Q. Mayne, *Model Predictive Control: Theory and Design*. Nob Hill, 2009.

[2] D. Q. Mayne, M. M. Seron, and S. V. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, vol. 41, no. 2, pp. 219 – 224, 2005.

[3] S. V. Raković, B. Kouvaritakis, M. Cannon, C. Panos, and R. Findeisen, "Parameterized tube model predictive control," *IEEE Transactions on Automatic Control*, vol. 57, no. 11, pp. 2746–2761, 2012.

[4] S. V. Raković, B. Kouvaritakis, R. Findeisen, and M. Cannon, "Homothetic tube model predictive control," *Automatica*, vol. 48, no. 8, pp. 1631–1638, 2012.

[5] D. Q. Mayne, E. C. Kerrigan, E. J. van Wyk, and P. Falugi, "Tube-based robust nonlinear model predictive control," *International Journal of Robust and Nonlinear Control*, vol. 21, no. 11, pp. 1341–1353, 2011.

[6] S. Singh, A. Majumdar, J.-J. Slotine, and M. Pavone, "Robust online motion planning via contraction theory and convex optimization," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2017, pp. 5883–5890.

[7] J. Bravo, T. Alamo, and E. Camacho, "Robust MPC of constrained discrete-time nonlinear systems based on approximated reachable sets," *Automatica*, vol. 42, no. 10, pp. 1745 – 1751, 2006.

[8] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for linear and hybrid systems*. Cambridge University Press, 2015.

[9] M. de la Pena, A. Bemporad, and C. Filippi, "Robust explicit MPC based on approximate multi-parametric convex programming," in *Proc. of the Conference on Decision and Control*, 2004, pp. 2491–2496.

[10] A. Alessio and A. Bemporad, "A survey on explicit model predictive control," in *Nonlinear Model Predictive Control: Towards New Challenging Applications*, L. Magni, D. M. Raimondo, and F. Allgöwer, Eds. Springer, 2009, pp. 345–369.

[11] D. Raimondo, S. Riverso, C. Jones, and M. Morari, "A robust explicit nonlinear MPC controller with input-to-state stability guarantees," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 9284 – 9289, 2011.

[12] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "LQR-trees: Feedback motion planning via sums-of-squares verification," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, 2010.

[13] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017.

[14] A. K. Winn and A. A. Julius, "Feedback control law generation for safety controller synthesis," in *Proc. of the Conference on Decision and Control*, 2013, pp. 3912–3917.

[15] A. Weiss, C. Danielson, K. Berntorp, I. Kolmanovsky, and S. Di Cairano, "Motion planning with invariant set trees," in *Proc. of the Conference on Control Technology and Applications*, 2017, pp. 1625–1630.

[16] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

[17] M. Zamani, G. Pola, M. Mazo Jr., and P. Tabuada, "Symbolic models for nonlinear control systems without stability assumptions," *IEEE Transactions on Automatic Control*, vol. 57, no. 7, pp. 1804–1809, 2012.

[18] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.

[19] A. Girard, "Controller synthesis for safety and reachability via approximate bisimulation," *Automatica*, vol. 48, no. 5, pp. 947–953, 2012.

[20] Y. Li, J. Liu, and N. Ozay, "Computing finite abstractions with robustness margins via local reachable set over-approximation," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 1–6, 2015.

[21] J. Liu, N. Ozay, U. Topcu, and R. M. Murray, "Synthesis of reactive switching protocols from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 58, no. 7, pp. 1771–1785, 2013.

[22] P. Nilsson and N. Ozay, "Incremental synthesis of switching protocols via abstraction refinement," in *Proc. of the Conference on Decision and Control*, 2014, pp. 6246–6253.

[23] G. Pola, A. Girard, and P. Tabuada, "Symbolic models for nonlinear control systems using approximate bisimulation," in *Proc. of the Conference on Decision and Control*, 2007, pp. 4656–4661.

[24] S. Sadraddini and C. Belta, "Safety control of monotone systems with bounded uncertainties," in *Proc. of the Conference on Decision and Control*, 2016, pp. 4874–4879.

[25] I. M. Mitchell, "Comparing forward and backward reachability as tools for safety analysis," in *Proc. of the International Conference on Hybrid Systems: Computation and Control*, 2007, pp. 428–443.

[26] F. Immler, M. Althoff, L. Benet, A. Chapoutot, X. Chen, M. Forets, L. Geretti, N. Kochdumper, D. P. Sanders, and C. Schilling, "ARCH-COMP19 category report: Continuous and hybrid systems with nonlinear dynamics," in *Proc. of the International Workshop on Applied Verification of Continuous and Hybrid Systems*, 2019, pp. 41–61.

[27] D. Bertsekas and I. Rhodes, "On the minimax reachability of target sets and target tubes," *Automatica*, vol. 7, no. 2, pp. 233 – 247, 1971.

[28] A. B. Kurzhanski and P. Varaiya, "Ellipsoidal techniques for reachability analysis: internal approximation," *Systems & control letters*, vol. 41, no. 3, pp. 201–211, 2000.

[29] A. Girard, C. Le Guernic, and O. Maler, "Efficient computation of reachable sets of linear time-invariant systems with inputs," in *Proc. of the International Conference on Hybrid Systems: Computation and Control*, 2006, pp. 257–271.

[30] A. Hamadeh and J. Goncalves, "Reachability analysis of continuous-time piecewise affine systems," *Automatica*, vol. 44, no. 12, pp. 3189–3194, 2008.

[31] B. Xue, M. Fränzle, and N. Zhan, "Inner-approximating reachable sets for polynomial systems with time-varying uncertainties," *IEEE Transactions on Automatic Control*, 2019.

[32] M. Korda, D. Henrion, and C. N. Jones, "Inner approximations of the region of attraction for polynomial dynamical systems," *IFAC Proceedings Volumes*, vol. 46, no. 23, pp. 534 – 539, 2013.

[33] B. Xue, M. Fränzle, and N. Zhan, "Under-approximating reach sets for polynomial continuous systems," in *Proc. of the International Conference on Hybrid Systems: Computation and Control*, 2018, pp. 51–60.

[34] M. Li, P. N. Mosaad, M. Fränzle, Z. She, and B. Xue, "Safe over- and under-approximation of reachable sets for autonomous dynamical systems," in *Proc. of the International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2018, pp. 252–270.

[35] B. Xue, Z. She, and A. Easwaran, "Underapproximating backward reachable sets by semialgebraic sets," *IEEE Transactions on Automatic Control*, vol. 62, no. 10, pp. 5185–5197, 2017.

[36] ——, "Under-approximating backward reachable sets by polytopes," in *Proc. of the International Conference on Computer Aided Verification*. Springer, 2016, pp. 457–476.

[37] X. Chen, S. Sankaranarayanan, and E. Ábrahám, "Under-approximate flowpipes for non-linear continuous systems," in *Proc. of the Conference on Formal Methods in Computer-Aided Design*, 2014, pp. 59–66.

[38] E. Goubault and S. Putot, "Forward inner-approximated reachability of non-linear continuous systems," in *Proc. of the International Conference on Hybrid Systems: Computation and Control*, 2017, pp. 1–10.

[39] ——, "Inner and outer reachability for the verification of control systems," in *Proc. of the International Conference on Hybrid Systems: Computation and Control*, 2019, pp. 11–22.

[40] D. Han, A. Rizaldi, A. El-Guindy, and M. Althoff, "On enlarging backward reachable sets via zonotopic set membership," in *Proc. of the International Symposium on Intelligent Control*, 2016, pp. 1–8.

[41] M. Chen, S. L. Herbert, M. S. Vashishtha, S. Bansal, and C. J. Tomlin, "Decomposition of reachable sets and tubes for a class of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 63, no. 11, pp. 3675–3688, 2018.

[42] L. Sha, "Using simplicity to control complexity," *IEEE Software*, vol. 18, no. 4, pp. 20–28, jul 2001.

[43] S. Bak, T. T. Johnson, M. Caccamo, and L. Sha, "Real-time reachability for verified simplex design," in *Proc. of the Real-Time Systems Symposium*, 2014, pp. 138–148.

[44] S. Bak, K. Manamcheri, S. Mitra, and M. Caccamo, "Sandboxing controllers for cyber-physical systems," in *Proc. of the International Conference on Cyber-Physical Systems*, 2011, pp. 3–12.

[45] J. Wolff and M. Buss, "Invariance control design for constrained nonlinear systems," *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 37 – 42, 2005, iFAC World Congress.

[46] M. Kimmel and S. Hirche, "Invariance control with chattering reduction," in *Proc. of the Conference on Decision and Control*, 2014, pp. 68–74.

[47] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[48] P. Wieland and F. Allgöwer, "Constructive safety using control barrier functions," *IFAC Proceedings Volumes*, vol. 40, no. 12, pp. 462–467, 2007.

[49] E. Gilbert and I. Kolmanovsky, "Nonlinear tracking control in the presence of state and control constraints: a generalized reference governor," *Automatica*, vol. 38, no. 12, pp. 2063 – 2073, 2002.

[50] A. Bemporad, "Reference governor for constrained nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 3, pp. 415–419, 1998.

[51] E. Garone, S. D. Cairano, and I. Kolmanovsky, "Reference and command governors for systems with constraints: A survey on theory and applications," *Automatica*, vol. 75, pp. 306 – 328, 2017.

[52] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "Fastrack: A modular framework for fast and guaranteed safe motion planning," in *Proc. of the Conference on Decision and Control*, 2017, pp. 1517–1522.

[53] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2737–2752, 2018.

[54] B. Könighofer, M. Alshiekh, R. Bloem, L. Humphrey, R. Könighofer, U. Topcu, and C. Wang, "Shield synthesis," *Formal Methods in System Design*, vol. 51, no. 2, pp. 332–361, 2017.

[55] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proc. of the Conference on Artificial Intelligence*. AAAI, 2018, pp. 2669–2678.

[56] B. Schürmann and M. Althoff, "Convex interpolation control with formal guarantees for disturbed and constrained nonlinear systems," in *Proc. of the International Conference on Hybrid Systems: Computation and Control*, 2017, pp. 121–130.

[57] ——, "Guaranteeing constraints of disturbed nonlinear systems using set-based optimal control in generator space," in *Proc. of the IFAC World Congress*, 2017, pp. 12 020–12 027.

[58] ——, "Optimal control of sets of solutions to formally guarantee constraints of disturbed linear systems," in *Proc. of the American Control Conference*, 2017, pp. 2522–2529.

[59] ——, "Optimizing sets of solutions for controlling constrained nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 66, no. 3, pp. 981–994, 2021.

[60] E. Frazzoli, "Robust hybrid control for autonomous vehicle motion planning," Ph.D. dissertation, Massachusetts Institute of Technology, 2001.

[61] A. Platzer and E. M. Clarke, "The image computation problem in hybrid systems model checking," in *Proc. of the International Conference on Hybrid Systems: Computation and Control*, 2007, pp. 473–486.
[62] M. Althoff, "Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets," in *Proc. of the International Conference on Hybrid Systems: Computation and Control*, 2013, pp. 173–182.
[63] N. Kochdumper and M. Althoff, "Sparse polynomial zonotopes: A novel set representation for reachability analysis," *IEEE Transactions on Automatic Control*, vol. 66, no. 9, pp. 4043–4058, 2020.
[64] D. Heß, M. Althoff, and T. Sattel, "Formal verification of maneuver automata for parameterized motion primitives," in *Proc. of the International Conference on Intelligent Robots and Systems*, 2014, pp. 1474–1481.
[65] I. Saha, R. Ramaithitima, V. Kumar, G. J. Pappas, and S. A. Seshia, "Automated composition of motion primitives for multi-robot systems from safe LTL specifications," in *Proc. of the International Conference on Intelligent Robots and Systems*, 2014, pp. 1525–1532.
[66] R. G. Sanfelice and E. Frazzoli, "A hybrid control framework for robust maneuver-based motion planning," in *Proc. of the American Control Conference*, 2008, pp. 2254–2259.
[67] A. Majumdar and R. Tedrake, "Robust online motion planning with regions of finite time invariance," in *Algorithmic Foundations of Robotics X*. Springer, 2013, pp. 543–558.
[68] B. Schürmann, D. Heß, J. Eilbrecht, O. Stursberg, F. Köster, and M. Althoff, "Ensuring drivability of planned motions using formal methods," in *Proc. of the Intelligent Transportation Systems Conference*, 2017, pp. 1661–1668.
[69] S. Magdici and M. Althoff, "Fail-safe motion planning of autonomous vehicles," in *Proc. of the International Conference on Intelligent Transportation Systems*, 2016, pp. 452–458.
[70] C. Combastel, "A state bounding observer based on zonotopes," in *Proc. of the European Control Conference*, 2003, pp. 2589–2594.
[71] M. Althoff and J. J. Rath, "Comparison of guaranteed state estimators for linear time-invariant systems," *Automatica*, vol. 130, p. 109662, 2021.
[72] E. B. Saff and A. D. Snider, *Fundamentals of Matrix Analysis with Applications*. John Wiley & Sons, 2015.
[73] B. Schürmann, A. El-Guindy, and M. Althoff, "Closed-form expressions of convex combinations," in *Proc. of the American Control Conference*, 2016, pp. 2795–2801.
[74] H. G. Bock and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.
[75] A. Girard, "Reachability of uncertain linear systems using zonotopes," in *Proc. of the International Conference on Hybrid Systems: Computation and Control*, ser. LNCS 3414. Springer, 2005, pp. 291–305.
[76] M. Althoff, "Reachability analysis and its application to the safety assessment of autonomous cars," Ph.D. dissertation, TU München, 2010.
[77] A.-K. Kopetzki, B. Schürmann, and M. Althoff, "Methods for order reduction of zonotopes," in *Proc. of the Conference on Decision and Control*, 2017, pp. 5626–5633.
[78] B. Schürmann, R. Vignali, M. Prandini, and M. Althoff, "Set-based control for disturbed piecewise affine systems with state and actuation constraints," *Nonlinear Analysis: Hybrid Systems*, vol. 36, p. 100826, 2020.
[79] S. Sadraddini and R. Tedrake, "Linear encodings for polytope containment problems," in *Proc. of the Conference on Decision and Control*, 2019, pp. 4367–4372.
[80] A. Kulmburg and M. Althoff, "On the co-np-completeness of the zonotope containment problem," *European Journal of Control*, 2021.
[81] N. Kochdumper, B. Schürmann, and M. Althoff, "Utilizing dependencies to obtain subsets of reachable sets," in *Proc. of the International Conference on Hybrid Systems: Computation and Control*, 2020, pp. 1–10.
[82] G. C. Shephard, "Combinatorial properties of associated zonotopes," *Canadian Journal of Mathematics*, vol. 26, no. 2, pp. 302–321, 1974.
[83] M. Maïga, N. Ramdani, L. Travé-Massuyès, and C. Combastel, "A comprehensive method for reachability analysis of uncertain nonlinear hybrid systems," *IEEE Transactions on Automatic Control*, vol. 61, no. 9, pp. 2341–2356, 2016.
[84] M. Althoff, "An introduction to CORA 2015," in *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015, pp. 120–151.
[85] B. Houska, H. Ferreau, and M. Diehl, "ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.
[86] H. Kwakernaak and R. Sivan, *Linear optimal control systems*. Wiley-interscience New York, 1972.
[87] M. Althoff, M. Koschi, and S. Manzinger, "Commonroad: Composable benchmarks for motion planning on roads," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 719–726.
[88] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica*, vol. 4, no. 4, pp. 373–395, 1984.
[89] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 903–918, 2014.
[90] O. Schütze, "Set oriented methods for global optimization," Ph.D. dissertation, Univ. Paderborn, 2004.
[91] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Athena Scientific Belmont, MA, 2005.
[92] F. L. Lewis, *Optimal Control*. Wiley, 1986.
[93] E. B. Lee and L. Markus, *Foundations of Optimal Control Theory*. Wiley, 1967.

Bastian Schürmann is a Ph.D. candidate in Computer Science at Technische Universität München, Germany. He received a Bachelor of Science in Electrical and Computer Engineering from Technische Universität Kaiserslautern, Germany in 2012; a Master of Science in Electrical Engineering from the University of California, Los Angeles, USA in 2014; and a Master of Science in Engineering Cybernetics from Universität Stuttgart, Germany in 2015. In 2018, he was a visiting student researcher at the California Institute of Technology. His research focuses on combining control theory, reachability analysis, and optimization. Application scenarios include the controlling of safety-critical systems, such as self-driving vehicles and robots collaborating with humans.

Moritz Klischat is a Ph.D. candidate at the Technical University of Munich in the Cyber-Physical Systems Group since 2018. He graduated with an M.Sc. degree in Mechanical Engineering from the Technical University of Munich, Germany, in 2017. His research focuses on motion planning and the generation of safety-critical test cases for automated vehicles.

Niklas Kochdumper received the B.S. degree in Mechanical Engineering in 2015 and the M.S. degree in Robotics, Cognition and Intelligence in 2017, both from Technische Universität München, Germany. He is currently pursuing the Ph.D. degree in computer science at Technische Universität München, Germany. His research interests include formal verification of continuous and hybrid systems, reachability analysis, computational geometry, controller synthesis and electrical circuits.

Matthias Althoff is an associate professor in computer science at Technische Universität München, Germany. He received his diploma engineering degree in Mechanical Engineering in 2005, and his Ph.D. degree in Electrical Engineering in 2010, both from Technische Universität München, Germany. From 2010 to 2012, he was a postdoctoral researcher at Carnegie Mellon University, Pittsburgh, USA, and from 2012 to 2013 an assistant professor at Technische Universität Ilmenau, Germany. His research interests include formal verification of continuous and hybrid systems, reachability analysis, planning algorithms, nonlinear control, automated vehicles, and power systems.