Formal Verification of Maneuver Automata for Parameterized Motion Primitives

Daniel Heß, Matthias Althoff, Thomas Sattel

Abstract— **An increasing amount of robotic systems is developed for safety-critical scenarios, such as automated cars operating in public road traffic or robots collaborating with humans in flexible manufacturing systems. For this reason, it is important to provide methods that formally verify the safety of robotic systems. This is challenging since robots operate in continuous action spaces in partially unknown environments so that there exists no finite set of scenarios that can be verified before deployment. Verifying the safety during the operation based on the current perception of the environment is often infeasible due to the computational demand of formal verification methods. In this work, we compute sets of behaviors for parameterized motion primitives using reachability analysis, which is used to build a maneuver automaton that connects motion primitives in a safe way. Thus, the computationally expensive task of building a maneuver automaton is performed offline. The proposed analysis method provides the whole set of possible behaviors so that it can be verified whether forbidden state-space regions are avoided during the operation of the robot, to e.g. avoid colliding with obstacles. The method is applied to continuous sets of parameterized motion primitives, making it possible to verify infinitely many motions within the parameter space, which to the best knowledge of the authors has not been published before. The approach is demonstrated for collision avoidance of road vehicles.**

I. Introduction

The ever-increasing autonomy of robots makes it possible to operate them in environments that until today are only entrusted to human supervision. Examples are automated road vehicles, automated mining robots, and robots collaborating with humans in flexible manufacturing. Those systems are either safety- or operation-critical and their deployment is only acceptable if one can guarantee that the robot behaves as specified. For this reason, the demand for formal methods, i.e. methods that can mathematically prove properties, is expected to grow.

Testing of robotic systems does not qualify as a formal verification technique since only a few scenarios can be tested, while infinitely many exist. There are many factors that have to be considered for formally verifying robotic systems: Sensor noise, disturbances, uncertain initial states, and changing environments. The first three sources of uncertainty (sensor noise, disturbances, initial states) can in principle be covered by most formal verification techniques, such as theorem proving [31], constraint propagation [9], barrier certificates [32], and reachability analysis [5]. The big challenge, however, is to verify safety of robots in unknown and changing environments, which can only be considered on-the-fly and not offline before deployment. The reason for this is that unexpected events can happen, such as a human worker entering the workspace of a robot, or a traffic participant unexpectedly entering the surrounding region of an automated vehicle. A system designer cannot think of all possible eventualities since infinitely many exist just by arguing that the initial position and velocity of an obstacle is described by real numbers and not by countable integers. Besides the position and velocity, the number, type and shape of other objects is constantly changing.

In this work, we propose to verify the system on-the-fly, i.e. during its operation, to cope with changing environments. This is especially challenging since even the offline verification is a difficult problem (see e.g. [5], [31]). We first consider previous work that formally verifies robotic systems offline. In [30] the dynamic window algorithm for obstacle avoidance is formally verified using theorem proving. The same technique is used in [26] to verify automatic cruise control under the assumption that all vehicles are automated. Other work uses barrier certificates to prove the safety of UAVs for given environments [6] or to find regions of attractions for an Acrobot [27]. Reachability analysis is used in [8], [12] to verify maneuvers of a quadrotor offline. In those works, reachability analysis is reformulated to an optimization problem of Hamilton-Jacobi equations, requiring to discretize the state space, which limits the applicability to systems with no more than $4-6$ continuous state variables.

There are only a few previous works that apply formal methods for on-the-fly verification. In [24], constraint satisfaction of the zero moment point in robotic walking is verified online, but without considering sensor noise, disturbance, and uncertain parameters. Planned motions in automated driving are formally verified in [1] using reachability analysis. Current on-the-fly verification techniques would fail in the event of surprises, which requires shorter verification times. In order to shorten the verification time, we propose to pre-compute reachable sets of motion primitives, which are popular in robotic path planning [11], [17], [25]. The ideas to verify motion primitives, which are later connected to obtain a fully verified maneuver, and to parameterize motion primitives in order to cover continuous sets of behaviors, are approached in [28] and [29] with sum of squares techniques. Our work differs in several aspects: Our method is not limited to a certain type of linear controllers. Instead different, nonlinear controllers may be applied for different maneuvers. Furthermore, our method is intrinsically able to account for

Matthias Althoff is with the Department of Computer Science, Technische Universität München, 85748 Garching, Germany, email: althoff@tum.de

D. Heß and T. Sattel are with the department of Mechanical Engineering, Ilmenau University of Technology, 98693 Ilmenau, Germany. { Daniel.Hess, Thomas.Sattel }@TU-Ilmenau.de

bounded disturbances and measurement noise and to give rigorous bounds on the system state, which reflect the size of disturbances and noise.

Constraints for safe vehicle movement can also be formulated in a robust model predictive control framework [21]. In model predictive control, an optimal input is computed based on solving an optimal control problem for a finite time horizon, where only the first section of the optimal input trajectory is executed. This procedure is repeated so that the solution adapts to the current situation. In tube-based model predictive control (tube-based MPC), concepts from reachability analysis are mixed with model predictive control. Most of the work on tube-based MPC considers linear systems [14], [34], but concepts for nonlinear systems also exist [7]. However, nonlinear tube-based MPC approaches are computationally too expensive to be used for an online application involving fast dynamics with several state variables, such as the vehicle dynamics of this work.

Another line of work provides formal methods to synthesize trajectories based on temporal logic specifications that are provably correct. In [20] temporal logic specifications are used to specify requirements on missions for unmanned aerial vehicles. Trajectories for automated vehicles in static environments are synthesized in [37] within a discretized environment. A discrete environment is also used in [22] to synthesize plans for teams of robots. Another work synthesizes robotic motion for a point mass (double integrator) by bounding the error to an abstract kinematic model and using the abstraction for the planning task [10].

Besides formally verifying safety, other works use a probabilistic setting and predict the variances when following a given reference trajectory [16], [19]. A further line of research is to design robust controllers for tracking concatenated motion primitives to account for sensor noise and disturbances [35]. Note that [35] does not provide deviations from reference trajectories, which are crucial to ensure that no collision occurs, as detailed later.

The paper is organized as follows. In Sec. II we provide a precise problem statement, which is followed by a description of the overall approach in Sec. III. In Sec. IV we present a technique for building maneuver automata which ensure that motion primitives are properly connected for on-the-fly verification. The technique is demonstrated in numerical experiments in Sec. V.

II. Problem Statement

The objective is to formally verify reach-avoid problems of robotic systems on-the-fly. The robotic system is specified by a nonlinear differential equation of the state vector $x(t) \in \mathbb{R}^n$ subject to a continuous input vector $u(t) \in \mathbb{R}^m$:

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

The function $f(x(t),u(t))$ is assumed to be Lipschitz continuous to ensure existence and uniqueness of the solution. The presented approach also works for hybrid (mixed discrete/continuous) systems [36], but to focus on the main innovations, we restrict ourselves to continuous systems. The solution of (1) is denoted by $\zeta(t;x^0,u(\cdot))$, where $x^0 \in \mathbb{R}^n$ is the initial state and $u(\cdot)$ is used to refer to the complete trajectory rather than the specific value at a point in time t denoted by $u(t)$.

The verification problem is to ensure that a set of forbidden states $\mathscr{F}(t) \subset \mathbb{R}^n$ is always avoided while a set of goal states $\mathscr{G}(t) \subset \mathbb{R}^n$ is eventually reached. This reach-avoid property has to hold for any initial state within a set of initial states $\mathscr{X}^0$ and for all input trajectories such that $u(t) \in \mathscr{U}(t)$. The initial set $\mathscr{X}^0$ is used to model measurement uncertainties and $\mathscr{U}(t)$ is used to model sensor noise and disturbances. The verification problem can be more formally defined as

$$\begin{aligned} &\Big(x^0 \in \mathscr{X}^0\Big) \wedge \Big(\forall t (u(t) \in \mathscr{U}(t))\Big) \wedge \\ &\Big(\forall t \forall x^0 \forall u(\cdot) \Big(\zeta(t;x^0,u(\cdot)) \notin \mathscr{F}(t)\Big)\Big) \wedge \\ &\Big(\exists t \forall x^0 \forall u(\cdot) \Big(\zeta(t;x^0,u(\cdot)) \in \mathscr{G}(t)\Big)\Big). \end{aligned} \tag{2}$$

Reach-avoid problems occur in many scenarios. For instance, in human-robot collaborative manufacturing, the robot might want to bring a work piece to a goal region, while avoiding a human worker. In this work, we consider automated vehicles that have to reach goal regions while avoiding other traffic participants. The occupancies of other traffic participants over time are the forbidden regions $\mathscr{F}(t)$, which grow over time to account for uncertainties in their future behavior [3].

III. Overall Approach

In order to verify (2), we compute the set of reachable states

$$\mathscr{R}^e(t) := \left\{ \zeta\left(t;x^0,u(\cdot)\right) \,\middle|\, x^0 \in \mathscr{X}^0, \forall \tilde{t} \in [0,t]\, u(\tilde{t}) \in \mathscr{U}(t) \right\}.$$

When the set of reachable states $\mathscr{R}^e(t)$ does not intersect $\mathscr{F}(t)$ for all times and is fully included in a goal set $\mathscr{G}(t)$ at some point in time, the planned motion of the robot is verified. There are two issues with this approach. First, one cannot compute the set of reachable states $\mathscr{R}^e(t)$ exactly [23]. For that reason, one computes overapproximations $\mathscr{R}(t) \supseteq \mathscr{R}^e(t)$, which is a standard approach (see [5]). The reason for over- rather than underapproximation is that this way one can guarantee that the real system meets (2) when the overapproximation is safe. The disadvantage, however, is that it might not be possible to verify safety when the overapproximation is too large, which is prevented by computing tight overapproximations in this work.

The other issue is that computing reachable sets typically requires too much time. For this reason, we compute reachable sets offline, which are then loaded from a database during the operation of the system. In order to handle the infinite number of possible motion plans, we build a set of so-called motion primitives [11], which can be combined to obtain a plan that drives the system into a goal set while avoiding forbidden sets. We allow that the parameters of the motion primitives are uncertain, making it possible to consider the full parameter space and thus an infinite number

of possible motion primitives. We denote the reachable set of the i-th motion primitive by $\mathscr{R}^i(t)$. Since time is reset when changing a motion primitive, the partial reachable sets always start at time $t = 0$. It is crucial to ensure that the final reachable set of one motion primitive is enclosed in the initial reachable set of the preceding motion primitive, i.e.,

$$\forall i : \mathscr{R}^i(t_i^f) \subset \mathscr{R}_0^{i+1}, \tag{3}$$

where t_i^f is the final time of the i-th motion primitive. This ensures that when starting in $\mathscr{X}^0$, all possible solutions $\zeta(t;x^0,u(\cdot))$ are within the partial reachable sets $\mathscr{R}^i([0,t_i^f])$ for the time intervals $[0,t_i^f]$. Maneuver automata are used to formalize which motion primitives can be connected with each other [11]. In this work, the states of the maneuver automaton refer to the motion primitives and the transitions between states model the connectivity of the motion primitives. Since we directly link the maneuvers with the automaton's states, we do not define a separate maneuver alphabet as in [11]. In this work, a maneuver automaton is defined as a tupel

$$MA = \{\mathscr{M}, \Delta, \mathscr{M}^0, \mathscr{G}\}, \tag{4}$$

where

- $\mathscr{M}$ is a finite set of maneuvers m_i, which are the discrete states of the maneuver automaton;
- Δ is the set of discrete transitions $\Delta \subseteq \mathscr{M} \times \mathscr{M}$. A transition from m_i to m_j is denoted by (m_i, m_j);
- $\mathscr{M}^0 \subset \mathscr{M}$ is a set of possible initial maneuvers;
- $\mathscr{G} \subset \mathscr{M}$ is a set of final, accepting maneuvers.

The semantics of the maneuver automaton is described non-formally. Starting from a maneuver $m^0 \subseteq \mathscr{M}^0$, the possible successive maneuvers are $\{m_i | (m_0, m_i) \in \Delta\}$, which in turn have a set of successive maneuvers. The concatenation of maneuvers is stopped as soon as a newly added maneuver is in the set of accepting maneuvers $\mathscr{G}$. Computing the set of transitions Δ from a set of maneuvers is described in detail in Sec. IV.

The idea of computing partial reachable sets, which are stored in a database is illustrated for the application scenario *automotive collision avoidance* in Fig. 1. During offline computation, the reachable sets of motion primitives are computed (step ①). Note that also the occupancy of the vehicle under all eventualities is stored. When the reachable set of the last point in time t_i^f is within the initial set of another partial reference trajectory, those partial plans can be connected (see (3)) and added in the maneuver automaton (step ②). During the online phase, the reachable sets of other vehicles are computed, which can be computed much faster than for the ego vehicle. The reason for this is that no complicated vehicle models are required and that no trajectory tracker has to be additionally considered [3]. Next, partial motion primitives are combined by the maneuver automaton (step ④) and it is checked whether the occupancies of the other vehicles intersect with the one of the ego vehicle, which is the vehicle for which the verification is performed (step ⑤). If there exist many possible safe solutions, the collision avoidance system is not activated and the human driver operates the vehicle. If, however, only one or a few crucial safe plans remain, one of those safe motion primitives is automatically executed (step ⑥). Due to the previous verification effort, it is ensured that the behavior of the maneuver execution is fully enclosed by the reachable sets and thus verified.

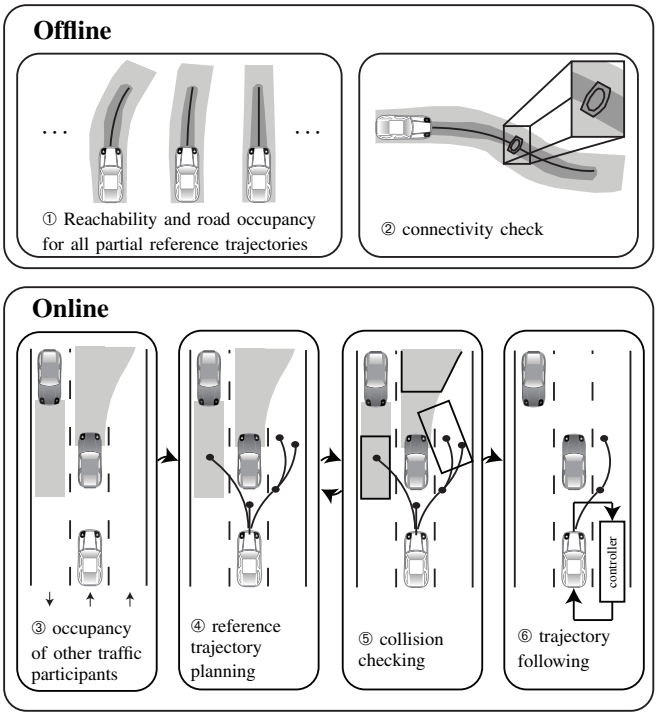

Fig. 1. Overview for formally verifying automated vehicles on-the-fly.

There are previous works on automotive collision avoidance systems, but none of them formally verifies the system considering sensor noise, disturbances, uncertain initial states, and changing environments. It is remarked that formal verification is based on mathematical models. In order to ensure that the exact behavior is included in the model, uncertain parameters and additional uncertain inputs are included [2].

IV. Building of Verified Maneuver Automata

In this section, a maneuver automaton *MA* (see (4)) is built from a set of motion primitives $\mathscr{M}$. A maneuver $m_i \in \mathscr{M}$ is defined in this work such that it represents an infinite number of similar reference trajectories $\{w(p,t)) | p \in \mathscr{P}^i\}$, where $\mathscr{P}^i$ is a set of parameters corresponding to the maneuver m_i (indicated by the same index) and the function $w(p,t)$ maps time t and the parameter vector p to a partial reference trajectory x_{ref} in the state space, which needs to be tracked by a controller. For instance, a simple scalar reference trajectory is $x_{\text{ref}}(t) = p_1 + p_2 t + p_3 t^2$ and a possible set of parameters is $\mathscr{P} = \{p | p_i \in [0,1], i \in \{1,2,3\}\}$. Uncertain parameters are included in the reachability analysis by defining additional state variables x_k with the dynamics $\dot{x}_k = 0$, so that the initial set for the reachability analysis is composed by uncertain

states of the system equations and uncertain parameters: $\mathcal{R}_0^i = \mathcal{X}_0^i \times \mathcal{P}^i$.

The reachable sets associated with two maneuvers m_i and m_j can be concatenated if (3) holds, which is stored in the maneuver automaton by adding the transition (m_i, m_j): $\Delta := \Delta \cup (m_i, m_j)$. An important design question is how to properly choose $\mathcal{R}_0^i$ and how to maximize the number of possible transitions in Δ. A trivial way of obtaining reachable sets $\mathcal{R}^i([0, t_i^f])$ for each maneuver is to guess initial sets $\mathcal{R}_0^i$ for each partial maneuver and check afterwards for which combinations of maneuvers the inclusion property in (3) holds. We found that it is advantageous to guess parallelotopes as initial sets whose axes are aligned with the eigenvectors of the linearized dynamics to take advantage of the eigenmotion of linear systems.

However, rather long partial reference trajectories are required to generate a sufficient number of transitions in the maneuver automaton using the trivial approach. To increase this number, subsets of the maneuver set $\mathcal{M}_{\text{sub}}^i \subset \mathcal{M}$ are constructed such that their nominal reference trajectories (parameter vector is the volumetric center of the parameter set $\mathcal{P}^i$) have a common final state x_{ref}^*, see Fig. 2. In addition, we construct the desired transitions in the maneuver automaton Δ_{des}. In Alg. 1 it is checked if we can find initial sets such that the transitions in Δ_{des} are safe in the sense that the *enclosure condition* (3) always holds. The algorithm starts by guessing initial sets and computing the reachable sets of all maneuvers in $\mathcal{M}$ (line 1 - 4). Next, it is checked for each maneuver if its initial set is a superset of all final sets of previous maneuvers in the maneuver automaton (line 9). For maneuvers where the *enclosure condition* does not hold, the new initial set is chosen as the union of the final sets of previous maneuvers (line 10). This ensures the *enclosure condition* on the expense that a larger initial set has to be chosen. Since this changes the initial assumption of initial sets, the reachable set of the corresponding partial reference trajectory has to be re-computed (line 11). This also alters the previously used final reachable set of the corresponding maneuver, such that previous *enclosure conditions* might not hold anymore. For this reason, the described procedure is repeated in an outer while loop (line 5) until the *enclosure conditions* for all transitions in Δ_{des} hold. Note that convergence is not guaranteed and that it might be required to redesign the trajectory tracking controller and/or the desired transition set Δ_{des} and/or the maneuvers m_i.

Alg. 1 works for any kind of set representation. In this work, zonotopes are used as a set representation since efficient algorithms exist for computing reachable sets based on zonotopes [4]. The algorithm in [4] (with some improvements) is used to implement $\text{REACH}(\mathcal{R}_0^j, m_j)$ in Alg. 1. Its basic principle is to linearize the system dynamics for consecutive time intervals $\tau_k = [t_k, t_{k+1}]$ and overapproximatively obtain the linearization error denoted by $\mathcal{L}(\tau_k)$ for each time interval. Since the linearization is re-computed for each time interval, we obtain linear differential inclusions $\dot{x}(t) \in A_k x(t) + B_k u(t) \oplus \mathcal{L}(\tau_k)$ that are only valid for $t \in \tau_k$, where $\oplus$ denotes the Minkowski addition: $\mathcal{C} \oplus \mathcal{D} = \{c + d | c \in \mathcal{C}, d \in \mathcal{D}\}$. There exist efficient algorithms for computing reachable sets of linear differential inclusions, see e.g. [13]. Zonotopes perform exceptional for reachability analysis since the required operations Minkowski sum and linear map can be efficiently computed. However, there exist no efficient algorithms for computing the enclosure of the union of two zonotopes and to detect whether a zonotope is enclosed by another zonotope, which are both required in Alg. 1. For this reason, new algorithms have been developed to implement $\text{UNION}(\{\mathcal{R}^i(t_i^f) : m_i \in pred(m_j)\})$ and $\text{SUBSET}(\mathcal{R}^i(t_i^f), \mathcal{R}_0^j)$. Due to space limitations we are not able to present those algorithms in detail in this paper.

Algorithm 1 Computes valid Maneuver Automaton

```
VERIFYMA
 1  for each m_i ∈ M
 2      R_0^i := INITIAL(m_i)
 3      R^i := REACH(R_0^i, m_i)
 4      done := 0
 5  while ¬done
 6      done := 1
 7      for each m_j ∈ M
 8          pred(m_j) := {m_i : (m_i, m_j) ∈ Δ}
 9          if ∃m_i ∈ pred(m_j) : ¬SUBSET(R^i(t_i^f), R_0^j)
10              R_0^j := UNION({R^i(t_i^f) : m_i ∈ pred(m_j)})
11              R^j := REACH(R_0^j, m_j)
12              done := 0
13          end
14      end
15  end
```

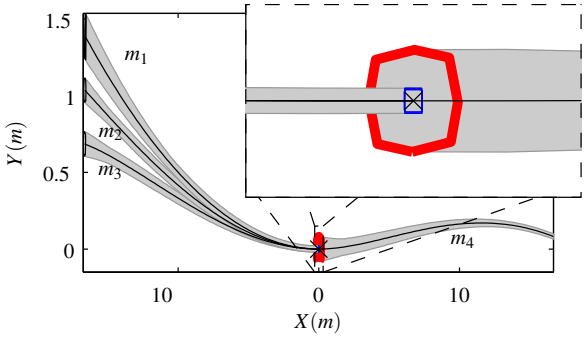

Fig. 2. Valid concatenation of predecessor maneuvers $\{m_1, m_2, m_3\}$ to a successor maneuver m_4. Final sets $\mathcal{R}^i(t_i^f)$ are plotted by thin blue lines, the initial set $\mathcal{R}_0^4$ by thick red lines, the reachable sets in gray color, and the marker '$\times$' refers to the point $x_{\text{ref}}^i(t_i^f) = x_{\text{ref}}^4(t^0)$ where the nominal partial reference trajectories are concatenated.

V. Numerical Examples

The presented construction of a maneuver automaton for the online verification of planned maneuvers is demonstrated for the application scenario *automotive collision avoidance*. For this application scenario we present our choice of parameterized maneuvers, construct the maneuver automaton based

on the selected maneuvers, and demonstrate its integration in an online planning approach. The objective of the collision avoidance system is to drive along an arbitrarily curved road while executing lane changes to avoid lane blocking obstacles.

First, we present our choice of partial reference trajectories. Next, we introduce the dynamic model of the vehicle and of the trajectory tracking controller for following the partial reference trajectories. In a further step, the maneuver automaton is constructed based on the reachable set along partial reference trajectories based on the dynamic model of the closed-loop dynamics of the vehicle. Finally, the online application of the maneuver automaton is demonstrated for a specific traffic situation. In the online phase, an optimization algorithm is used to adjust the parameters of partial reference trajectories.

A. Partial Reference Trajectories

The chosen trajectory tracking controller of the vehicle requires reference values $X^*(t)$ for the x- and $Y^*(t)$ for the y-position, which are combined in the reference trajectory $x_{\text{ref}}(t) := [X^*(t), Y^*(t)]$. Since x- and y-positions cannot be arbitrarily chosen due to dependencies among those variables, we derive $x_{\text{ref}}(t)$ from a point mass traveling at velocity $v^*(t)$, following a curvature $\kappa^*(t)$ with the current orientation $\theta^*(t)$:

$$\begin{aligned} \dot{X}^* &= \cos(\theta^*)\, v^*(t) \\ \dot{Y}^* &= \sin(\theta^*)\, v^*(t) \\ \dot{\theta}^* &= v^*(t)\, \kappa^*(t) \end{aligned} \tag{5}$$

The values $X^*(t)$, $Y^*(t)$, and $\theta^*(t)$ are obtained by integration of the above kinematic equations. Since our application is invariant to position and orientation, the initial values $X^*(0)$, $Y^*(0)$ and $\theta^*(0)$ can be arbitrarily chosen. Different functions can be used to model the independent inputs $v^*(t)$ and $\kappa^*(t)$, such as polynomials, sinusoidals, and Euler spirals. In this work, we use polynomials of first order:

$$\begin{aligned} v^*(t) &= p_1 + p_3 t \\ \kappa^*(t) &= p_2 + p_4 t. \end{aligned} \tag{6}$$

This choice results in spiral trajectories, specifically Euler spirals for the case of $\dot{v}^* = 0$ after integration of (5).

B. Vehicle Model

We describe the vehicle dynamics using relative coordinates. If absolute values would be used, the x-position and y-position vary greatly depending on the choice of parameters $p_1, \ldots, p_4$ of the reference trajectory, which is disadvantageous for constructing the maneuver automaton according to Alg. 1. To describe the behavior in relative coordinates, we define the position vector $\rho = [X, Y]^T$, the orientation ψ, the slip angle β, the absolute velocity v (velocities in x- and y-direction are denoted by v_x, v_y), the slip angle $\beta = \tan^{-1}(v_y/v_x)$, the rotation matrix $R(\cdot) = \begin{pmatrix} \cos(\cdot) & -\sin(\cdot) \\ sin(\cdot) & \cos(\cdot) \end{pmatrix}$, the path tangential error e_t, and the path normal tracking error e_n. The vector of the position error $e_{tn} = [e_t, e_n]^T$ is obtained as $e_{tn} = R(-\theta^*)(\rho - x_{\text{ref}})$. Further errors are the heading error $e_\theta = \psi + \beta - \theta^*$, the velocity error $e_v = v - v^*$, and the turn rate error $e_\omega = \omega - v^* \kappa^*$. We assume that the vehicle has steering angle δ and acceleration/brake force F_b as inputs. The translational and rotational acceleration of the vehicle result from the balance of forces and moments acting on the vehicle, according to the bicycle model formulation [33]:

$$\begin{aligned} \dot{v}_x &= \left(\cos(\delta) F_{x,f} - \sin(\delta) F_{y,f} + F_{x,r}\right)/m + v_y \omega \\ \dot{v}_y &= \left(\sin(\delta) F_{x,f} + \cos(\delta) F_{y,f} + F_{y,r}\right)/m - v_x \omega \\ \dot{\omega} &= \left(l_f \sin(\delta) F_{x,f} + l_f \cos(\delta) F_{y,f} + l_r F_{y,r}\right)/J \end{aligned} \tag{7}$$

The tire forces origin from a linear tire model [33] and the brake balance $b_b \in [0,1]$ that determines the fraction of the braking force applied to the front axle:

$$\begin{aligned} F_{x,f} &= b_b F_b + F_{err,x,f} \\ F_{y,f} &= -C_f \left(\tan^{-1}((v_y + l_f \omega)/v_x) - \delta\right) + F_{err,y,f} \\ F_{x,r} &= (1 - b_b) F_b + F_{err,x,r} \\ F_{y,r} &= -C_r \tan^{-1}((v_y - l_r \omega)/v_x) + F_{err,y,r} \end{aligned} \tag{8}$$

The vehicle state and the reference trajectory are thus fully defined by the combined vector

$$x = [e_t, e_n, e_\theta, e_v, e_\omega, \beta]. \tag{9}$$

The final set of equations is obtained by inserting the tire forces from (8) into (7), which in turn are inserted into the differentiation of the above mentioned formulas determining the state variables in (9). The differentiation of the rotation matrix results in $\frac{d}{dt}\left(R(\phi)p\right) = R(\phi)\left(S(\dot{\phi})p + \dot{p}\right)$, where $S(\phi) = \begin{pmatrix} 0 & -\phi \\ \phi & 0 \end{pmatrix}$. The final set of differential equation of the open-loop dynamics in relation to a reference trajectory is:

$$\begin{aligned} \begin{bmatrix} \dot{e}_t \\ \dot{e}_n \end{bmatrix} &= R(-e_\theta) \begin{bmatrix} v \\ 0 \end{bmatrix} - \begin{bmatrix} v^* \\ 0 \end{bmatrix} - S(\kappa^* v^*) \begin{bmatrix} e_t \\ e_n \end{bmatrix} \\ \dot{e}_\theta &= e_\omega + \dot{\beta} \\ \dot{e}_v &= (\cos(\beta)\dot{v}_x + \sin(\beta)\dot{v}_y) - \dot{v}^* \\ \dot{e}_\omega &= \dot{\omega} - \dot{v}^* \kappa^* - v^* \dot{\kappa}^* \\ \dot{\beta} &= (-\sin(\beta)\dot{v}_x + \cos(\beta)\dot{v}_y)/v \end{aligned} \tag{10}$$

For trajectory tracking we apply input-output linearization as described in [18] so that the evolution of the tracking error is equivalent to a linear second order system in the error-free case (no measurement errors, exact model): $\ddot{e}_{tn} = -k_0 e_{tn} - k_1 \dot{e}_{tn}$. Inserting the controller equations into the open-loop dynamics in (10) results in the set of differential equations of the closed-loop dynamics, which is not presented due to space restrictions. The interested reader is referred to [18] for more information. Further, the equations for the uncertain parameters p_j $(j = 1, \ldots, 4)$ are also integrated as new state variable x_i $(i = 7, \ldots, 10)$ with $\dot{x}_i = 0$ in order to consider uncertain parameters in the reachability analysis as mentioned in Sec. IV.

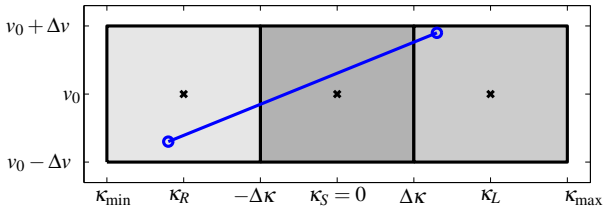

Fig. 3. Parameter space with valid example trajectory. Each parameterized motion primitive allows the uncertain set-trajectory parameters p_1, p_2 to vary inside one of the grey boxes and p_3, p_4 to vary inside another (or the same) box.

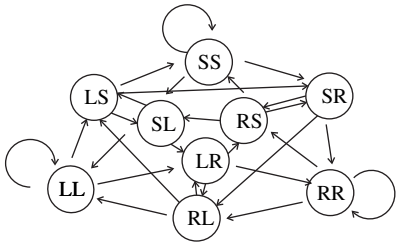

Fig. 4. The resulting maneuver automaton. S denotes the parameter set for straight movement, L for left movement, and R for right movement. Their combination refers to the transition between the parameter sets S, L, and R.

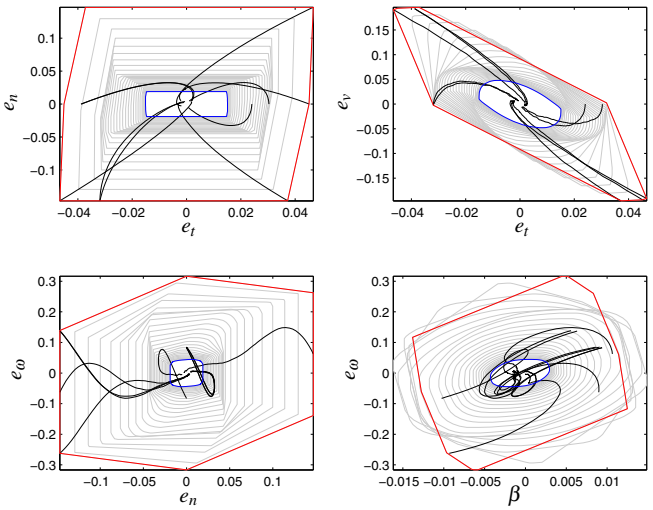

Fig. 5. Reachable sets on which the *enclosure condition* (3) has been positively tested for a transition from the *parameterized RR* maneuver to itself. The initial set of the maneuver is plotted in red, the final set in blue, reachable sets of intermediate time intervals in gray, and exemplary simulations of the system behavior in black. The enclosure condition is fulfilled since the final set is fully contained in the initial set.

C. Maneuver Automaton

In order to generate a maneuver automaton that can be grasped by a human reader, we only introduce three parameter regions for the partial reference trajectories. In principle, this approach would work for arbitrary many parameter regions. Each parameter region has the same size, but a shifted center. The uncertainty around a center is $\mathscr{I} = [-\Delta v, +\Delta v] \times [-\Delta\kappa, +\Delta\kappa]$ and the parameter regions are $R = [v_0, -2\Delta\kappa]^T \oplus \mathscr{I}$, $S = [v_0, 0]^T \oplus \mathscr{I}$, $L = [v_0, +2\Delta\kappa]^T \oplus \mathscr{I}$, where $v_0 = 16.7m/s$, $\Delta v = 0.2m/s$ and $\Delta\kappa = 0.005m^{-1}$ and L stands for left, S for straight, and R for right. The parameters p_1, p_2 may be selected from any of the three parameter space regions so that $[p_1, p_2]^T \in (R \cup S \cup L)$.

In order to connect partial reference trajectories without jumps, it is required that $v_i^*(t_i^f) = v_j^*(0)$ and $\kappa_i^*(t_i^f) = \kappa_j^*(0)$. This condition is not ensured when the set of values for $v^*(t)$ and $\kappa^*(t)$ in (6) grows over time, which is restricted for the particular choice in (6) by selecting the parameters p_3, p_4 as

$$[p_3, p_4]^T \in \left(-[p_1, p_2]^T \oplus (R \cup S \cup L)\right) / t_i^f .$$

A maneuver is denoted by *LS* when the maneuver transitions from the parameter region L to the region S, meaning that a left turn maneuver is followed by a straight maneuver. The line in Fig. 3 is an example for a trajectory which can be categorized as an *RL* maneuver. The overall maximum and minimum parameter values of the three regions are $v_{\min}^* = v_0 - \Delta v$, $v_{\max}^* = v_0 + \Delta v$, $\kappa_{\min}^* = -3\Delta\kappa$, $\kappa_{\max}^* = +3\Delta\kappa$. In lateral direction this corresponds to a region of $a_y \in [-4.2, +4.2] ms^{-2}$ acceleration, thus covering roughly 50% of the physically possible spiral trajectories by only three regions. The desired set of transitions Δ_{des} of the maneuver automaton is chosen such that the graph of the maneuver automaton is fully connected, see Fig. 4. This ensures that parameters of trajectories can be arbitrarily chosen within $[v_{\min}^*, v_{\max}^*]$ and $[\kappa_{\min}^*, \kappa_{\max}^*]$.

For the reachability analysis, the initial sets of the vehicle are chosen with the origin as its center, except for the slip angle, where we use the steady state slip angle. The initial sets $\mathscr{R}_0^i$ are chosen as interval hulls in the eigenvector space with a normalized length of 0.05. Alg. 1 is able to prove all connections to be valid after one iteration. Fig. 5 and 6 show examples of projections of reachable sets to verify the *enclosure condition* (3) together with exemplary simulations that illustrate individual behaviors. All 9 nominal partial reference trajectories are plotted in Fig. 7 together with the area covered by all partial reference trajectories when their parameters are uncertain within the specified parameter space. One can see that due to uncertain parameters a large region of possible reference trajectories is covered. Out of the 9 partial reference trajectories, we take a closer look at the *RR* maneuver (connecting the parameter region R with the region R) in Fig. 8. The line shows the nominal reference trajectory and the light gray area shows the area covered by possible reference trajectories when the trajectory parameters are uncertain. The dark gray region shows the reachable set when following the nominal reference trajectory. This reachable set can be easily recovered for any reference trajectory within the light gray area since the uncertainty in the reference trajectory parameters is already considered in the reachability analysis.

D. Online Planning

During the online planning, the steps illustrated in the lower half of Fig. 1 are executed. First, the occupancy of other vehicles over time is computed, then partial maneuvers are combined according to the generated maneuver automaton, and finally it is checked which of the full maneuvers

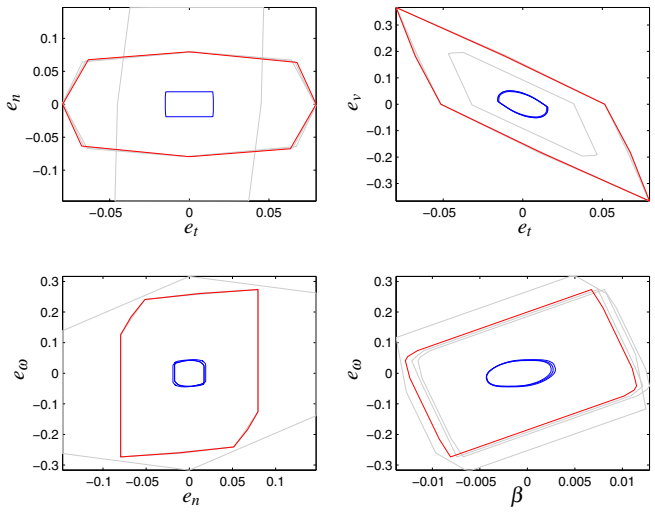

Fig. 6. Reachable sets on which the *enclosure condition* (3) has been tested for all three maneuvers m_i which connect to the *RS*-maneuver. The three initial sets are plotted in gray, the final sets in blue (sometimes overlapping), and the initial set of the *RS*-maneuver in red. The enclosure conditions are fulfilled since the final sets are all included in the initial set of the *RS*-maneuver.

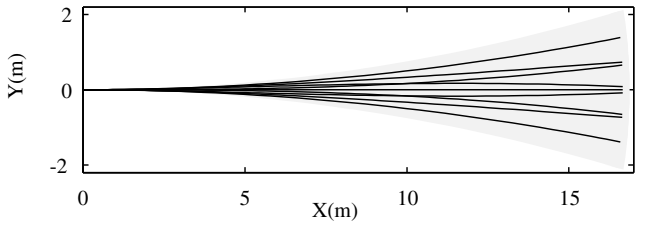

Fig. 7. Position region covered by the 9 parameterized reference trajectories for the full range of parameter values. Black lines show the 9 nominal reference trajectories (parameter value is the volumetric center of the parameter sets $\mathscr{P}^i$) and the gray region shows the coverage of all reference trajectories when the parameters are within $\mathscr{P}^i$.

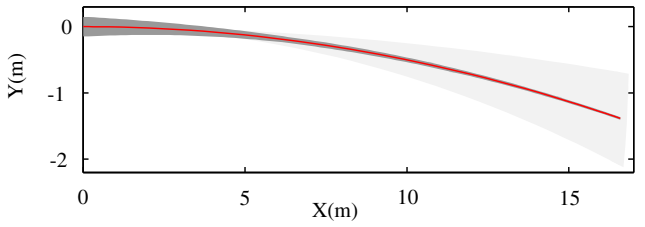

Fig. 8. Closer look at the *RR* maneuver. The red line shows the nominal reference trajectory, the light gray area shows the area covered by the reference trajectory when the parameters are uncertain, and the dark gray area shows the reachable set when following the nominal reference trajectory.

are collision-free. If there is only one or a few crucial collision-free paths, the selected maneuver is executed. The occupancy prediction of other vehicles is already described in [3], the combination of maneuvers according to the maneuver automaton is trivial and can be supported by standard search algorithms (e.g. A*), and collision checks can be computed as described in e.g. [15]. For this reason, we focus on a novel aspect in this work: The optimization of the parameter values of selected partial reference trajectories. Given the sequence of maneuvers, the initial velocity and curvature

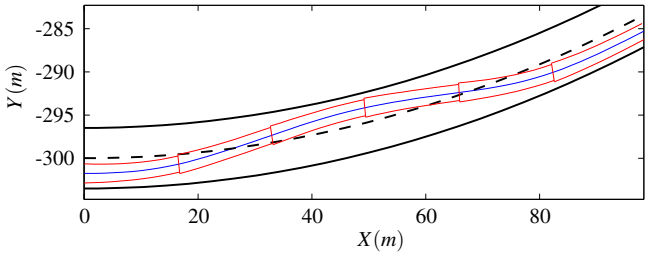

Fig. 9. Evasion maneuver on a curved road. The red areas show the occupancy of the vehicle, the blue line shows the complete reference trajectory assembled from partial reference trajectories.

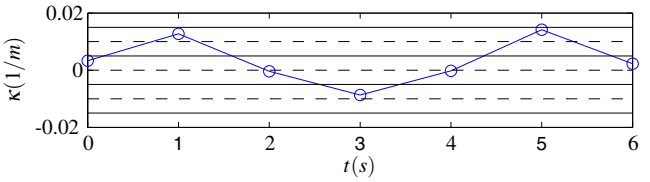

Fig. 10. The values of the curvature κ plotted over time; circles indicate the switch from one partial motion primitive to another.

as well as the final velocity and curvature can be freely chosen within $[v^*_{min}, v^*_{max}]$ and $[\kappa^*_{min}, \kappa^*_{max}]$ without changing the verification result. The only restriction is that the final parameter values of a predecessor maneuver m_i and the initial parameter values of a successor maneuver m_j must be equal: $v^*_i(t^f_i) = v^*_j(0)$, $\kappa^*_i(t^f_i) = \kappa^*_j(0)$ (note that time is reset after changing the maneuver). We optimize q, which is the vector of N velocity and curvature pairs at the beginning of each maneuver. To simplify notation, we introduce the starting times of the maneuvers when time is not reset: $T_i = \sum_{k=1}^{i} t^f_i$, so that the optimization vector can be written as

$$q = [v^*(0), \kappa^*(0), \cdots, v^*(T_i), \kappa^*(T_i), \cdots, v^*(T_N), \kappa^*(T_N)].$$

The cost function is chosen as the integral over the absolute acceleration. The optimization is computed using fmincon in MATLAB. The result of the optimization of a plan for evasion of an obstacle on the right lane of a curved road is shown in Fig. 9 and 10. In Fig. 9 the blue curve is the reference trajectory and the red regions correspond to the area occupied by the vehicle. The occupied area is constructed from the reachable sets projected onto the dimensions e_t, e_n, e_θ and β as presented in [1]. Fig. 10 shows the κ values over time, where circles indicate the switch from one partial motion primitive to another. The approach guarantees safe execution within modeled uncertainties since all possible behaviors are already considered by the preceding reachability analysis.

VI. Conclusion and Future Work

To the best knowledge of the authors, this paper is the first work that describes how to pre-compute reachable sets of robotic systems for an infinite number of possible reference trajectories. A maneuver automaton is automatically generated to decide whether certain partial reference trajectories can be connected such that the system behavior remains within pre-computed reachable sets. Since those partial

reachable sets are computed offline, one can quickly plan behaviors online, which formally guarantee that forbidden regions in the state space, such as a region occupied by an obstacle, will not be entered by the system.

The approach is demonstrated for the application *automotive collision avoidance*. The results show that one can continuously cover the set of relevant behaviors using offline computed reachable sets and thus rigorously determine whether a reference trajectory is collision-free or not. The same approach can be applied for different vehicle models or even different applications. The approach is not limited to a fixed system description and may easily accommodate different control strategies, for example to adapt controller parameters to the distinct requirements of each maneuver primitive. Future work includes the integration of real sensor data from a collaboration with a car manufacturer to see how well the approach works on recorded scenarios, and the implementation of the software in a real vehicle.

VII. Acknowledgments

The authors gratefully acknowledge partial financial support by the German Research Foundation (DFG) under grant number AL 1185/3-1.

References

[1] M. Althoff and J. M. Dolan. Set-based computation of vehicle behaviors for the online verification of autonomous vehicles. In *Proc. of the 14th IEEE Conference on Intelligent Transportation Systems*, pages 1162–1167, 2011.

[2] M. Althoff and J. M. Dolan. Reachability computation of low-order models for the safety verification of high-order road vehicle models. In *Proc. of the American Control Conference*, pages 3559–3566, 2012.

[3] M. Althoff, D. Heß, and F. Gambert. Road occupancy prediction of traffic participants. In *Proc. of the 16th International IEEE Conference on Intelligent Transportation Systems*, pages 99–105, 2013.

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

[5] E. Asarin, T. Dang, G. Frehse, A. Girard, C. Le Guernic, and O. Maler. Recent progress in continuous and hybrid reachability analysis. In *Proc. of the 2006 IEEE Conference on Computer Aided Control Systems Design*, pages 1582–1587, 2006.

[6] A. T. R. Barry, Andrew J.; Majumdar. Safety verification of reactive controllers for UAV flight in cluttered environments using barrier certificates. In *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[7] J. M. Bravo, T. Alamo, and E. F. Camacho. Robust MPC of constrained discrete-time nonlinear systems based on approximated reachable sets. *Automatica*, 42:1745–1751, 2006.

[8] J. Ding, E. Li, H. Huang, and C. J. Tomlin. Reachability-based synthesis of feedback policies for motion planning under bounded disturbances. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2160–2165. IEEE, 2011.

[9] A. Eggers, M. Fränzle, and C. Herde. Application of constraint solving and ode-enclosure methods to the analysis of hybrid systems. In *Proc. of the International Conference on Numerical Analysis and Applied Mathematics*, pages 1326–1330, 2009.

[10] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343–352, 2009.

[11] E. Frazzoli, M. A. Dahleh, and E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *Robotics, IEEE Transactions on*, 21(6):1077–1091, 2005.

[12] J. H. Gillula, H. Huang, M. P. Vitus, and C. J. Tomlin. Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice. In *IEEE International Conference on Robotics and Automation*, pages 1649–1654, 2010.

[13] A. Girard. Reachability of uncertain linear systems using zonotopes. In *Hybrid Systems: Computation and Control*, LNCS 3414, pages 291–305. Springer, 2005.

[14] R. Gonzalez, M. Fiacchini, T. Alamo, J. L. Guzman, and F. Rodriguez. Online robust tube-based MPC for time-varying systems: A practical approach. *International Journal of Control*, 84(6):1157–1170, 2011.

[15] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A hierarchical structure for rapid interference detection. *Computer Graphics*, 30:171–180, 1996.

[16] M. Greytak and F. Hover. Motion planning with an analytic risk cost for holonomic vehicles. In *Proceedings of the 48th IEEE Conference on Decision and Control, held jointly with the 28th Chinese Control Conference.*, pages 5655–5660. IEEE, 2009.

[17] D. J. Grymin, C. B. Neas, and M. Farhood. A hierarchical approach for primitive-based motion planning and control of autonomous vehicles. *Robotics and Autonomous Systems*, 62(2):214 – 228, 2014.

[18] D. Heß, M. Althoff, and T. Sattel. Comparison of trajectory tracking controllers for emergency situations. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 163–170, 2013.

[19] D. Heß and T. Sattel. Double-lane change optimization for a stochastic vehicle model subject to collision probability constraints. In *Proc. of the 14th International IEEE Conference on Intelligent Transportation Systems*, pages 206–211, 2011.

[20] S. Karaman and E. Frazzoli. Linear temporal logic vehicle routing with applications to multi-UAV mission planning. *Journal of Robust and Nonlinear Control*, 21(12):1372–1395, 2011.

[21] E. C. Kerrigan and J. M. Maciejowski. Feedback min-max model predictive control using a single linear program: Robust stability and the explicit solution. *International Journal of Robust and Nonlinear Control*, 14(4):395–413, 2004.

[22] M. Kloetzer and C. Belta. Automatic deployment of distributed teams of robots from temporal logic specifications. *IEEE Transactions on Robotics*, 26(1):48–61, 2010.

[23] G. Lafferriere, G. J. Pappas, and S. Yovine. A new class of decidable hybrid systems. In *Hybrid Systems: Computation and Control*, LNCS 1569, pages 137–151. Springer, 1999.

[24] S. Lengagne, N. Ramdani, and P. Fraisse. Planning and fast replanning safe motions for humanoid robots. *IEEE Transactions on Robotics*, 27(6):1095–1106, 2011.

[25] C. Liu and C. G. Atkeson. Standing balance control using a trajectory library. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3031–3036, 2009.

[26] S. M. Loos, D. Witmer, P. Steenkiste, and A. Platzer. Efficiency analysis of formally verified adaptive cruise controllers. In *Proc. of the 16th International IEEE Conference on Intelligent Transportation Systems*, pages 1565–1570, 2013.

[27] A. Majumdar, A. A. Ahmadi, and R. Tedrake. Control design along trajectories with sums of squares programming. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2013.

[28] A. Majumdar and R. Tedrake. Robust online motion planning with regions of finite time invariance. In *Algorithmic Foundations of Robotics X*, pages 543–558. Springer, 2013.

[29] A. Majumdar, M. Tobenkin, and R. Tedrake. Algebraic verification for parameterized motion planning libraries. In *American Control Conference (ACC), 2012*, pages 250–257. IEEE, 2012.

[30] S. Mitsch, K. Ghorbal, and A. Platzer. On provably safe obstacle avoidance for autonomous robotic ground vehicles. In *Proc. of Robotics: Science and Systems IX*, 2013.

[31] A. Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, 2010. ISBN 978-3-642-14508-7.

[32] S. Prajna. Barrier certificates for nonlinear model validation. *Automatica*, 42(1):117–126, 2006.

[33] R. Rajamani. *Vehicle Dynamics and Control*. Springer, 2006.

[34] S. V. Raković, B. Kouvaritakis, M. Cannon, C. Panos, and R. Findeisen. Parameterized tube model predictive control. *IEEE Transactions on Automatic Control*, 57(11):2746–2761, 2012.

[35] R. G. Sanfelice and E. Frazzoli. A hybrid control framework for robust maneuver-based motion planning. In *American Control Conference, 2008*, pages 2254–2259. IEEE, 2008.

[36] A. van der Schaft and H. Schumacher. *An Introduction to Hybrid Dynamical Systems*. Springer, 2000.

[37] T. Wongpiromsarn, U. Topcu, and R. M. Murray. Receding horizon temporal logic planning. *IEEE Transactions on Automatic Control*, 57(11):2817–2830, 2012.