Safety Assessment of Autonomous Cars using Verification Techniques

Matthias Althoff, Olaf Stursberg and Martin Buss

Abstract— **A common requirement for autonomous cars is a safe locomotion which is evaluated by the method of hybrid verification applied online. The approach checks avoidance of static obstacles and dynamic traffic participants, which are described by imprecise data on their positions and velocities. The nonlinear dynamics of the autonomous car and other traffic participants is conservatively abstracted to Markov chains, which allows the efficient computation of their future positions probabilistically. The result is the probability of a crash for a given time horizon, showing if a given control strategy may lead to unsafe situations.**

I. INTRODUCTION

Ensuring safe trajectories obtained from path planning of autonomous vehicles in dynamic environments is an open research problem. Due to the importance of passenger safety in cognitive and other cars, safety should be assessed in terms of a guarantee. In traffic scenarios, measurements, disturbances and decisions of traffic participants are uncertain. This results in a set of possible initial states, disturbance trajectories, and behavior predictions for each road user. The fact that one has to deal with sets instead of single values of positions and velocities leads to an infinite number of possible outcomes of a traffic scene for a given time horizon. As one cannot simulate all possible behaviors of traffic participants, simulation techniques can only prove that a system is unsafe, but never the opposite. For this reason, methods from hybrid verification are used, which allow to compute the set of all possible behaviors.

Algorithmic verification techniques for hybrid systems [1] are an extension of model checking [2] for discrete systems. Hybrid systems, which combine discrete and continuous dynamics are useful for modeling road traffic due to the interaction of decisions, such as lane changing or turning, with continuous vehicle dynamics. In order to verify a hybrid system, the set of states reachable within a finite or infinite time interval is computed. Informally speaking, the reachable set of a system is the tube that encloses all possible trajectories of the system. If the tube does not intersect a set of dangerous states, the system is called safe. Computations of reachable sets have been performed using level sets [3], ellipsoids [4], polytopes [5], oriented rectangular hulls [6] and zonotopes [7] to represent (conservative approximations) of the tube. Verification results of road traffic problems have been published in [8] and [9]. Shortcomings of simulation-based forward collision avoidance systems are described in [10]. However, to the best knowledge of the authors, no work on algorithmic verification for traffic scenarios that is applied online has been published before. The online application is motivated by the large variety of possible situations appearing in traffic. If offline verification is applied, one can only investigate a certain class of situations, as it is done e.g. for the verification of an automatic cruise controller in [8]. In order to speed up verification for online application, the continuous dynamics of the verified system is simplified to Markov models, see [11]. The same idea has been applied to stochastic hybrid systems [12] and to stochastic aircraft conflict situations [13].

The verification process is illustrated in Fig. 1. The nonlinear differential equations of the autonomous car and other traffic participants are abstracted to linear ones based on a state space discretization (sec. III). Because of the superposition principle of the resulting linear models, their reachable sets can be computed separately for the state- and input-dependent solution (sec. IV). The state-dependent solution is used to further abstract the linear system to a Markov chain for efficient computations of the reachable set during online operation (sec. V). The input-dependent solution is calculated separately for the linear model and is used to update the state of the Markov chain so that the input is considered. Based on the obtained reachable sets of the autonomous car and other traffic participants, a probability for a crash is obtained for any time step k.

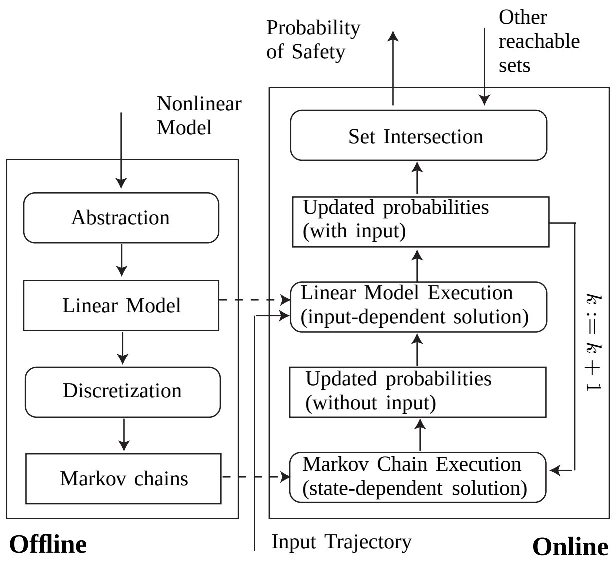

Probability of Safety
Other reachable sets
Nonlinear Model
Set Intersection
Abstraction
Updated probabilities (with input)
Linear Model
Linear Model Execution (input-dependent solution)
$k := k + 1$
Discretization
Updated probabilities (without input)
Markov chains
Markov Chain Execution (state-dependent solution)
Offline
Input Trajectory
Online

Fig. 1. Verification process overview

All authors are with Institute of Automatic Control Engineering (LSR) , Technische Universität München, 80290 München, Germany {althoff,stursberg,mb}@tum.de

II. Investigated scenario

In order to motivate the methods to be presented subsequently, a typical traffic scenario is exemplarily investigated. In this scenario, safety is checked for an autonomous car changing lanes on a straight road in order to circumvent a static obstacle while a car is approaching from the opposite direction on the neighbored lane, as illustrated in Fig. 2. It is assumed, that the other car respects the road traffic regulations, i.e. it does not change its lane and stays below speed limit.

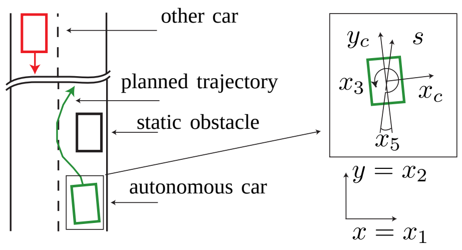


Fig. 2. Verification scenario

A. Model of the Autonomous Car

The dynamics of the autonomous vehicle is modeled by a standard bicycle model, see e.g. [14]. Additional equations are required to determine the position of the autonomous car.

$$\begin{aligned}
\dot{x}_1 &= s\cos(x_3 + x_5) \approx s\cos(x_3)\\
\dot{x}_2 &= s\sin(x_3 + x_5) \approx s\sin(x_3)\\
\dot{x}_3 &= x_4\\
\dot{x}_4 &= -\frac{c_1}{s}x_4 - c_2 x_5 + c_3 u\\
\dot{x}_5 &= \left[-1 - \frac{c_4}{s^2}\right] x_4 - \frac{c_5}{s}x_5 + \frac{c_6}{s}u
\end{aligned} \tag{1}$$

The position of the center of gravity of the autonomous car in road fixed coordinates are denoted by x_1 and x_2 respectively, see Fig. 2. The angle between the car fixed coordinate system (y_c- and x_c-axis) and the road fixed coordinate system (y- and x-axis) is given by x_3. The lateral car dynamics is described by the yaw rate x_4, and the angle x_5 between the longitudinal axis and the velocity vector of the center of gravity. The constants $c_1 - c_6$ are known car specific parameters and s is the speed of the car which is assumed to be constant. There is one input u describing the steering wheel angle. Note, that it is assumed that $|x_5|$ is much smaller than $|x_3|$ in order to allow simplification of the first two equations of the model as shown.

B. Model of the Other Car

The motion of the other car is only along the longitudinal axis and is modeled as a hybrid system. It contains the discrete modes *standstill*, *speed limit*, *brake* and *accelerate* as depicted in Fig. 3. The invariants of the discrete states are denoted by I and the transitions by t. The transition conditions are attached to the transition arrows. When a transition is taken, the continuous states are not reset. The continuous dynamics is described by the position z_1 and

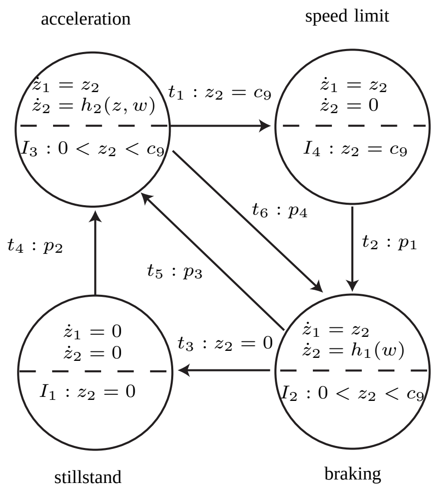


Fig. 3. Other car model

the velocity z_2. Note that the z_1-axis equals the x_2-axis of the autonomous car, and that the x value on the road is set constant for the other car (see Fig. 2). The brake dynamics $h_1(w)$ is linear while the acceleration dynamics $h_2(z, w)$ is nonlinear due to the dependence on the velocity:

$$\begin{aligned}
h_1(w) &= -c_7 w, \quad w \in [0, 1]\\
h_2(z, w) &= c_7\left(1 - \sqrt{\frac{z_2}{c_8}}\right)w, \quad w \in [0, 1]
\end{aligned} \tag{2}$$

Note that the input w of the brake and acceleration model is known only to be in the bounds $w \in [0, 1]$. The discrete dynamics of the hybrid model also contains uncertainty. Apart from the transitions t_1 and t_3, all other transitions are depending on probabilities p. It is believed that a stochastic driver model is best suited as usually only very few information is available about other traffic participants.

III. Abstraction to Linear Systems

The abstraction of the nonlinear equations in (1) and (2) to linear ones is performed for two reasons: one advantage is the simpler abstraction to Markov chains, the other the superposition of reachable sets which allows its separate computation of state and input-dependent parts. In order to abstract from nonlinear to linear dynamics, the rectangular discretization of a subset $X \subset \mathbb{R}^n$ of a state space is generally introduced:

Definition 1 (State space discretization): The discretization function $D : X \to \mathbb{I}$ assigns to each state $x \in X \subset \mathbb{R}^n$ an identifier $i \in \mathbb{I} \subset \mathbb{N}^+$ where $\mathbb{I}$ is the finite set of identifiers. The subset that is mapped to an identifier i is denoted by $X_i = \{x | D(x) = i\}$ and referred to as a cell. The state space is discretized rectangularly and equidistant so that all cells X_i are interval hulls with equal lengths: $X_i =]\underline{x}_i, \overline{x}_i]$, $\overline{x}_i - \underline{x}_i = c, \quad \underline{x}_i, \overline{x}_i, c \in \mathbb{R}^n$.

For simplicity, the abstraction is discussed for the autonomous car only, but performed for the other car, too. The dynamics of the autonomous car with additive disturbance vector v (e.g. wind, slope), input vector u, state vector x and initial value $x(0)$ is given as follows:

$$\dot{x} = f(x,u) + v,$$
$$\|u\|_\infty < \delta, \quad \|v\|_\infty < \mu, \quad x(0) \in X_i$$

The input and disturbance are bounded by sets and the initial state is bounded by a cell from definition 1. The abstraction to a linear system is performed by a first order Taylor expansion with remainder, resulting in a differential inclusion:

$$\begin{aligned} \dot{x} \in \quad & f(x_i^*, u^*) + \underbrace{\frac{\partial f(x,u)}{\partial x}\Big|_{x=x_i^*, u=u^*}}_{A_i} \Delta x \\ & + \underbrace{\frac{\partial f(x,u)}{\partial u})\Big|_{x=x_i^*, u=u^*}}_{B_i} \Delta u \oplus E_i + v \end{aligned} \tag{3}$$

with linearization points $x_i^* = mid(X_i)$, $u^* = 0$, where the operator $mid()$ returns the volumetric center of a set. The symbol $\oplus$ denotes the Minkowski sum[1] and E_i is the linearization error.

A. Computation of the Linearization Error

The linearization error E_i is conservatively approximated by the Lagrange remainder and the mean-value theorem. For the vector $z = \begin{bmatrix} x & u \end{bmatrix}^T$ and the linearization point $z_i^* = \begin{bmatrix} x_i^* & u^* \end{bmatrix}^T$, the error is given by:

$$\begin{aligned} E_{j,i} &= \frac{1}{2} \left(z - z_i^*\right)^T \frac{\partial^2 f_j(\xi)}{\partial z^2} \left(z - z_i^*\right), \\ z, \xi &= R_i([0,T]) \times \square(\delta) \end{aligned} \tag{4}$$

where $\square(\delta)$ is a hypercube of center 0 and of edge length 2δ. Let $R_i([0,T])$ denote the set of states x along trajectories starting within X_i and evolve for the time interval $t \in [0,T]$. The index i refers to the starting cell X_i and the index j to the j^{th} equation of f. The set $R_i([0,T])$ is formally defined over an auxiliary set $R_i(T)$:

Definition 2: $R_i(T)$ is an overapproximated set of the exact reachable set $R_i^e(T)$ that can be reached starting from X_i (for $t = 0$) at time $t = T$:

$R_i^e(T) = \{x | x(t)$ is solution of (3), for $t = T, x(0) \in X_i\}$

so that $R_i(T) \supset R_i^e(T)$.

Definition 3: $R_i([0,T])$ is the union of all overapproximated reachable sets $R_i(t)$ for $t \in [0,T]$:

$$R_i([0,T]) = \bigcup_{t\in[0,T]} R_i(t)$$

In order to obtain $E_{j,i}$ from (4), interval arithmetics [15] is used[2] as it guarantees the enclosure of the exact solution of $E_{j,i}$. Additionally, interval arithmetics is computationally cheap. The disadvantage is that one may obtain large overapproximations of $E_{j,i}$ [15]. For later computations, the linearization error is further abstracted by a hypercube so that $\forall j : E_{j,i} \in [-\epsilon_i, \epsilon_i]$. The abstracted linear system can finally be formulated as:

$$\begin{aligned} &\dot{x} \in A_i x + B_i u + f(x_i^*, u^*) \oplus \square(\epsilon_i) + v, \\ &\|u\|_\infty < \delta, \quad \|v\|_\infty < \mu, \quad x(0) \in X_i \end{aligned} \tag{5}$$

An alternative approach for the abstraction of nonlinear to linear systems based on the Lipschitz property of $f(x,u)$ has been published in [16].

B. Iteration and Convergence of the Error Estimation

A drawback of the presented abstraction method is that the reachable set $R_i([0,T])$ and the linearization error E_i are mutually dependent, so that E_i has to be found iteratively. It may also happen that the error estimation does not converge when the time horizon T is chosen too large. However, one can find a time step T so that the linearization procedure is converging:

Proposition 1: If $f(x,u)$ is Lipschitz, then

$$\exists T, \alpha : R_i([0,T]) \subset X_i \oplus \square(\alpha) \tag{6}$$

Proof: From the Lipschitz condition follows that $\|\dot{x}\|_\infty < \beta < \infty$ in (3). This leads to the following overapproximation: $R_i([0,T]) \subset X_i \oplus \square(\beta)T$. Consequently, one can choose $T < \frac{\alpha}{\beta}$. ■

IV. Reachable Set Computation

After abstracting the nonlinear model to a linear one, the reachable set can be composed of two types of reachable sets: a state-dependent and an input-dependent one. The state-dependent set is computed offline while the input-dependent one is computed online. A further distinction is made by computing the reachable set at time points and time intervals. This procedure is applied to improve the error propagation of the time interval solution when computing with Markov chains, as it will be shown in section V. In table I different types of reachable sets are defined. For example, $\hat{R}$ is defined as the reachable set of (5) with $x(0), v \neq 0$, but with $f(x^*, u^*), u = 0$. The reachable sets are finally computed by Minkowski addition of the following partial reachable sets:

	state-dependent (offline)	input-dep. (online)
$R(T) =$	$\hat{R}(T) \oplus \check{R}(T)$	$\oplus \bar{R}(T)$
$R([0,T]) =$	$\hat{R}([0,T]) \oplus \check{R}([0,T]) \oplus \bar{R}([0,T])$	

TABLE I
TYPES OF REACHABLE SETS

Reachable set	$x(0)$	$f(x^*, u^*)$	u	v
R	$\neq 0$	$\neq 0$	$\neq 0$	$\neq 0$
$\bar{R}$	$= 0$	$= 0$	$\neq 0$	$= 0$
$\hat{R}$	$\neq 0$	$= 0$	$= 0$	$\neq 0$
$\check{R}$	$= 0$	$\neq 0$	$= 0$	$= 0$

[1]$C \oplus D = \{c + d | c \in C, d \in D\}$

[2]used tool: b4m, Technical University of Hamburg-Harburg

A. State-dependent Solution

The sets $\hat{R}_i(T)$ and $\hat{R}_i([0,T])$ are computed as described in [7]. The geometry of these reachable sets is described by zonotopes:

Definition 4 (Zonotope, taken from [7]): A zonotope Z is a set such that:

$$Z = \left\{ x \in \mathbb{R}^n : x = c + \sum_{i=1}^{p} x_i g_i, \quad -1 \leq x_i \leq 1 \right\}$$

where $c, g_1, \ldots, g_p \in \mathbb{R}^n$. The order of a zonotope is $\frac{p}{n}$.

Analogously to $\hat{R}_i([0,T])$, $\bar{R}_i([0,T])$ is computed by regarding the input u as a disturbance with all possible values for u: $\|u\|_\infty < \delta$. The value $\check{R}_i(T)$ is well known to be computed as:

$$\begin{aligned}\check{R}_i(T) &= \int_0^T e^{A_i(t-\tau)}\, d\tau f(x_i^*, u^*) \\ &= A_i^{-1}(e^{A_iT} - I) f(x_i^*, u^*)\end{aligned}$$

where I is the identity matrix. The time interval solution $\check{R}_i([0,T])$ is approximated by a line from the origin to $\check{R}_i(T)$ so that $\check{R}_i([0,T]) \approx \frac{t}{T}\check{R}_i(T)$, $t \in [0,T]$. This line is bloated by a hypercube $\Box(\eta)$ that is obtained as described in [17]:

$$\check{R}_i([0,T]) = \frac{t}{T}\check{R}_i(T) + \Box(\eta_i)$$
$$\eta_i = \left\| A_i^{-1}\left[e^{A_iT} - I - A_iT - 0.375(A_iT)^2 \right] f(x_i^*, u^*) \right\|_\infty$$

B. Input-dependent Solution

The remaining task is to calculate the reachable set $\bar{R}_i(T)$ resulting from input u which is supposed to be generated by an online controller. As $\bar{R}_i(T)$ has to be computed online, the computational efficiency is increased by sampling u with a first order hold as illustrated in Fig. 4:

$$\begin{gathered} u(t) \in u(k) + \frac{u(k+1) - u(k)}{T}\Delta t + \Box(\gamma) \\ t \in [(k-1)T, kT], \quad \Delta t \in [0,T] \end{gathered}$$

The sampling error γ (see Fig. 4) is considered by the state-dependent solution. As the state-dependent solution is computed offline, an upper threshold $\overline{\gamma}$ for all expected input trajectories u is defined in advance. If an input trajectory u exceeds this limit, the trajectory is regarded as unsafe. The set that is added to the right side of (5) is obtained by $\Box(\|B_i\|_\infty \overline{\gamma})$. The improved computational efficiency due to the sampling results from the analytical computation of $\bar{R}_i(kT)$, which is given for $k \in \mathbb{R}^+$ by:

$$\begin{aligned}\bar{R}_i(kT) = \quad & A_i^{-1}(e^{A_iT} - I) B_i\, u(k) \\ & + A_i^{-2}(e^{A_iT} - I - A_iT) B_i \dot{u}(k)\end{aligned}$$

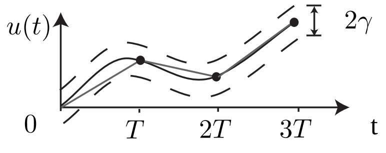


Fig. 4. Sampled input trajectory

The solution for the constant value $u(k)$ is analogous to the computation of $\check{R}_i(T)$. The solution for $\lambda\Delta t$ with $\lambda = \frac{u(k+1) - u(k)}{T}$ is obtained by

$$\begin{aligned}&\int_0^t e^{A(t-\tau)}\tau\, d\tau B\lambda \\ &= \left[tA^{-1}(e^{At} - I) - A^{-2}(e^{At}(At - I) + I) \right] B\lambda \\ &= A^{-2}(e^{At} - I - At) B\lambda\end{aligned}$$

C. Reachability of the Hybrid System

The reachable set of the hybrid system of the other car (see Fig. 3) is computed for each invariant region I as described in the two previous subsections. When the reachable set intersects an invariant, the computation is continued in each invariant for the intersection of the reachable set with the corresponding invariant. Note that the intersection of the reachable set with one of the invariants, does not result in zonotopes in general. For this reason, intersections are enclosed by zonotopes as it is done in [7].

D. Conversion from Zonotopes to Polytopes

The zonotopes of the reachable sets $R_i(T)$ and $R_i([0,T])$ are converted to a polytope representation in order to compute intersections with state space cells X_i as a prerequisite of the Markov chain generation. The order of these zonotopes is greater than one and can be controlled by a zonotope reduction technique in [7]. The conversion from zonotopes to V-polytopes[3] is computed by the convex hull of possible vertices of the zonotope:

$$h_k = c \pm g_1 .. \pm g_p, \quad k \in \{1, \ldots, 2^n\}$$

where h_k is a potential vertex, c the center and $g_1, \ldots, g_p$ are the generators of the zonotope. This computation results in 2^p vertex candidates so that the computational effort can be drastically reduced by setting the zonotope order to one which results in 2^n vertices. This drastic order reduction is performed by algorithm 1 that differs from the one in [7]. The algorithm is based on picking n generators of the Zonotope Z that span the reduced zonotope in a best way. The reduced zonotope encloses the original one by stretching the chosen generators. This is achieved by applying the box operator $box()$ which returns an interval hull that encloses

[3]V-polytope: convex polytope is represented by vertices

Algorithm 1 Reduce zonotope

```
Input: $Z = \left[c, G\right]$, $G = \left[g_1, \ldots, g_p\right]$
Output: $Z_{red}$ (reduced zonotope)

  $P = g_j : \quad \max(\|g_j^T G\|_1)$
  for $k = 1..n$ do
      $g^* \;=\; g_j : \quad \min(\|g_j^T P\|_1 / \|g_j\|_2^{\xi+1})$
      $P \;:=\; \left[P \quad \frac{1}{|g^*|} g^*\right]$
  end for
  $Z_{red} = P box(P^{-1} Z)$
```

the argument. The first generator is picked so that $\|G^T g_j\|_1$ is maximized which returns a long generator that is well aligned with other generators. The further generators are chosen according to:

$$g_j: \quad \min\left(\frac{1}{|g_j|^\xi}\sum_k |\cos(\alpha_{kj})|\right) = \min\left(\frac{\|g_j^T P\|_1}{\|g_j\|_2^{\xi+1}}\right)$$

where α_{kj} is the angle between the generators g_k and g_j. The parameter ξ weights the length of the generator g_k against its angle to g_j. A reduction example of a randomly generated zonotope can be seen in Fig. 5.

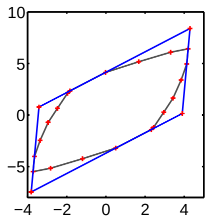


Fig. 5. Example for zonotope reduction

V. Markov Chains

The polytopes describing the reachable sets $R_i(T)$ and $R_i([0,T])$ are used to compute the transition probabilities of discrete time Markov chains which abstractly represent the continuous linear dynamics of (5). In this paper, probabilities are assigned to the fact that the system state is in a cell X_i at a particular time point or interval. Based on the assumption that the state of the system is equally like to be in any point of the reachable set, the transition probabilities of the state-dependent Markov chains are calculated similarly as in [11] by:

$$\Phi^*_{ij}(T) = \frac{V((\hat{R}_j(T) \oplus \check{R}_j(T)) \cap X_i)}{V(\hat{R}_j(T) \oplus \check{R}_j(T))}$$

$$\Phi_{ij}([0,T]) = \frac{V(R_j([0,T]) \cap X_i)}{V(R_j([0,T]))}$$

where $V()$ is an operator determining the volume of a polytope. The matrix $\Phi^*_{ij}(T)$ contains the probabilities that a state along the trajectory (without input u) starting in cell j lies in cell i after time T. $\Phi_{ij}([0,T])$ contains the probabilities within the time span $t = [0,T]$ with $\|u\|_\infty < \delta$. The state-dependent transition probabilities $\Phi^*_{ij}(T)$ and $\Phi_{ij}([0,T])$ can be computed offline. In contrast, the transition Matrix for the input-dependent solution is computed online from:

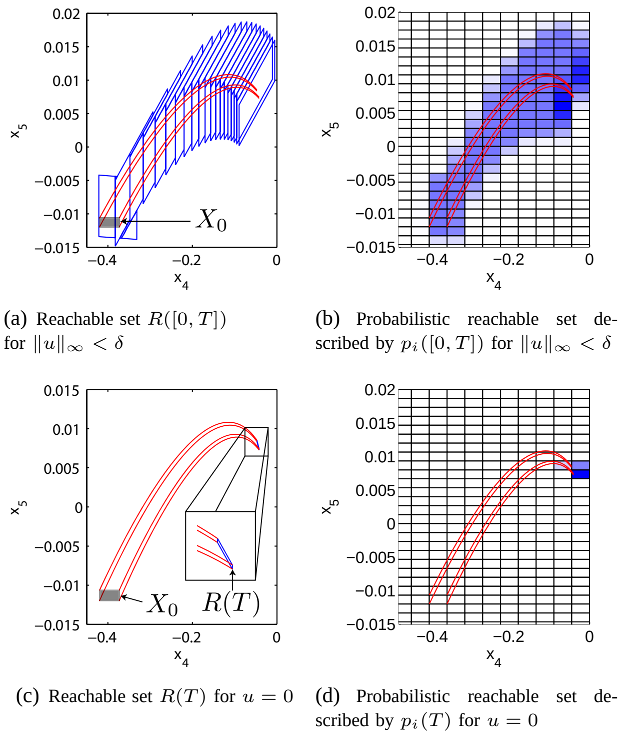
$$\bar{\Phi}_{ij}(T) = \frac{V((\bar{R}_j(T) \oplus X_j) \cap X_i)}{V(X_j)}$$

For linear systems, where $\bar{R}_j(T)$ is the same for all cells, the computation of the columns of $\bar{\Phi}_{ij}(T)$ has to be performed once and can be copied for the remaining columns. The probability p_i that the state is in cell X_i between the times kT and $(k+1)T$ is calculated based on the probability at time points kT:

$$p_i((k+1)T) = \Phi^*_{il}(T)\bar{\Phi}_{lj}(T)p_j(kT)$$

$$p_i([kT,(k+1)T]) = \Phi_{ij}([0,T])p_j(kT)$$

As $p_i([kT,(k+1)T])$ is computed based on $p_j(kT)$, overapproximations made by the computation of $p_i([kT,(k+1)T])$ are not propagating. An example of the transformation of the continuous reachable set to Markov chains is visualized in Fig. 6 for the variables x_4 and x_5 of the bicycle model (1). The continuous reachable sets on the left side as well as the probabilistic occupancy of the state space cells on the right side are given for the time point and time interval case. Additionally, sample trajectories starting from the vertices of the initial set are plotted in the figures.



(a) Reachable set $R([0,T])$ for $\|u\|_\infty < \delta$

(b) Probabilistic reachable set described by $p_i([0,T])$ for $\|u\|_\infty < \delta$

(c) Reachable set $R(T)$ for $u = 0$

(d) Probabilistic reachable set described by $p_i(T)$ for $u = 0$

Fig. 6. Reachable set of the bicycle model (1) described by zonotopes and cell probabilities with $x(0) \in X_{186}$

VI. Verification Results

This section shows results of the presented methods applied to the traffic scenario introduced in section II. Car parameters and initial conditions are given in the tables II and III. The variables $p_{acc}, p_{brake}, p_{sl}, p_{ss}$ refer to the probability that the discrete state is in one of the modes *acceleration* (acc), *brake* (brake), *speedlimit* (sl) or *stillstand* (ss). The state space is discretized as follows: the road is discretized by 50 segments for the x and y direction. The speed of the other car is discretized by 21 segments and all other states of the autonomous car are discretized by 30 segments. As the solution of the autonomous car and the other car are independent and the autonomous car dynamics

TABLE II

PARAMETER VALUES

autonomous car			other car		
c_1	160	$\frac{m}{s^2 \cdot rad}$	c_7	10	$\frac{m}{s^2}$
c_2	-1.6	$\frac{1}{s^2 \cdot rad}$	c_8	60	$\frac{m}{s}$
c_3	53	$\frac{1}{s^2 \cdot rad}$	c_9	15	$\frac{m}{s}$
c_4	-3.5	$\frac{m^2}{s^2 \cdot rad}$	p_1	0.5	–
c_5	156	$\frac{m}{s^2 \cdot rad}$	p_2	0.5	–
c_6	78	$\frac{m}{s^2 \cdot rad}$	p_3	0.5	–
s	15	$\frac{m}{s}$	p_4	0.5	–

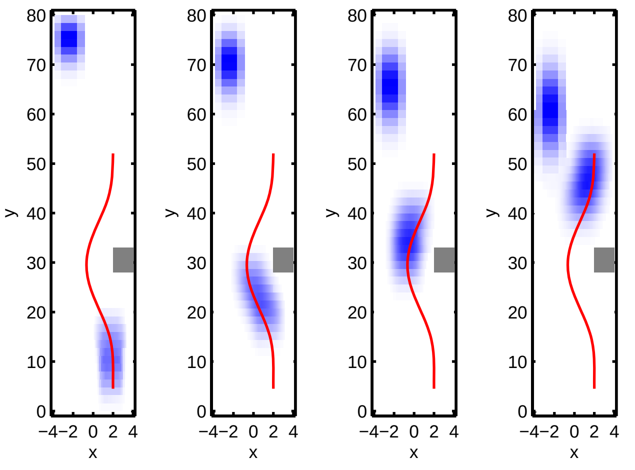
TABLE III

SET OF INITIAL CONDITIONS

autonomous car			other car		
x_1	$[3, 6]$	m	z_1	$[75, 80]$	m
x_2	$[1.8, 2.2]$	m	z_2	$[3, 8]$	m/s
x_3	$[-0.01, 0.01]$	rad	p_{acc}, p_{brake}	0.5	–
x_4	$[-0.01, 0.01]$	rad/s	p_{sl}, p_{ss}	0	–
x_5	$[-0.01, 0.01]$	rad			

is coupled in one direction ($x_{4,5} \longrightarrow x_3 \longrightarrow x_{1,2}$), one can build separate Markov chains for the system parts. Due to these separate Markov chains, the total number of states is 6900 and the execution time is 0.88 seconds for $t = [0, 3.2]$ seconds. The calculations have been performed with Matlab on a notebook dual core processor (1.66 GHz).

The resulting reachable sets of both cars are visualized in Fig. 7 for four time intervals. The car starting from the right lane is the autonomous car, the one starting from the left one is the other car. The full line shows a sample trajectory of the autonomous car for an initial value in the set of initial conditions. The box represents the static obstacle. Dark color of the discretized road cells indicates high probability and light color small probability that a car is located in a cell at the given time interval. Note that the probabilities refer to the presence of the whole car and not to its center of mass only (car length: 4m, car width: 2m).



(a) t=0-0.8 sec (b) t=0.8-1.6 sec (c) t=1.6-2.4 sec (d) t=2.4-3.2 sec

Fig. 7. Reachable sets of the traffic scenario

VII. CONCLUSIONS

The potential of hybrid verification in the safety assessment of cognitive cars has been demonstrated. The presented approach is general and can be applied to more complex traffic situations. It considers uncertainties as they naturally appear in traffic, and the presented approach is scalable: if the traffic situation is more complex, one can use Markov chains of lower resolution, i.e. less discrete states so that verification stays faster than real time.

ACKNOWLEDGMENTS

The authors gratefully acknowledge support of this work by the Deutsche Forschungsgemeinschaft (German Research Foundation) within the Transregional Collaborative Research Centre 28 Cognitive Automobiles.

REFERENCES

[1] A. van der Schaft and H. Schumacher, *An Introduction to Hybrid Dynamical Systems*. Springer, 2000.
[2] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*. MIT Press, 2000.
[3] C. Tomlin, I. Mitchell, A. Bayen, and M. Oishi, “Computational techniques for the verification and control of hybrid systems,” in *Proceedings of the IEEE*, vol. 91, 2003, pp. 986–1001.
[4] O. Botchkarev and S. Tripakis, “Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations,” in *Hybrid Systems - Computation and Control*, ser. LNCS 1790. Springer, 2000, pp. 73–88.
[5] A. Chutinan and B. H. Krogh, “Computational techniques for hybrid system verification,” in *IEEE Transactions on Automatic Control*, vol. 48, no. 1, 2003, pp. 64–75.
[6] O. Stursberg and B. H. Krogh, “Efficient representation and computation of reachable sets for hybrid systems,” in *Hybrid Systems - Computation and Control*, ser. LNCS 2623. Springer, 2003, pp. 482–497.
[7] A. Girard, “Reachability of uncertain linear systems using zonotopes,” in *Hybrid Systems : Computation and Control*, vol. 3414, 2005, pp. 291–305.
[8] O. Stursberg, A. Fehnker, Z. Han, and B. H. Krogh, “Verification of a cruise control system using counterexample-guided search,” *Control Engineering Practice*, vol. 12/10, pp. 1269–1278, 2004.
[9] J. Hu, J. Lygeros, M. Prandini, and S. Shankar, “A probabilistic framework for highway safety analysis,” in *Proceedings of the 38th Conference on Decision and Control*, 1999.
[10] K. Lee and H. Peng, “Evaluation of automotive forward collision warning and collision avoidance algorithms,” *Vehicle System Dynamics*, vol. 43, no. 10, pp. 735–751, 2005.
[11] J. Lunze and B. Nixdorf, “Representation of hybrid systems by means of stochastic automata,” *Mathematical and Computer Modeling of Dynamical Systems*, vol. 4, pp. 383–422, 2001.
[12] X. Koutsoukos and D. Riley, “Computational methods for reachability analysis of stochastic hybrid systems,” in *Hybrid Systems: Computation and Control*, 2006, pp. 377–391.
[13] M. Prandini and J. Hu, “A stochastic approximation method for reachability computations,” Final Report of the Hybridge Project, pp. 115–147, 2005.
[14] R. Rajamani, *Vehicle Dynamics and Control*. Springer, 2005.
[15] L. Jaulin, M. Kieffer, and O. Didrit, *Applied Interval Analysis*. Springer, 2006.
[16] E. Asarin, T. Dang, and A. Girard, “Reachability analysis of nonlinear systems using conservative approximation,” in *Hybrid Systems: Control and Computation*, 2003, pp. 20–35.
[17] T. Dang, “Vérification et synthèse des systèmes hybrides,” Ph.D. dissertation, Institut National Polytechnique de Grenoble, 2000.