2017 IEEE Intelligent Vehicles Symposium (IV)
June 11-14, 2017, Redondo Beach, CA, USA

SPOT: A Tool for Set-Based Prediction of Traffic Participants

Markus Koschi and Matthias Althoff

Abstract— **Predicting the movement of other traffic participants is an integral part in the motion planning of most automated road vehicles. While simple predictions, e.g. based on assuming constant velocity, may suffice for deciding a driving strategy, predicting the set of all possible behaviors is required to ensure safe motion plans. In this work, we propose a novel tool for the latter problem based on reachability analysis: Set-Based Prediction Of Traffic Participants (*SPOT*). Our tool can predict the future occupancy of other traffic participants, including all possible maneuvers (e.g. full acceleration, full braking, and arbitrary lane changes), by considering physical constraints and assuming that the traffic participants abide by the traffic rules. However, we remove assumptions for each traffic participant individually as soon as a violation of a traffic rule is detected. Removal of assumptions automatically results in larger occupancies and thus a smaller drivable area for the ego vehicle, ensuring that the ego vehicle does not cause a collision during the time horizon of the prediction. Experimental results show that we obtain the set of future occupancies within a fraction of the prediction horizon. Our tool is available at spot.in.tum.de.**

I. Introduction

While maneuvering on public roads, one must constantly anticipate possible behaviors of other traffic participants. For this reason, predicting the movement of other traffic participants has become an active research area in automated driving [1].

We briefly review existing prediction techniques and refer to [1] for a more extensive survey. Single future behaviors are predicted in e.g. [2], [3]. Since another traffic participant will most likely not follow the prediction exactly, such approaches cannot be used to ensure safe motion. This deficiency is mitigated by computing several possible future behaviors for each traffic participant; often, probabilities are assigned to each behavior obtained from Monte Carlo simulation [4], [5]. Instead of using Monte Carlo simulation, one can also predict the probability distribution of the occupancy as done in e.g. [6], [7].

None of the previously reviewed approaches can guarantee that all possible behaviors for given assumptions are computed. This, however, is required to ensure safety as described in [8] (referred to as zone model) or in [9]. Our proposed tool *SPOT* uses reachability analysis to predict the occupancy of surrounding traffic participants in a rigorous and set-based way. The obtained results are provably an over-approximation compared to the exact set of possible occupancies, given the assumptions made.

Markus Koschi and Matthias Althoff are with the Department of Informatics, Technical University of Munich, 85748 Garching, Germany. {markus.koschi, matthias.althoff}@tum.de

To the best of our knowledge, no tool for set-based prediction of other traffic participants exists. Furthermore, we have not found any other stand-alone tool for the prediction of other traffic participants. Simple prediction techniques, however, are often integrated in other software frameworks (see e.g. [10], [11]). This paper significantly extends our previous work on set-based prediction [12]:

- We present the open source tool *SPOT* to compute the future occupancy of multiple traffic participants on arbitrary road networks.
- We can additionally handle opposite driving directions and intersections.
- For the first time, we describe how we react to observed violations of assumptions, e.g. when a traffic participant performs an illegal lane change.
- We introduce a novel abstraction which considers that other traffic participants are prohibited from merging in front of the ego vehicle in a way which violates the safe distance. However, as mentioned above, we provide the necessary reactions if this happens.

The remainder of this paper is organized as follows: Sec. II presents the concept of our tool and its applications. In Sec. III, we define the models of the road network and traffic participants, which are required for our set-based prediction. Its implementation is described in Sec. IV, where we introduce the overall algorithm and the computation steps needed to obtain the future occupancies. Sec. V presents numerical examples of *SPOT* for different multi-lane roads with several traffic participants.

II. Conceptual Overview and Benefits

SPOT computes the future occupancy for consecutive time intervals as shown in Fig. 1. To ensure that we do not miss any possible behavior, we compute an over-approximation of the occupancy based on reachability analysis. Please note that it is impossible to compute the exact occupancy for the nonlinear dynamics of traffic participants [13]. In addition to uncertain future behaviors, we can also consider uncertain initial states bounded by sets, as described in detail later, to account for uncertainties in the perception of a traffic scene.

SPOT is designed for verifying motion plans of short time horizons, since due to the full consideration of uncertainties, the future occupancy of other traffic participants grows over time and thus limits the solution space for the ego vehicle. For this reason, we suggest performing trajectory planning for two time horizons in parallel as in [12]. Non-formal occupancy prediction techniques help to find long-term motion plans: One could predict a single behavior or probabilistic occupancies for example (see overtaking maneuver in upper

978-1-5090-4804-5/17/$31.00 ©2017 IEEE

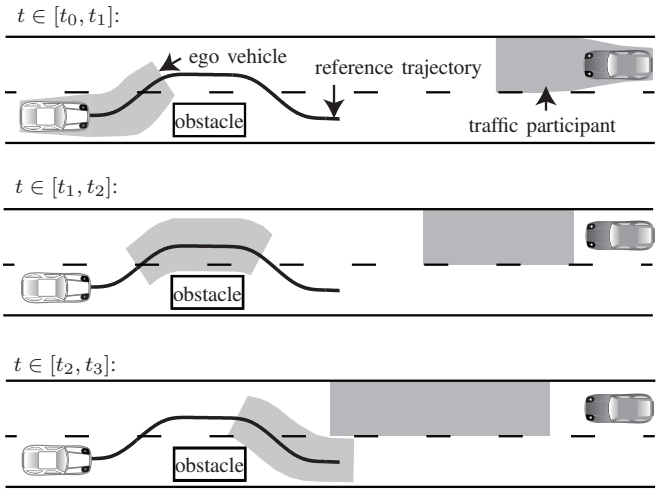


Fig. 1. Snapshots of the predicted occupancy of the traffic participant for selected consecutive time intervals.

part of Fig. 2). We suggest applying our formal prediction to the first part of the long-term plan only, as shown in the lower part of Fig. 2, since the short time horizon does not result in overly large occupancy sets. One should additionally plan a fail-safe maneuver into a safe state [14]. If we verify the first part of the intended trajectory as safe using our set-based prediction, this part can be executed while we try to verify the next part. Otherwise, the previously verified fail-safe trajectory is initiated. Thus, we can even guarantee that no collision occurs beyond the prediction horizon. To summarize, non-formal techniques provide long-term plans based on likely behaviors of other traffic participants, while set-based prediction guarantees safe maneuvers. For more details on integrating the verification into the framework of automated vehicles, the interested reader is refered to [12].

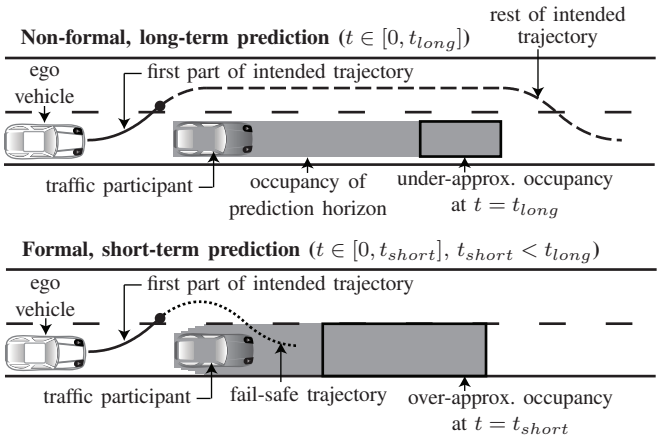


Fig. 2. Comparing non-formal, long-term prediction with formal, short-term prediction.

a) Application Areas: Based on the future occupancies of other traffic participants obtained by *SPOT*, the drivable area of the ego vehicle can be computed [15]. Such drivable areas can be used to prune the search space of the ego vehicle to more efficiently find feasible, safe motion plans. *SPOT* can also be used to verify planned motions [9]: If none of the computed occupancies intersects with the occupancy of the ego vehicle for all points in time, one can guarantee that the ego vehicle does not cause a collision (see Fig. 1).

b) Features: SPOT provides the following key features:

- The tool is open source so that parts of the code can be used for one's own purposes.
- Uncertainties in measurements (position, orientation, dimensions, velocity, and acceleration) and in the future behavior of traffic participants are explicitly considered.
- Even though *SPOT* is implemented in MATLAB (generally slower than compiled code), we achieve computation times within a fraction of the predicted horizon (typically less than $1.0\,\%$).
- *SPOT* has only a few lines of code (fewer than 5000). This makes understanding how it works, as well as extending its features, easy.
- *SPOT* is designed to be fed by different types of input. As a stand-alone tool, the user defines a traffic scenario in an XML file and specifies the time horizon for the prediction. It is also possible to embed *SPOT* in a motion planner and feed it with environment data. Given a planned trajectory, *SPOT* can check it for collisions with the predicted occupancies.

III. Models

In order to predict the future behavior of other traffic participants, the surrounding environment of the ego vehicle is modeled in a traffic scene, which consists of a road network, traffic participants, and static obstacles.

A. Road Network Model

The road network is modeled by *lanelets* [16], which are atomic, interconnected, and drivable road segments. They are defined by their *left* and *right bound*, where each bound is represented by an array of points (a polyline), as shown in Fig. 6. The driving direction of a lanelet is implicitly defined by its left and right bound. To consider all possible routes through the road network, we introduce relations between two lanelets: predecessor, successor, left, right. We further define *lanes* as the union of lanelets which are longitudinally adjacent, i.e. are successors and predecessors of each other. Note that a lanelet which has multiple successors, as in the case of road forks, becomes an element of multiple lanes (see Fig. 6).

B. Traffic Participant Model

In addition to the road network, the dynamics of traffic participants have to be modeled. Please note that in the current version of *SPOT*, we only consider vehicles; other traffic participants like cyclists and pedestrians are added in the future.

We typically do not have a precise model M_0 of other traffic participants (in contrast to the ego vehicle). Thus, we use different abstractions for the complicated dynamics described by differential inclusions

$$\dot{x} \in f_i\big(x(t), u(t)\big), \tag{1}$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the input, and the index i refers to the i^{th} model M_i. The possible initial states and the inputs are bounded by sets: $x(0) \in \mathcal{X}_0, \forall t : u(t) \in \mathcal{U}$. We denote a possible solution of (1) as $\chi_i(t, x(0), u(t))$. To define our abstractions, we require the definition of a reachable set, i.e. the set of states reachable at a certain point in time r from a set of initial states subject to uncertain inputs: $\mathcal{R}(M_i, r) = \left\{\chi_i(r, x(0), u(t)) \middle| x(0) \in \mathcal{X}_0, \forall t : u(t) \in \mathcal{U}\right\}$ (the notation $\mathcal{R}(M_i, r)$ neglects the dependence on $\mathcal{X}_0$ and $\mathcal{U}$ for brevity). Further, we introduce a mapping from the state space to the set of points in Cartesian space occupied by the traffic participant as $\mathrm{map}(x) : \mathcal{X} \to \mathcal{P}(\mathbb{R}^2)$, where $\mathcal{P}(\mathbb{R}^2)$ is the power set of $\mathbb{R}^2$. For a set of states $\mathcal{X}$, the mapping is defined as $\mathrm{map}(\mathcal{X}) := \{\mathrm{map}(x) | x \in \mathcal{X}\}$. Ultimately, we are interested in the over-approximative occupancy $\mathcal{O}(t)$ of a traffic participant, which is defined as

$$\forall t > 0 : \mathcal{O}(t) \supseteq \mathrm{map}\big(\mathcal{R}(M_0, t)\big). \tag{2}$$

Since reachability analysis of the original model M_0 is computationally expensive [12], our aim is to find abstractions M_j ($j = 1, \ldots, m$) such that $\forall t > 0 : \mathcal{R}(M_0, t) \subseteq \mathcal{R}(M_j, t)$. Thus, we can efficiently over-approximate the reachable set of the original model as

$$\forall t > 0 : \\ \mathrm{map}\big(\mathcal{R}(M_0, t)\big) \subseteq \bigcap_{j=1}^{m} \mathrm{map}\big(\mathcal{R}(M_j, t)\big) \subseteq \bigcap_{j=1}^{m} \mathcal{O}_j(t). \tag{3}$$

In particular, we have implemented three abstractions in *SPOT* so far: M_1, resulting in the *acceleration-based occupancy* $\mathcal{O}_1(t)$ (see Sec. IV-D); M_2, resulting in the *lane-following occupancy* $\mathcal{O}_2(t)$ (see Sec. IV-E); and M_3, resulting in the *safe distance occupancy* $\mathcal{O}_3(t)$ (see Sec. IV-F). The set of the overall occupancy of a traffic participant is

$$\mathcal{O}(t) = \mathcal{O}_1(t) \cap \mathcal{O}_2(t) \cap \mathcal{O}_3^{\complement}(t), \tag{4}$$

where $\mathcal{O}_3^{\complement}(t)$ denotes the complement of $\mathcal{O}_3(t)$. Since we compute $\mathcal{O}_3(t)$ as an under-approximation, i.e. $\mathcal{O}_3(t) \subseteq \mathrm{map}\big(\mathcal{R}(M_3, t)\big)$, $\mathcal{O}_3^{\complement}(t)$ is over-approximative and thus $\mathcal{O}(t)$ is over-approximative. As set representation for the occupancy, we choose polygons consisting of a tuple of vertices.

In addition to providing abstractions for the dynamics of traffic participants, we also consider a number of constraints listed in Tab. I: While $C_{a_{\max}}$ and C_{engine} are physical constraints, the other ones are a formalization of the Vienna Convention on Road Traffic [17]. The parameterized speed $v_{\max}$ is selected to be the minimum of three parameters: capable velocity of the obstacle, official speed limit of the obstacle's current lane times a speeding factor $f_S > 1$, and the critical velocity v_{critical} beyond which one cannot follow a lane anymore due to reaching maximum tire forces (computed based on [18]).

Due to measurement uncertainties, the possible set of initial states $\mathcal{X}_0$ is constructed by the Cartesian product of intervals, e.g. of position $[x_0, y_0]^T$ and of velocity $[v_{x,0}, v_{y,0}]^T$. Based on $\mathcal{X}_0$, each obstacle is characterized by a set of parameters, which are listed in Tab. II. The shape of a traffic participant is enclosed by a rectangle of length l and width w, where measurement uncertainties in terms of the obstacle's dimension are considered by enlarging its dimensions to $\tilde{l} < l$ and width $\tilde{w} < w$ as shown in Fig. 5. Note that l and w can vary when the initial state is updated, e.g. through updated sensor measurements.

TABLE I
OBSTACLE CONSTRAINTS.

Constraint	Variable	Description
$C_{a_{\max}}$	$a_{\max}$	Maximum absolute acceleration is limited by $a_{\max}$.
$C_{v_{\max}}$	$v_{\max}$	Positive longitudinal acceleration is stopped when a parameterized speed $v_{\max}$ is reached.
C_{engine}	v_S	Above a parameterized speed v_S, acceleration in driving direction is $a_{\mathrm{long}} = a_{\max}\frac{v_S}{v}$, which models limited engine power.
C_{back}	b_{back}	Driving backwards in a lane is not allowed.
C_{lane}	b_{lane}	Leaving the lane is forbidden. Changing lanes is only allowed if the new lane has the same driving direction as the previous one.
C_{safe}	–	Minimum distance to the ego vehicle ξ_{safe} must be kept to comply with safe distance regulations.

TABLE II
OBSTACLE PARAMETERS BASED ON THE CONSTRAINTS IN TAB. I.

Parameter	Variable	Parameter	Variable
max. acceleration	$a_{\max}$ in $\mathrm{m/s^2}$	Boolean for C_{back}	b_{back}
max. velocity	$v_{\max}$ in $\mathrm{m/s}$	Boolean for C_{lane}	b_{lane}
switching velocity	v_S in $\mathrm{m/s}$	length (incl. unc.)	l in m
speeding factor	f_S	width (incl. unc.)	w in m

IV. IMPLEMENTATION

This section describes the representation of traffic scenes, the overall structure of *SPOT*, and the abstractions currently implemented in *SPOT*.

A. Traffic Scenes

To represent traffic scenes with all the details required for set-based prediction, we use the *CommonRoad* XML data format [19] as specified in its documentation[1]. One can either download existing traffic scenes (from our website[2] or from the CommonRoad website[1]), or create new ones as described in our manual[2]. It is also possible to upload new scenarios through our website. Please note that XML files specifying traffic scenes are the only interface required for *SPOT* and can also be used to import perception information from a real vehicle.

[1] commonroad.in.tum.de
[2] spot.in.tum.de

B. Class Structure

The architecture of *SPOT* is presented in Fig. 3 using the class diagram of UML[3]. We emulate the perception of the ego vehicle (class `Perception`) to consider sensor range limitations among others. Our model of the traffic scene is stored as a map (class `Map`) and contains lanes (class `Lane`), obstacles (class `Obstacle`), and the ego vehicle itself (class `Vehicle`). A hierarchical class structure behind the superclass `Obstacle` allows us to distinguish between static and dynamic obstacles (class `StaticObstacle` and `DynamicObstacle`) and to represent different types of traffic participants, like passenger cars, trucks, and bicycles. To combine these different types, we use the terms *obstacle* and *traffic participant* interchangeably. Each obstacle has a property of the class `Occupancy`, which is computed as described in the next section. For further details on our class structure, please see our manual.

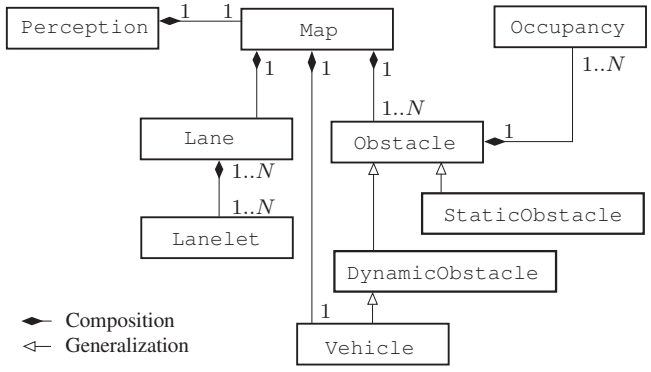


Fig. 3. Unified Modeling Language (UML) class diagram of *SPOT*.

C. Overall Algorithm

Alg. 1 shows the overall algorithm of *SPOT*, which predicts the occupancy for a traffic participant up to the prediction horizon and can run in parallel for each traffic participant. First, the constraint management configures the parameters according to the set of last-measured states of the obstacle (line 1) as detailed in the next paragraph. Second, the reachable lanes are extracted from the road network with respect to C_{lane} (line 2) as presented in the subsequent paragraph *Reachable Lanes*. Next, the occupancies of all abstractions are computed (line 4 and Sec. IV-D to IV-F). Finally, the overall occupancy of an obstacle is returned as the intersection of all occupancies according to (4) (line 6). Please note that we store a separate occupancy polygon for each traffic participant, time interval, and lane to make an efficient collision detection possible.

a) Managing Constraints: As mentioned before, we immediately adapt our model when constraints are violated. During the prediction, the method `manageConstraints` constantly checks for violations of the constraints based on the set of last-measured states, i.e. initial states $\mathcal{X}_0$, and previously-measured states $\mathcal{X}_{-1}$ of every obstacle. As soon as we detect violations of traffic rules, we individually adjust the obstacle's parameters according to Tab. III. We briefly motivate our actions for each constraint: If a traffic participant is driving faster than the speed limit by more than the speeding factor f_S, we increase the latter to ensure an over-approximative occupancy. When constraint C_{engine} is violated, our classification of the engine limits has been incorrect, and thus we remove this constraint. As soon as we detect that a traffic participant is driving backwards, its occupancy for negative speeds is included in our prediction. Likewise, we anticipate an illegal lane change of an obstacle in the set of its reachable lanes (which are described next). Note that $C_{a_{\max}}$ cannot be violated, since it is a physical constraint. Constraint C_{safe} must also not be checked for violations as explained later.

[3]uml.org

Algorithm 1 Occupancy Prediction for an Obstacle

Require: map
1: parameters ← MANAGECONSTRAINTS
2: reachableLanes ← REACH(map, parameters)
3: **for all** VALIDABSTRACTIONS(parameters) **do**
4: $\quad\mathcal{O}_j$ ← OCCUPANCY$_j$(reachableLanes, parameters)
5: **end for**
6: **return** $\mathcal{O}$ ← INTERSECT($\mathcal{O}_j$)

b) Reachable Lanes: To consider all possible routes through the road network, the method `reach` searches for all reachable lanes according to constraint C_{lane}. As mentioned before, a lane is the union of longitudinally adjacent lanelets. Additionally, we define the current lanes of an obstacle as all lanes in which the obstacle is currently positioned (e.g. in Fig. 6, lane_2 and lane_3 are the current lanes of the traffic participant) and introduce $\text{currentLane}(\text{lane})$ as a predicate which evaluates to true if the obstacle is positioned in the given lane. The set of reachable lanes $\mathcal{O}_{\text{road}}$ of a traffic participant is defined as follows, where b_{lane} is taken from Tab. II, and $\text{drivingDir}(\text{lane})$ returns the driving direction of the given lane:

$$
\begin{aligned}
b_{\text{lane}} : \mathcal{O}_{\text{road}} \in \big\{\text{lane}_i \big| \forall j :\ & \\
& \text{currentLane}(\text{lane}_j) \wedge \big(\text{lane}_i = \text{lane}_j \vee \\
& \text{lane}_i = \text{left}(\text{lane}_j) \vee \text{lane}_i = \text{right}(\text{lane}_j)\big) \wedge \\
& \text{drivingDir}(\text{lane}_i) = \text{drivingDir}(\text{lane}_j)\big\} \\
\neg b_{\text{lane}} : \mathcal{O}_{\text{road}} \in \big\{\text{lane}_i \big| \forall j :\ & \\
& \text{currentLane}(\text{lane}_j) \wedge \big(\text{lane}_i = \text{lane}_j \vee \\
& \text{lane}_i = \text{left}(\text{lane}_j) \vee \text{lane}_i = \text{right}(\text{lane}_j)\big)\big\}
\end{aligned}
$$

D. Acceleration-Based Occupancy (Abstraction M_1)

Abstraction M_1 considers the limits of the absolute acceleration ($C_{a_{\max}}$) of traffic participants while ignoring other constraints. As a result, the occupancy for a reference point of a traffic participant (see Fig. 5) at time t can be exactly computed by a circle with center $c(t)$ and radius $r(t)$ [20]:

$$
c(t) = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \begin{bmatrix} v_{x,0} \\ v_{y,0} \end{bmatrix} t, \quad r(t) = \frac{1}{2} a_{\max} t^2.
$$

TABLE III
CONSTRAINT MANAGEMENT.

Constraint	Check constraint condition	If constraint condition is violated, do
$C_{v_{\max}}$	$v_0 > \text{speedLimit} \cdot f_S$	$f_S = f_S + \frac{v_0}{\text{speedLimit}}$
C_{engine}	$a_0 > a_{\max}\frac{v_S}{v_0} \wedge v_S < v_0 < v_{\max}$	$v_S \to \infty$
C_{back}	$v_0 < 0 \wedge b_{\text{back}}$	$b_{\text{back}} = \text{false}$
C_{lane}	$\text{drivingDir}(\text{lane}(\mathcal{X}_0)) \neq \text{drivingDir}(\text{lane}(\mathcal{X}_{-1}))$	$b_{\text{lane}} = \text{false}$

The occupancy for a given time interval $\tau_k = [t_k, t_{k+1}]$ is bounded by two circles at t_k and t_{k+1} and a concave bound as derived in [9]. We can over-approximate this occupancy with a convex polygon $Q(q_1, \ldots, q_6)$ as shown in Fig. 4. The coordinates of the vertices q_i are given in [12].

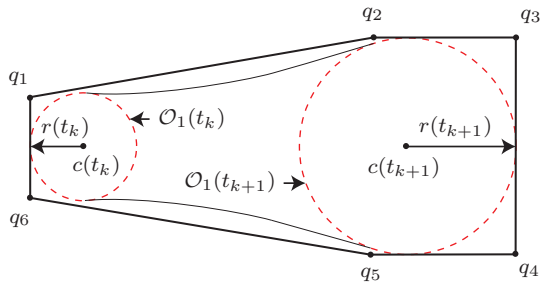


Fig. 4. Polygon $Q(q_1, \ldots, q_6)$ encloses the occupancy for a reference point of a traffic participant for the time interval $\tau_k = [t_k, t_{k+1}]$.

Next, the dimensions of the traffic participant (including uncertainties) are added as shown in Fig. 5. Polygon Q, representing the occupancy for the reference point, is enlarged by half of l and w in each direction to obtain polygon P, which is the over-approximative occupancy $\mathcal{O}_1(\tau_k)$.

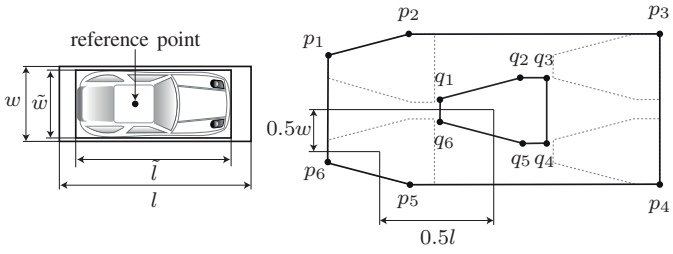


Fig. 5. Polygon $P(p_1, \ldots, p_6)$ is the occupancy $\mathcal{O}_1(\tau_k)$ of an vehicle. The vertices $q_1, \ldots, q_6$ are taken from Fig. 4.

E. Lane-Following Occupancy (Abstraction M_2)

In the previous subsection on abstraction M_1, we have only considered maximum absolute acceleration. However, as constraints $C_{v_{\max}}$ and C_{engine} describe, a traffic participant cannot always accelerate with a_{max} in driving direction, but is restricted by maximum velocity and maximum engine power. A traffic participant also cannot drive backwards unless for a parking maneuver, which forbids movement in negative driving direction (C_{back}). Thus, abstraction M_2 constrains the movement of the traffic participant in driving direction when following lanes, while assuming that any movement in lateral direction is allowed, i.e. perpendicular to the driving direction. Since motion is limited to the reachable lanes of the road network (C_{lane}), the occupancy is only extended to all laterally adjacent lanes which are part of $\mathcal{O}_{\text{road}}$. To sum up, abstraction M_2 fully considers $C_{v_{\max}}$, C_{engine}, C_{back}, and C_{lane}, while $C_{a_{\max}}$ is only considered in driving direction.

In order to efficiently enforce these constraints, we require the shortest path through the road network to obtain an over-approximation of the reachable set. Since this is a time-consuming optimization problem, we adopt the solution in [12]. As shown in Fig. 6, the shortest path is obtained by following inner lane bounds, while changing the side of the lane instantaneously at inflection points. Clearly, this is not possible in reality; however, this does not result in too conservative over-approximations in practice. The reachable occupancy $\mathcal{O}_2(t)$ is computed by moving along inner lane bounds and assuming that everything perpendicular to them is reachable. Uncertainties in the initial velocity and acceleration are considered by choosing the respective maximum value in driving direction, which ensures an over-approximation as proven in [9]. The details for computing $\mathcal{O}_2(t)$ are presented in [12].

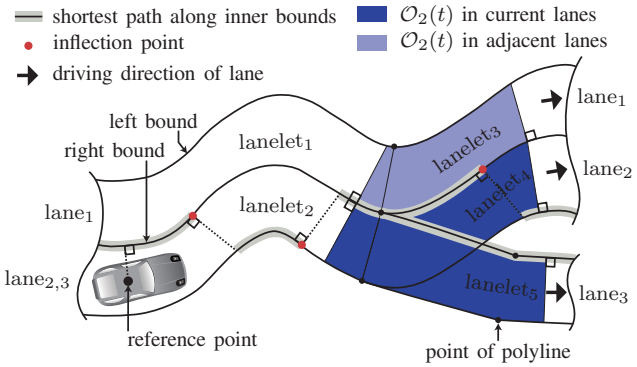


Fig. 6. Occupancy $\mathcal{O}_2(t)$ is obtained for all reachable lanes $\mathcal{O}_{\text{road}}$.

F. Safe Distance Occupancy (Abstraction M_3)

Intersecting $\mathcal{O}_1(t)$ with $\mathcal{O}_2(t)$ results in an over-approximative occupancy, which can be used by the ego vehicle to safely plan its future motion. However, our prediction can block space in front of the ego vehicle since we allow other traffic participants to change lanes. For example, imagine a scenario as depicted in Fig. 9. When the future occupancy of an obstacle in an adjacent lane reaches

in front of the ego vehicle, an intended trajectory cannot be verified as safe, even though traffic regulations forbid such a maneuver: The Vienna Convention on Road Traffic demands sufficient distance between two successive vehicles [17, 13§5], and the German traffic law, *Straßenverkehrs-Ordnung* (StVO), states that one shall not endanger any following traffic participants when changing lanes [21, 5§4]. Consequently, one has to keep a sufficient distance from the preceding and following traffic participants in order to abide by the traffic rules. We denote this distance by *safe distance* and consider it under constraint C_{safe}. In the following, we explain this novel abstraction in more detail.

Assuming equal maximum deceleration among traffic participants, the safe distance can be computed according to [22] as

$$\xi_{\text{safe}} = \frac{1}{2a_{\max}} \left(v_{f,0}^2 - v_{p,0}^2 \right) + v_{f,0}\delta, \tag{5}$$

where $v_{f,0}$ and $v_{p,0}$ are the initial velocities of the following and preceding vehicles, respectively, and δ is the reaction delay, i.e. the time between the preceding vehicle's full brake at time zero and the following vehicle's full braking. To under-approximate the safe distance, we use $\delta = 0.3\,\text{s}$, which is an assumption of [22] for automated vehicles and much shorter than the reaction time of human drivers of approximately $\delta = 1.0\,\text{s}$ [23, Fig. 1].

Abstraction M_3 is applied to traffic participants which are in laterally adjacent lanes to the ego vehicle. Since we are interested in the safe distance in front and in rear of the ego vehicle, (5) is evaluated at each time step for two cases: The traffic participant is considered in one case to be the preceding vehicle (resulting in $\xi_{\text{safe,front}}$) and in the other case to be the following vehicle (resulting in $\xi_{\text{safe,rear}}$), as shown in Fig. 7. After taking the minimum distance and constructing a polygon perpendicular to the corresponding lane bounds analogously to M_2 (see [12]), we obtain the occupancy $\mathcal{O}_3(\tau_k)$, which is an under-approximation of the legal safe distance.

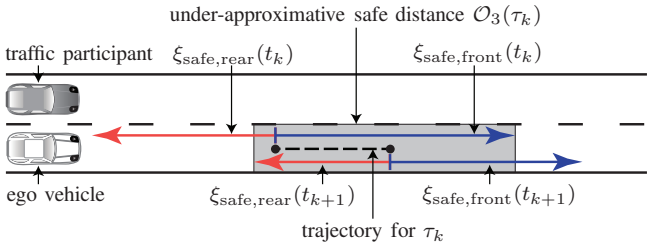


Fig. 7. Front and rear safe distances ξ_{safe} of the ego vehicle at t_k and t_{k+1} with respect to the traffic participant result in $\mathcal{O}_3(\tau_k)$.

As mentioned in Sec. IV-C, constraint C_{safe} does not have to be considered in the constraint management. Due to C_{safe}, we do not predict lane change maneuvers which do not comply with the legal safe distance. However, when a traffic participant initiates a lane change, we consider its occupancy in its new lane regardless of safe distance regulations, since abstraction M_3 restricts the occupancy only on laterally adjacent lanes. If the assumption on the safe distance is violated (i.e. another traffic participant does not abide by the traffic rules) and the ego vehicle is not able to find a evasive trajectory, a resulting collision is the sole responsibility of the other traffic participant.

To sum up, abstraction M_3 enables the ego vehicle to verify the part of its trajectory where other traffic participants are not allowed due to safe distance regulations as safe. If obstacles cut in nevertheless, their occupancy is automatically predicted in their new lane, and the ego vehicle can react by trying to regain a sufficient distance.

V. Numerical Examples

We demonstrate the features of *SPOT* on two different multi-lane road networks. Both scenarios are taken from the CommonRoad benchmarks[4] [19] (each unique ID is mentioned later) and are based on real roads. We use a time step size of $\Delta t = t_{k+1} - t_k = 0.5\,\text{s}$ and a prediction horizon of $t_f = 3\,\text{s}$. All obstacles are assigned the parameters listed in Tab. IV, in which we have obtained $a_{\max}$ by choosing a friction coefficient of $\mu = 0.82$ for a dry, good road and a gravity constant of $g = 9.81\,\text{m/s}^2$ [24, Fig. 3.3].

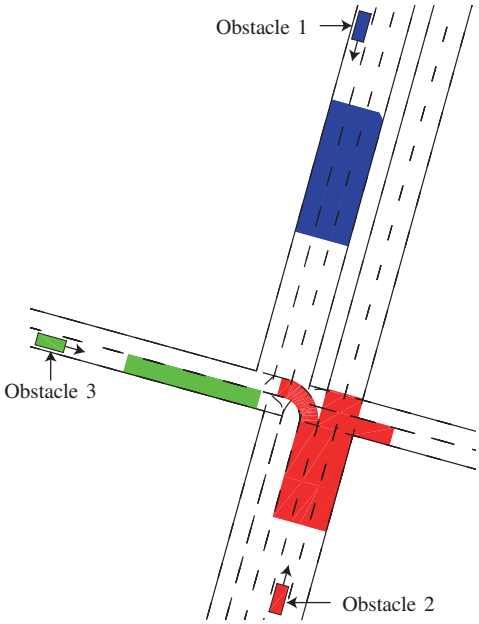
TABLE IV
Obstacle configuration for Scenario I and II.

Parameter	Value	Parameter	Value
$a_{\max}$	$8.0\,\text{m/s}^2$	a_0	$0\,\text{m/s}^2$
$v_{\max}$	$50.0\,\text{m/s}$	$v_{0,\text{Scenario I}}$	$13.89\,\text{m/s}$
v_S	$15.0\,\text{m/s}$	$v_{0,\text{ego vehicle}}$	$33.0\,\text{m/s}$
b_{back}	true	$v_{0,\text{Obstacle 4}}$	$35.0\,\text{m/s}$
b_{lane}	true	$v_{0,\text{Obstacle 5}}$	$27.0\,\text{m/s}$
l	$4.8\,\text{m}$	f_S	1.2
w	$2.0\,\text{m}$		

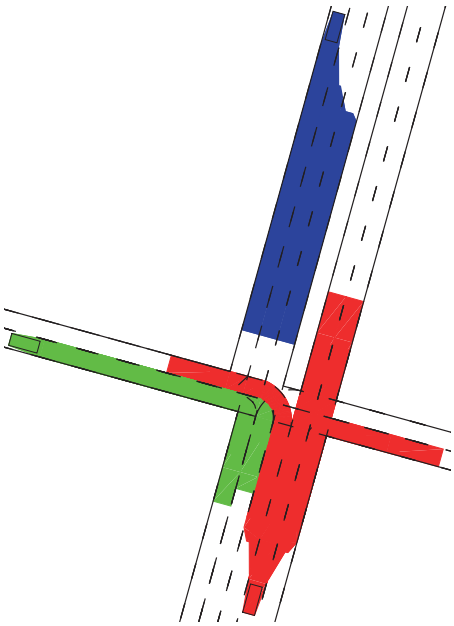
A. Intersection (Scenario I)

At road intersections, occupancy prediction is not only particularly important but also challenging. Scenario I presents an intersection in Munich's inner city (CommonRoad ID: S=GER_MUC_3a): The north-south street Leopoldstraße (5 lanes) is crossed by Hohenzollernstraße (2 lanes) and Nikolaistraße (2 lanes), which is modeled as an uncontrolled intersection. Fig. 8(a) shows the initial configuration at t_0 with Obstacles 1-3, which are all subject to the official speed limit of $50.0\,\text{km/h} \approx 13.89\,\text{m/s}$. While Obstacle 1 (blue) is driving south and can maneuver to the two left adjacent lanes, Obstacle 2 (red) is heading north with the possibility of continuing to one of the two straight lanes or taking a right or left turn. Note the median strip on Leopoldstraße where driving is not allowed. Obstacle 3 (green) is located on Hohenzollernstraße and can take a right turn. While Fig. 8(a) illustrates the occupancies $\mathcal{O}(t)$ for an intermediate time interval $t \in [t_3, t_4] = [1.5\,\text{s}, 2.0\,\text{s}]$, Fig. 8(b) shows $\mathcal{O}(t)$ for the entire prediction horizon $t \in [t_0, t_f]$.

[4]commonroad.in.tum.de



(a) Initial configuration and $\mathcal{O}(t)$ for $t \in [t_3, t_4]$.



(b) $\mathcal{O}(t)$ for the whole time horizon $t \in [t_0, t_f]$.

Fig. 8. Occupancies of Obstacles 1, 2, and 3 in Scenario I. The plot shows the initial configuration at t_0 and the predicted occupancies $\mathcal{O}(t)$ for selected time intervals.

The computation times of *SPOT* for predicting the occupancies in Scenario I for the whole time horizon are presented in Tab. V. They have been obtained using MATLAB 2016a on a machine with a 2.6 GHz Intel Core i7 processor with 20 GB 1600 MHz DDR3 memory and without using parallelization. It can be seen that the computation only requires a fraction of the prediction horizon. *SPOT* can also parallelize the independent prediction of each obstacle on a machine with a multi-core processor and thus compute the future occupancy of many surrounding obstacles in similar time. Since the computation time was below 0.05 s for all tested scenarios, the required time seems to be fairly independent of the traffic scene.

TABLE V
COMPUTATION TIMES FOR SCENARIO I.

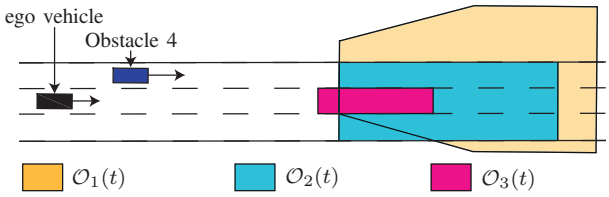
	Computation time	**Prediction horizon**	**Fraction of the prediction horizon**
Obstacle 1	0.017 s	3 s	0.56 %
Obstacle 2	0.025 s	3 s	0.83 %
Obstacle 3	0.012 s	3 s	0.39 %

B. Multi-Lane Highway (Scenario II)

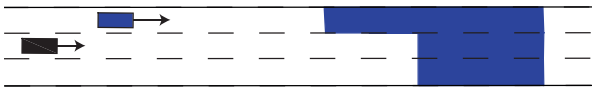
Scenario II features a three lane highway, where the ego vehicle is located in the middle lane (CommonRoad ID: S=GER_A9_2a). As described in Tab. IV, Obstacle 4 is driving faster than the ego vehicle (see Fig. 9), while Obstacle 5 is slower (see Fig. 10). In this scenario, a speed limit does not exist, which is common for a German Autobahn. To highlight the effect of abstraction M_3 (safe distance occupancy), the reaction time is increased to $\delta = 2.0\,\mathrm{s}$. Please note that for the sake of clarity, we plot the predicted occupancies for $t \in [t_2, t_3] = [1.0\,\mathrm{s}, 1.5\,\mathrm{s}]$ only and separately for each obstacle.

In Fig. 9(a), the three occupancies $\mathcal{O}_1(t)$, $\mathcal{O}_2(t)$, and $\mathcal{O}_3(t)$ of Obstacle 4 are drawn separately. It can been seen that $\mathcal{O}_2(t)$ is smaller than $\mathcal{O}_1(t)$, since motion in driving direction is restricted more in abstraction M_2. By combining the three occupancies, we obtain $\mathcal{O}(t)$, which is shown in Fig. 9(b) and considers that Obstacle 4 must keep a sufficient distance from the ego vehicle. Please note that the occupancy in the right-most lane has automatically been shortened by extending $\mathcal{O}_3(t)$ to the right lane, since Obstacle 4 cannot drive backwards.

The predicted occupancies for Obstacle 5 are plotted in Fig. 10. Since Obstacle 5 is slower than the ego vehicle, it can only change lanes after the ego vehicle has passed.



(a) Separate occupancies $\mathcal{O}_1(t)$, $\mathcal{O}_2(t)$, and $\mathcal{O}_3(t)$.



(b) Combined occupancy $\mathcal{O}(t) = \mathcal{O}_1(t) \cap \mathcal{O}_2(t) \cap \mathcal{O}_3^{\complement}(t)$.

Fig. 9. Occupancy of Obstacle 4 in Scenario II. The plot shows the initial configuration at t_0 and the predicted occupancies $\mathcal{O}(t)$ for $t \in [t_2, t_3]$.

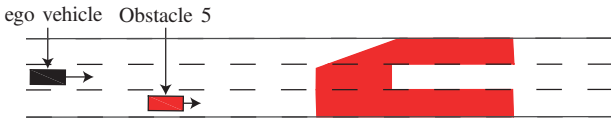


Fig. 10. Occupancy of Obstacle 5 in Scenario II. The plot shows the initial configuration at t_0 and the predicted occupancies $\mathcal{O}(t)$ for $t \in [t_2, t_3]$.

VI. Conclusion and Future Work

We present the first tool for set-based prediction of traffic participants, which is available as open source software at spot.in.tum.de and can be easily adapted to one's own needs. Using reachability analysis, we compute the set of future occupancies of each surrounding traffic participant on arbitrary road networks. These traffic scenes can be specified in XML files, for which we provide a set of examples on our website. Several applications can benefit from our tool, where most importantly, *SPOT* can be used to verify intended trajectories, since our approach is inherently safe. We have introduced six constraints for obtaining tight over-approximations, but adapt them individually as soon as one is violated.

In particular, most of our constraints are derived from traffic rules. For improved prediction, we wish to consider more regulations, e.g. at intersections, but applicable traffic rules have not yet been thoroughly formalized [25]. Future research also includes sensor limits and interaction between traffic participants. We additionally plan to integrate our software in the *Robot Operating System* (ROS) [26] making it possible to test *SPOT* in real vehicles.

Acknowledgment

The authors thank Mona Beikirch for her valuable contribution to abstraction M_3 and gratefully acknowledge partial financial support by the BMW Group within the CAR@TUM project and by the Deutsche Forschungsgemeinschaft (German Research Foundation) within the Priority Programme SPP 1835 Cooperative Interacting Automobiles (grant number: AL 1185/4-1).

References

[1] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *ROBOMECH Journal*, vol. 1, no. 1, pp. 1–14, 2014.

[2] A. Barth and U. Franke, "Where will the oncoming vehicle be the next second?" in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2008, pp. 1068–1073.

[3] A. Eidehall, "Multi-target threat assessment for automotive applications," in *Proc. of the 14th Int. IEEE Conference on Intelligent Transportation Systems*, 2011, pp. 433–438.

[4] A. Eidehall and L. Petersson, "Statistical threat assessment for general road scenes using Monte Carlo sampling," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, pp. 137–147, 2008.

[5] A. E. Broadhurst, S. Baker, and T. Kanade, "Monte Carlo road safety reasoning," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2005, pp. 319–324.

[6] T. Gindele, S. Brechtel, and R. Dillmann, "Learning driver behavior models from traffic observations for decision making and planning," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 1, pp. 69–79, 2015.

[7] M. Althoff, O. Stursberg, and M. Buss, "Model-based probabilistic collision detection in autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 2, pp. 299–310, 2009.

[8] B. Vanholme, D. Gruyer, B. Lusetti, S. Glaser, and S. Mammar, "Highly automated driving on highways based on legal safety," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 333–347, 2013.

[9] 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.

[10] J. Wei, J. M. Snider, J. Kim, J. M. Dolan, R. Rajkumar, and B. Litkouhi, "Towards a viable autonomous driving research platform," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2013, pp. 763–770.

[11] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, E. Kaus, R. G.Herrtwich, C. Rabe, D. Pfeiffer, F. Lindner, F. Stein, F. Erbs, M. Enzweiler, C. Knöppel, J. Hipp, M. Haueis, M. Trepte, C. Brenk, A. Tamke, M. Ghanaat, M. Braun, A. Joos, H. Fritz, H. Mock, M. Hein, and E. Zeeb, "Making Bertha drive – an autonomous journey on a historic route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, 2014.

[12] M. Althoff and S. Magdici, "Set-based prediction of traffic participants on arbitrary road networks," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 2, pp. 187–202, 2016.

[13] M. Althoff, "Reachability analysis and its application to the safety assessment of autonomous cars," Dissertation, Technische Universität München, 2010, http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20100715-963752-1-4.

[14] S. Magdici and M. Althoff, "Fail-safe motion planning of autonomous vehicles," in *Proc. of the 19th IEEE International Conference on Intelligent Transportation Systems*, 2016, pp. 452–458.

[15] S. Söntges and M. Althoff, "Determining the nonexistence of evasive trajectories for collision avoidance systems," in *Proc. of the 18th IEEE International Conference on Intelligent Transportation Systems*, 2015, pp. 956–961.

[16] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2014, pp. 420–425.

[17] United Nations Economic Commission for Europe, "Vienna convention on road traffic," United Nations, 1968.

[18] E. Velenis and P. Tsiotras, "Optimal velocity profile generation for given acceleration limits: theoretical analysis," in *Proc. of the American Control Conference*, 2005, pp. 1478–1483.

[19] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017.

[20] C. Schmidt, F. Oechsle, and W. Branz, "Research on trajectory planning in emergency situations with multiple objects," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2006, pp. 988–992.

[21] Bundesministerium für Verkehr und digitale Infrastruktur, "Straßenverkehrs-Ordnung," Bundesrepublik Deutschland, 2013.

[22] M. Althoff and R. Lösch, "Can automated road vehicles harmonize with traffic flow while guaranteeing a safe distance?" in *Proc. of the 19th International IEEE Conference on Intelligent Transportation Systems*, 2016, pp. 485–491.

[23] G. Johansson and K. Rumar, "Drivers brake reaction times," *Human Factors*, vol. 13, no. 1, pp. 23–27, 1971.

[24] C.-G. Wallman and H. Åström, "Friction measurement methods and the correlation between road friction and traffic safety," in *VTI meddelande*. Swedish National Road and Transport Research Institute, 2001.

[25] A. Rizaldi and M. Althoff, "Formalising traffic rules for accountability of autonomous vehicles," in *Proc. of the 18th IEEE International Conference on Intelligent Transportation Systems*, 2015, pp. 1658–1665.

[26] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.