Formal Falsification Criteria as a Basis for Behavior Planning based on Reinforcement Learning Algorithms

Master Thesis No. 792/20

Editor: Robert Alexander Hülsmann, B. Eng. | 2741778
Supervisors: Felix Glatzki, M. Sc.
Moritz Lippert, M. Sc.

TECHNISCHE UNIVERSITÄT DARMSTADT

FZD FAHRZEUGTECHNIK TU DARMSTADT

Robert Alexander Hülsmann, B. Eng.
Matriculation No.: 2741778
Study program: Autonome Systeme

Master Thesis No. 792/20
Topic: Formal Falsification Criteria as a Basis for Behavior Planning based on Reinforcement Learning Algorithms

Submitted: May 6, 2021

Technische Universität Darmstadt
Fachgebiet Fahrzeugtechnik
Prof. Dr. rer. nat. Hermann Winner
Otto-Berndt-Straße 2
64287 Darmstadt

Masterthesis Nr. 792/20 im Studiengang Autonome Systeme (30 CP)

von Robert Hülsmann

Voraussichtlicher Beginn: 19.10.2020
Bearbeitungsdauer: 6 Monate

Thema: **Formale Falsifikationskriterien als Grundlage für die Verhaltensplanung basierend auf Reinforcement Learning Algorithmen**

Topic: ***Formal Falsification Criteria as a Basis for Behavior Planning based on Reinforcement Learning Algorithms***

Am Fachgebiet Fahrzeugtechnik der TU Darmstadt (FZD) wird innerhalb der Projekte PRORETA 5 und UNICAR*agil* an automatisierten Fahrzeugen für den urbanen Verkehr geforscht. Sowohl für die Funktionsspezifikation zum Entwicklungsbeginn als auch für die Online-Verhaltensplanung eines solchen Fahrzeugs wird eine streckenspezifische, semantische Verhaltensbeschreibung entwickelt. Diese muss zum Zweck der Maschinenlesbarkeit analysiert und anschließend formalisiert werden. Auf Basis formaler Kriterien ist anschließend der Einsatz von Reinforcement Learning Algorithmen für die Verhaltensplanung möglich. Neben der Verhaltensplanung bilden diese Kriterien eine Grundlage zur Anforderungs- und Testspezifikation.

Ziel dieser Arbeit ist die Formalisierung von Verhaltensgrenzen mithilfe von Versagenskriterien und die anschließende Anwendung eines Reinforcement Learning Ansatzes zur Verhaltensplanung. Als Grundlage dient die bei FZD entwickelte verhaltenssemantische Szeneriebeschreibung.

Als Ergebnisse der Arbeit sind nachzuweisen:

1. Die Literatur zum Stand der Technik ist bzgl. Verhaltensbeschreibung und -modellierung, Verhaltensplanung auf Basis von Reinforcement Learning sowie Falsifikationskriterien für das Sollverhalten automatisierter Fahrzeuge ausgewertet.
2. Die Einarbeitung in die bei FZD entwickelte verhaltenssemantische Szeneriebeschreibung (BSSD) ist abgeschlossen.
3. Die ausgewählten Streckenabschnitte sind in das BSSD-Format konvertiert.
4. Die formalen Falsifikationskriterien der BSSD als Grundlage für einen Verhaltensplaner auf Basis von Reinforcement Learning (RL) sind abgeleitet.

Fachgebiet Fahrzeugtechnik

Prof. Dr. rer. nat. Hermann Winner

Otto-Berndt-Straße 2
64287 Darmstadt

Bearbeiter:
Felix Glatzki M.Sc.
Tel. +49 6151 16 – 24219
felix.glatzki@fzd.tu-darmstadt.de

Moritz Lippert M.Sc.
Tel. +49 6151 16 – 24244
moritz.lippert@tu-darmstadt.de

Fax +49 6151 16 – 24205
www.fahrzeugtechnik-darmstadt.de

Datum
22.10.2020

CP	Dauer Zwischenkolloq (Vortrag / gesamt)	Dauer Abschlusskolloq (Vortrag / gesamt)
< 18	5 - 7 / 15 min	15 / 45 min
> 18	10 / 20 min	20 / 60 min

TECHNISCHE UNIVERSITÄT DARMSTADT

5. Ein einfacher RL-Verhaltensplaner ist auf die zuvor ausgewählten Streckenabschnitte angewandt worden.
6. Die Methodik und Anwendung sind evaluiert.
7. Die Methodik des Vorgehens und der oben genannten Arbeitsschritte inklusive deren Diskussion sind dokumentiert.

Schwerpunkte der Bewertung:

- Methodik des Vorgehens
- Vollständigkeit
- Belastbarkeit der Argumentation
- Nachvollziehbarkeit

Die Abgabe sämtlicher Messdaten und des Programmcodes in verständlicher Form wird vorausgesetzt. Die Arbeit bleibt Eigentum des Fachgebiets. Auf das Merkblatt des Fachgebiets wird hingewiesen.

Prof. Dr. rer. nat. Hermann Winner

Felix Glatzki, M. Sc.
(Betreuer)

Moritz Lippert, M. Sc.
(Co-Betreuer)

Sworn Declaration

Erklärung zur Abschlussarbeit gemäß § 22 Abs. 7 APB TU Darmstadt

Hiermit versichere ich, Robert Alexander Hülsmann, B. Eng., die vorliegende Master-Thesis / Bachelor-Thesis gemäß § 22 Abs. 7 APB TU Darmstadt ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Falle eines Plagiats (§38 Abs.2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

English Translation for information purposes only:

Thesis Statement pursuant to § 22 paragraph 7 of APB TU Darmstadt

I herewith formally declare that I, Robert Alexander Hülsmann, B. Eng., have written the submitted thesis independently pursuant to § 22 paragraph 7 of APB TU Darmstadt. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources, which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form.

I am aware, that in case of an attempt at deception based on plagiarism (§38 Abs. 2 APB), the thesis would be graded with 5,0 and counted as one failed examination attempt. The thesis may only be repeated once.

Datum / Date:	Unterschrift / Signature:
06.05.2021	Robert Hülsmann

Abstract

For the purpose of compliance with behavioral rules by Autonomous Vehicle (AV), especially in urban traffic, the Behavior-Semantic Scenery Description (BSSD) can be used to describe the limits of the legal behavioral space for each route segment of a road map. In order to test the applicability of BSSD to an online behavior planner, the task of this thesis was to convert selected route segments into the BSSD format, derive behavioral boundaries as formal falsification criteria from the specification of BSSD, and subsequently use them for the application of a Reinforcement Learning (RL) behavior planner.

For this purpose, these criteria were first extracted from the specification, their logical form were identified and formalized. A training environment for the behavior planner including the simulation of other Traffic Participant (TP) and a visualization was created and the falsification criteria and the behavior planner were implemented to the extent as it was possible within the scope of this thesis. Finally, the behavior planner was trained and evaluated.

The formalization and implementation of the falsification criteria highlighted strengths and weaknesses in the machine interpretability of BSSD. Two of the six extracted criteria could not be fully formalized and implemented. For the remaining criteria, however, it was possible to complete this task.

The evaluation of the learned behavior model showed that in the training environment, a vehicle controlled by the simple behavior planner reacts to changes in the maximum permitted speeds and adjusts its speed promptly. Also, lane changes are avoided since they are prohibited or not possible in most places in the selected road section. For compliance with other falsification criteria, improvements must be made to the behavior planner and to the level of penalties for violating the behavior limits respectively rewards for error-free progress. Some suggestions for this have been given in this elaboration.

Zusammenfassung

Zum Zwecke der Einhaltbarkeit von Verhaltensregeln durch autonome Fahrzeuge besonders im urbanen Verkehr kann mit Hilfe der BSSD für jedes Streckensegment einer Straßenkarte beschrieben werden, welche Grenzen des legalen Verhaltensraums dort gelten. Um die Anwendbarkeit von BSSD auf einen Online-Verhaltensplaner zu prüfen, war es die Aufgabe dieser Thesis, ausgewählte Streckenabschnitte in das BSSD-Format zu konvertieren, Verhaltensgrenzen als formale Falsifikationskriterien aus der Spezifikation von BSSD abzuleiten und diese anschließend bei der Anwendung eines RL-Verhaltensplaner einzusetzen.

Hierzu wurden diese Kriterien erst aus der Spezifikation extrahiert, ihre logische Form identifiziert und formalisiert. Eine Trainingsumgebung für den Verhaltensplaner inklusive der Simulation von anderen Verkehrsteilnehmern und einer Visualisierung wurde erstellt und die Falsifikationskriterien und der Verhaltensplaner in dem Maße umgesetzt, wie es im Rahmen dieser Thesis möglich war. Abschließend wurde der Verhaltensplaner trainiert und evaluiert.

Bei der Formalisierung und der Implementierung der Falsifikationskriterien zeigten sich Stärken und Schwächen in der maschinellen Interpretierbarkeit von BSSD. Zwei der sechs extrahierten Kriterien konnten nicht vollständig formalisiert und implementiert werden. Für die restlichen Kriterien konnte diese Arbeit hingegen abgeschlossen werden.

Die Evaluation des gelernten Verhaltensmodells zeigte, dass in der Trainingsumgebung ein Fahrzeug, welches von dem einfachen Verhaltensplaner gesteuert wird, auf Änderungen der zulässigen Höchstgeschwindigkeiten reagiert und seine Geschwindigkeit rechtzeitig anpasst. Auch werden Fahrstreifenwechsel vermieden, da diese im gewählten Streckenabschnitt an den meisten Stellen verboten oder nicht möglich sind. Für die Einhaltung weiterer Falsifikationskriterien müssen Verbesserungen am Verhaltensplaner und an der Höhe der Bestrafungen für die Überschreitungen der Verhaltensgrenzen bzw. der Belohnungen für fehlerfreies Vorankommen vorgenommen werden. In dieser Ausarbeitung wurden einige Vorschläge hierfür gegeben.

Contents

Symbols and Indices

Latin formula symbols:

Symbol	Unit	Description
A	$-$	State-dependent Action Advantage Function
a	$\frac{\mathrm{m}}{\mathrm{s}^2}$	Acceleration
B	$-$	Boundary Violation
b	$-$	Neuron Bias
C	$-$	Collision
d	m	Distance
F	$-$	Falsification Criterion
f	$-$	Activation Function
l	$-$	Lane Change State
O	$-$	Obstruction
P	$-$	Point
p	$-$	Probability
Q	$-$	Quality Function
r	$-$	Reservation Right
S	$-$	Coordinate System
t	s	Time (Step)
T	$-$	Transformation Matrix
v	$\frac{\mathrm{km}}{\mathrm{h}}$	Speed
V	$-$	State Value Function
w	$-$	Neuron Input Weight
x	$-$	Neuron Output Value
x	m	X-axis Component of a Position; Longitudinal Position
y	m	Y-axis Component of a Position; Lateral Position
z	m	Z-axis Component of a Position

Fraktur formula symbols:

Symbol	Unit	Description
$\mathfrak{A}$	–	List of Areas
$\mathfrak{a}$	–	Area
$\mathfrak{B}$	–	List of Boundaries
$\mathfrak{b}$	–	Boundary
$\mathfrak{C}$	–	List of Condition Elements
$\mathfrak{c}$	–	Condition Element
$\mathfrak{e}$	–	Ego Vehicle
$\mathfrak{O}$	–	List of Objects
$\mathfrak{o}$	–	Object
$\mathfrak{P}$	–	List of TP
$\mathfrak{p}$	–	TP
$\mathfrak{R}$	–	List of Reservation Elements
$\mathfrak{r}$	–	Reservation Element
$\mathfrak{S}$	–	List of Segments
$\mathfrak{s}$	–	Segment

Greek formula symbols:

Symbol	Unit	Description
δ	s	Duration
ϵ	–	Exploration Exploitation Trade-off Value
γ	–	Discount Factor
Θ	–	Probability Model

Indices:

Symbol	Description
c	Collision
f	Final
last	Latest Possible
lat	Lateral
lon	Longitudinal
max	Maximum
min	Minimum
NRL	No Red Light Condition
NST	No Stagnant Traffic Condition
o	Obstruction
overtake	Overtaking Regulation
q	Segment Sequence
speed	Speed Limit
stop	Stop Condition; Stop Zone
w	World

List of Abbreviations

ACC	Adaptive Cruise Control
ADAC	General German Automobile Club (ger.: Allgemeiner Deutscher Automobil-Club e. V.)
ADAM	Adaptive Moment Estimation
AI	Artificial Intelligence
AV	Autonomous Vehicle
BSSD	Behavior-Semantic Scenery Description
BT-KAT-OWI	Uniform Federal Catalogue of Offences for Road Traffic Offences (ger.: Bundeseinheitlicher Tatbestandskatalog für Straßenverkehrsordnungswidrigkeiten)
CC	Cruise Control
DDQN	Dueling Deep Q-Networks
DQN	Deep Q-Networks
FOL	First-Order Logic
FoV	Field of View
FZD	Department of Automotive Engineering at Technische Universität Darmstadt (ger.: Fachgebiet Fahrzeugtechnik der Technischen Universität Darmstadt)
HOL	Higher-Order Logic
ID	Identification Number
IDE	Integrated Development Environment
LTL	Linear Temporal Logic
ML	Machine Learning
NN	Artificial Neural Network
PolFHa	Police Specialist Handbook (ger.: Polizei-Fach-Handbuch)
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
StVO	Road Traffic Regulations of the Federal Republic of Germany (ger.: Straßenverkehrs-Ordnung der Bundesrepublik Deutschland)
TP	Traffic Participant
TTCCP	Time To Critical Collision Probability
XML	Extensible Markup Language

List of Figures

List of Tables

1 Introduction

In the field of autonomous driving intensive research and development has been going on for several years. The need to comply with applicable traffic regulations, to have a wide all-round view and to predict and assess the behavior of other TP makes this field of application one of the most difficult of Artificial Intelligence (AI). A requirement for success is an accurate representation of relevant parts of the environment of an AV, based on which it can plan its behavior.

For this purpose, mostly semantic environment representations, such as OpenDRIVE[1] or Lanelet2[2], are used, which represent road networks with their geometric shapes, as well as the static objects in the environment, such as signs or traffic islands. Compared to a geometric environment map, which is more intuitive for humans, semantic representations have the advantage that they are easier to read by machines. In addition, it is easier to map information that has no or little geometric context.

At the Department of Automotive Engineering at Technische Universität Darmstadt (ger.: Fachgebiet Fahrzeugtechnik der Technischen Universität Darmstadt) (FZD) a novel, route-specific, semantic description of behavior is currently being developed. Mainly, this thesis refers to an internal draft of the BSSD specification.[3]

Update May 2022: One year after this thesis was submitted, the internal draft of BSSD resulted in two publications.[4,5]

In contrast to OpenDRIVE, for example, BSSD does not describe geometric characteristics of the road network but the location-specific behavioral rules for each position in the road network. It is a specification of a file format, which is based on Extensible Markup Language (XML). The edges of a road network connecting nodes are divided into segments within which the specified behavior does not change per lane.

This thesis deals with the development of an online behavior planner based on BSSD for automated vehicles with focus on urban traffic. The goal is to formalize falsification criteria using the behavioral specifications encoded in BSSD to define the limits of the intended behavior of such a planner. Subsequently, this planner is to be implemented using RL, a subset of Machine Learning (ML). Accordingly, the development of an architecture is part of this thesis, which enables the training of the planner in this way. Before that a synthetic road map (demo map) will be converted into BSSD, which is used for testing purposes during development and evaluation.

[1] Dupuis, M.: OpenDRIVE (2006).

[2] Poggenhans, F. et al.: Lanelet2 (2018).

[3] Glatzki, F.; Lippert, M.: BSSD Specification (2021).

[4] Glatzki, F. et al.: Behavioral Attributes for a BSSD (2021)

[5] Lippert, M. et al.: BSSD of Road Networks for Automated Driving (2022)

1.1 Motivation

BSSD has the potential to facilitate compliance with traffic regulations for AV and especially to monitor compliance by control authorities during development and testing. The motivation of this thesis is to test BSSD for its applicability by behavior planners for AV.

A RL architecture is used as behavior planner, as these are currently very promising due to the fact that no training data is required to learn a behavior. Instead, an AV moves in a simulation environment. Simulations are a popular tool to accelerate the development of autonomous driving, as thousands of driving scenarios and different design parameters can be virtually tested accurately, quickly, and inexpensively.[6] For each action of the planner in the simulation, it is rewarded or penalized and continuously improves its behavior – comparable to a human learner driver. Here, BSSD describes the limits of legal behavior, i.e., the boundaries of the space of possible actions that the planner may not exceed.

However, whether a behavior planner is trained in the correct way can only be determined if it can be shown that the falsification criteria, i.e., the criteria describing the different ways in which the boundary can be violated, are correctly formulated. To do this, falsification criteria were derived from the BSSD and formalized so that it can be shown that the planner's evaluation is correct at each training step.

1.2 Ascertainment of Assignment

To test BSSD for its applicability by online behavior planners, the goal is the formalization of falsification criteria, which are to be derived from the BSSD, and the following applying of a RL algorithm for behavior planning. For this purpose, the falsification criteria represent the behavioral limits.

The task is divided in such a way that at first a literature research on the state of the art regarding behavior description and modeling, behavior planning based on reinforcement learning as well as falsification criteria for the target behavior of automated vehicles is carried out. Following this, a familiarization with the BSSD specification is necessary in order to then select route sections for subsequent work packages and convert them into the BSSD format. Then, the derivation of formal falsification criteria from the BSSD follows as a basis for a behavior planner based on RL, which is subsequently to be implemented in a simple variant and applied to the previously selected route sections. Finally, the methodology and application are to be evaluated and the methodology of the procedure and the work steps including their discussion are to be documented.

1.3 Methodology

Figure 1-1 shows how the task was divided into main work steps and further into task packages. These are explained in more detail below.

The first main step of this thesis was a *Literature Research* of the three most important parts of this work:

- description and modeling of behavior
- behavior planning based on RL
- falsification criteria for target behavior of AV

[6] Sovani, S.: Simulation beschleunigt Entwicklung des autonomen Fahrens (2017).

The results of the research are described in section 3.

This was followed by the main work step of the *Scenery Description*, in which the first work package was the familiarization with BSSD. A synthetic demo map was then generated and subsequently converted to BSSD. This is described in section 4.

Section 5 describes the measures taken to reduce the complexity of the topic so that it can be dealt with in a meaningful way in this thesis. (This is not shown in figure 1-1.)

As soon as the topic-specific research and familiarization with BSSD was completed, the main work step to *Formalize the Falsification Criteria* was started. Here, potential violations of the traffic rules that fit the abstraction level of BSSD were studied at first in order to formalize them as a formal-logical proposition in the second step. This is described in section 6.

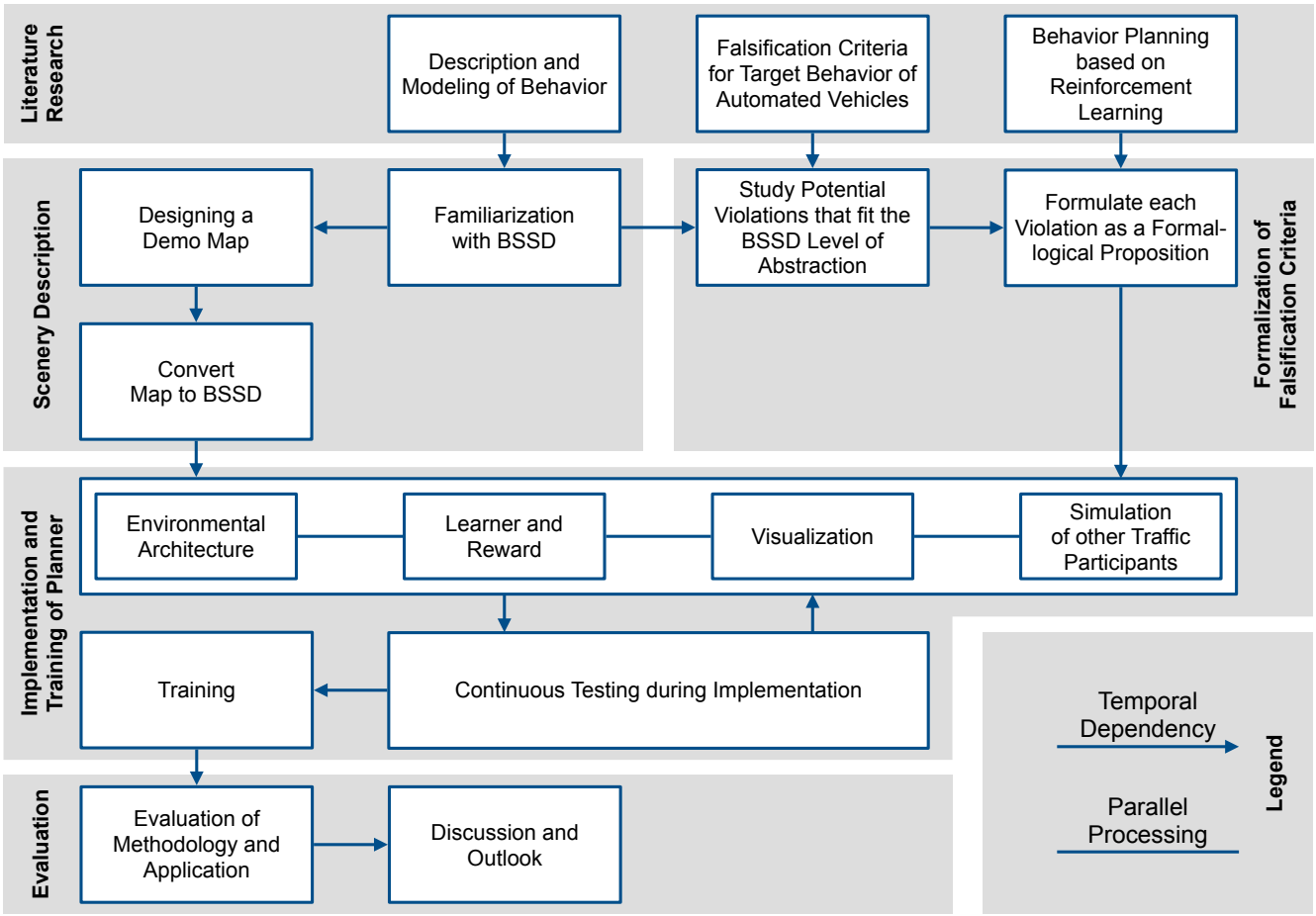


Figure 1-1: Overall Methodology.

When *Scenery Description* and *Formalization of Falsification Criteria* were completed, the *Implementation and Training of Planner* followed. This was done in work packages, which are described in detail in section 7. Since an intensive parallel processing had to take place, a certain chronological order was difficult to define in advance. Tests were always performed during and after the processing of task packages and code segments in order to be able to continuously monitor the error-free functionality of the software project.

An *Evaluation*, as described in section 8, is the last main step in the process. This also includes a discussion of the results. Finally, an outlook is given on topics that are recommended as a continuation of this work. However, for a start necessary foundations are explained in section 2.

2 Foundations

The following topic-specific foundations should contribute to the comprehensibility of the thesis.

2.1 Scenario, Scene, Scenery and Situation

In this thesis the terminologies *scenario*, *scene*, *scenery* and *situation* are used as defined by Ulbrich et al.[7] for AV. „A *scenario* describes the temporal development between several *scenes* in a sequence of *scenes*.“ „A *scene* describes a snapshot of the environment including the *scenery* and dynamic elements, as well as all actors' and observers' self-representations, and the relationships among those entities.“ „The *scenery* subsumes all geo-spatially stationary aspects of the scene. This entails metric, semantic and topological information about roads and all their components like lanes, lane markings, road surfaces, or the roads' domain types.“ While *scenario*, *scene* and *scenery* describe the environment objectively, a *situation* is the subjective derivation of a scene from the point of view of an object as a result of a process of information selection and augmentation.

2.2 Artificial Neuronal Network

An Artificial Neural Network (NN) is a sub-discipline of AI. Such a network consists of a large number of mathematical models of neurons arranged in layers. The first layer is the so-called input layer, the last layer is the output layer and the layers in between are the hidden layers. Often, each neuron on a layer is connected to all neurons on the previous layer. Such a layer is called fully connected layer. Through these connections, input data in the form of a number vector passes from the input layer through all neurons to the output layer. The vector that arrives at the output layer is the result of the computation by the NN. A NN with several layers is called a deep NN. There is no definition how many layers in minimum have to be part of an NN to call it *deep*.

A neuron i has several parameters which are adjusted during the training process. These parameters are the weights w_{ij}, which describe, how the output x_{i} neuron i is influenced by the output x_{j} from neurons $j \in \{1, \ldots, n\}$ of the previous layer. Another parameter is the offset, called a bias b_{i}. Within a neuron, the weighted input values and the bias are summed. In the following, the sum is mapped the output of neuron i by an activation function f.[8]

$$x_{\mathrm{i}} = f\left(\sum_{j=1}^{n} w_{\mathrm{ij}} x_{\mathrm{j}} \; + b_{\mathrm{i}}\right)$$

The activation function in each neuron must be non-linear, otherwise the NN loses its multilayered character and can solve only less complex tasks. The activation function Rectified Linear Unit (ReLU) f_{ReLU} has been widely used for several years, since in 2011 Glorot et al.[9] proved that networks with ReLU lead to better results than the functions predominantly used until then. It represents an identity in

[7] Ulbrich, S. et al.: Defining Terms for Automated Driving (2015).

[8] Ertel, W.: Grundkurs Künstliche Intelligenz (2016), pp. 268-269.

[9] Glorot, X. et al.: Deep Sparse Rectifier Neural Networks (2011).

the positive value range, i.e. a linearity, but produces the output 0 for all negative values, as illustrated in figure 2-1.

$$f_{\mathrm{ReLU}}(x) = \max(0, x)$$

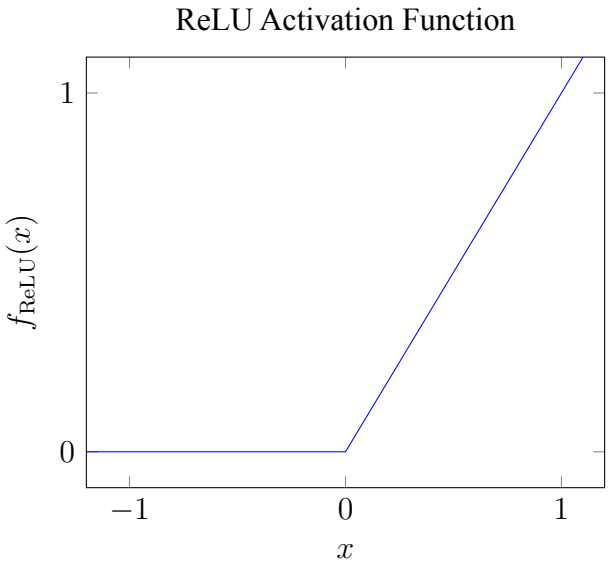


Figure 2-1: ReLU Activation Function.

Using pairs of input vectors and the required output vectors of the whole NN, the parameters of the neurons can be determined, e.g. from a training data set. For this purpose, the actual output of the NN is first calculated by forward propagation with initial parameter values and this output is compared with the target output. From this the loss (prediction error) is calculated. Using an optimization function, the parameters of each neuron are changed during backward propagation by training several thousand input-output pairs, to reduce the loss to minimum. The optimizer called Adaptive Moment Estimation (ADAM) is a "simple and computationally efficient algorithm for gradient-based optimization of stochastic objective functions"[10]. This optimizer is often applied to train networks, as it provides fast convergence in many non-convex optimization problems.[10]

The learning rate is an essential hyperparameter when training NN. It is the step size during optimization and thus defines how much the weights change between two iterations in the direction determined by the optimizer. The value of a good learning rate often depends on the NN and its task and must be found out experimentally. If the learning rate is too small, training takes a long time; if it is too large, the optimizer jumps over the optimal value without reaching it. To make the training process efficient, the optimization is not performed after every single input-output pair, but for groups of pairs, the so-called batches.

[10] Kingma, D. P.; Ba, J. L.: Adam (2015).

2.3 Reinforcement Learning

„The goal of RL […] is to learn good policies for sequential decision problems, by optimizing a cumulative future reward signal.“[11] In the field of RL, very good results are currently being achieved with Q-learning.[12] A quality function Q is learned that specifies for each state how good which action is. „An agent tries an action at a particular state, and evaluates its consequences in terms of the immediate reward or penalty it receives and its estimate of the value of the state to which it is taken.“[13]

Deep Q-Networks (DQN) are the union of Q-learning and deep NN.[14] Here, a loss is not minimized, as with traditional NN, but the reward is maximized. But Q-learning „is known to sometimes learn unrealistically high action values because it includes a maximization step over estimated action values, which tends to prefer overestimated to underestimated values.“[11] To avoid this, Double Q-learning was introduced. Here Q is separated into Q^A and Q^B. Q^A is updated using Q^B and vice versa. This double estimation approach decouples the action choice from the action evaluation during Q-learning.[11,15]

Dueling Deep Q-Networks (DDQN) are used to achieve a further increase in performance in the area of RL. It is a single DQN, which starts as usual with layers. But then the network is divided into two streams, so that separate estimates can be made for state value function V and state-dependent action advantage function A. The approach is that although „in some states, it is of paramount importance to know which action to take, but in many other states the choice of action has no repercussion on what happens.“[16] This allows learning to be generalized across different actions. In order to train the NN like a normal NN, the two functions are combined to Q again. But the action is selected by the highest value of A.

Since in many applications the input of a RL algorithm depends on the selected action of the last time step, it must be ensured that during training on the one hand different actions are tried in different states, but later a good strategy is found and this is further refined. For this purpose the parameter ϵ is used, which describes the trade-off between exploration and exploitation. At the beginning of training, $\epsilon = 1$, so the choice of action is always random and many state-action combinations are tried. With each training iteration, ϵ gets smaller thereby increasing the probability that the action was actually chosen by the RL algorithm. When $\epsilon = 0$ there is no more random influence and the strategy found so far has manifested. To test the trained RL agent, $\epsilon = 0$ is then always chosen. This is called the Epsilon-Greedy algorithm.[17,18]

Another important variable of RL is the discount factor γ. It describes whether a RL agent should be optimized for short-term or long-term rewards. ”The discount factor γ is used to exponentially decrease the weight of reinforcements received in the future”[19]. If $\gamma = 0$, then only the current reward is considered, while if $\gamma = 1$, the RL agent will seek a long-term high reward.

---

[11] Hasselt, H. v. et al.: Deep Reinforcement Learning with Double Q-learning (2016).

[12] Hessel, M. et al.: Rainbow (2018).

[13] Watkins, C. J. C. H.; Dayan, P.: Q-learning (1992).

[14] Mnih, V. et al.: Human-level control through deep reinforcement learning (2015).

[15] Hasselt, H. v.: Double Q-learning (2010).

[16] Wang, Z. et al.: Dueling Deep Reinforcement Learning (2016).

[17] Ertel, W.: Grundkurs Künstliche Intelligenz (2016), p. 328.

[18] Parkinson, A.: Epsilon-Greedy Algorithm for Reinforcement Learning (2019).

[19] Harmon, M. E.; Harmon, S. S.: Reinforcement Learning (1997).

3 Related Work

The following three subsections present the results of the literature review.

3.1 Description and Modeling of Behavior

A necessary condition to provide rule-compliant autonomous driving is to have access to a complete description of the prescribed behavioral constraints.

Kohlhaas et al.[20] use ontologies, the mapping of knowledge using uniform terminology and connections between those terms,[21] to model relevant traffic elements and relationships. This results in a semantic representation built on top of a geometric representation. The semantic state space includes, among other things, conditions that model traffic rules.

Another ontology-based framework for AV decision making based on traffic regulations was developed by Buechel et al.[22] In their paper, the authors also highlight the advantages of using ontology. For example, an implementation based on that uses „the same concepts as used in the traffic regulations and is both human and machine readable“[22]. It should also enable improved situational awareness, the application of traffic situation dependent decisions, and driving in compliance with traffic regulations in complex traffic scenarios.

The open source project Lanelet2 by Poggenhans et al.[23] represents another map framework. Here a map consists of three layers: The physical layer contains real, observable elements. In the relational layer, the elements of the physical layer are linked to lanes, areas and traffic rules. The topological layer results from the contexts and neighborhood relationships of relational layer. A traffic light, for example, is linked to the lanes to which it applies. For traffic rules, this should make it transparent to which type of TP they refer. According to the authors, it would then only have to be comprehensible whether the element associated with a rule also valid for this TP.

3.2 Behavior Planning Based on Reinforcement Learning

As already mentioned in the section 2, currently there is much progress in the field of reinforcement learning with Q-learning algorithms. DeepTraffic by Fridman et al.[24] is a traffic simulation for learning a self-driving car using DQN. It can be run in the browser and serves as an environment for competition between humans to find the best hyperparameters and for research related to it.

In this environment, the ego vehicle is steered along an endless seven-lane road by the DQN through heavy traffic. The goal is to move as fast as possible without touching other TP. The AI of the ego agent receives the environment as a 2D plan view with a fixed size of the Field of View (FoV), in which ego always keeps the same position. ”This means, the dynamics of DeepTraffic are relative to the ego vehicle.”[24]

[20] Kohlhaas, R. et al.: Semantic state space for high-level maneuver planning (2014).

[21] Hesse, W.: Ontologie(n) (2002).

[22] Buechel, M. et al.: Ontology-based traffic scene modeling (2017).

[23] Poggenhans, F. et al.: Lanelet2 (2018).

[24] Fridman, L. et al.: DeepTraffic (2018).

DeepTraffic uses Experience Replay. It helps to make the learning process more efficient by being able to revert to previous experience during the learning process.[25] "By using experience replay the behavior distribution is averaged over many of its previous states, smoothing out learning and avoiding oscillations or divergence in the parameters."[26]

In addition, DeepTraffic uses a safety system that relieves the learning process of the task of avoiding collisions. It "prevents the vehicle from colliding by either (1) preventing an action which would lead to a crash or (2) altering the speed of the vehicle."[27] This should allow the learning process to focus more on learning efficient movement patterns.

A safety system is also used in the framework for tactical decision making by Hoel et al.[28] The space of actions that the ego vehicle can select includes:

- Stay in current lane, keep current Adaptive Cruise Control (ACC) setpoint,
- Stay in current lane, decrease ACC setpoint,
- Stay in current state, increase ACC setpoint,
- Change lanes to right, keep ACC and
- Change lanes to left, keep ACC.

These high-level actions are then passed to a lower level operational decision making layer, the Intelligent Driver Model. The use of that model and ACC enables the allowed time gap to be respected. Furthermore, "at every time step, the action space is pruned, in order to remove actions that lead to collisions"[28] A lane change action is only allowed if the ego vehicle or the vehicle in the new lane must brake with a lower acceleration than the threshold $a_{\min}$ to avoid a collision. According to the authors, Intelligent Driver Model itself is always crash-free, which should mean that the ego vehicle will never cause a collision.

In the framework of Hoel et al.[28] a continuous driving on the freeway is simulated like in DeepTraffic[27]. The state space consists mainly of physical states and driver states of ego and the other TP. Physical states are the longitudinal and lateral positions and speeds. Positions and speeds of the surrounding vehicles are expressed relative to the ego vehicle. Driver State includes other information about the TP such as set speed or set time gap. Each entry of the state space is converted to a state vector and normalized to $[-1, 1]$ before entering the neural network.

Since the goal of this learning algorithm is also to progress quickly and safely, but Intelligent Driver Model does not allow accidents, the reward model is simply to assign the highest value when the desired speed is reached. However, a penalty value is still subtracted for each lane change. The idea for the learning algorithm used here is based on the idea of AlphaGo Zero[29] and is a combination of Monte Carlo tree search and deep reinforcement learning. The NN includes two special types layers: convolution and maxpool layer. In the convolution layer, the signal from the previous layer is discretely convolved with a

[25] Lin, L.-J.: Diss., Reinforcement Learning for Robots (1993), pp. 29-48.

[26] Mnih, V. et al.: Playing Atari with Deep Reinforcement Learning (2013).

[27] Fridman, L. et al.: DeepTraffic (2018).

[28] Hoel, C.-J. et al.: Planning and Deep Reinforcement Learning (2020).

[29] Silver, D. et al.: Mastering Go without human knowledge (2017).

filter kernel. This layer is usually followed by a pooling layer. With maxpooling, only the information that has the highest value in a dynamic search area is further processed. Convolutions and pooling allow the number of TP to be changed without having to retrain the whole network.

Due to the fact that an AV is a safety-critical system, it is especially important here to detect and eliminate possible safety violations of the ego vehicle, even if adversarial agents are acting on the ego vehicle. This is Qin et al.'s[30] motivation for their method, in which adversarial agents are trained to point out flaws in ego's behavior. Adversarial agents can be other cars, pedestrians, bicyclists, traffic lights, or even weather events.

"One challenge is that it is easy to obtain maximally antagonistic adversaries that would cause all but the most overly conservative ego agents to violate their safety requirements."[30] To solve this problem, here they adjust the intensity of the adversarial agents by restricting their behavior to certain rules, such as obeying traffic rules. They show that their algorithm is guaranteed to find errors of the system under test if they exist and demonstrate their approach using three case studies from autonomous driving.

The paper by Wang et al.[31] also addresses the problem that RL methods often fail to adapt to safety-critical scenarios because the learned policy overfits the training environment. They present a generic framework for integrating temporal-logical falsification into adversarial learning to improve the robustness of policies. For an explanation of temporal logic, see subection 3.3.

3.3 Falsification Criteria for Target Behavior of Automated Vehicles

According to Rizaldi et al.[32], traffic rules are rarely specified in as much detail as an engineer would expect. They figured out that this makes it impossible to monitor the behavior of AV. Nevertheless, using a theorem prover, they attempt to first abstractly codify the traffic rules and then specify each atomic proposition in a verified manner. Here, in this paper, they limit themselves to the paragraph of overtaking in Road Traffic Regulations of the Federal Republic of Germany (ger.: Straßenverkehrs-Ordnung der Bundesrepublik Deutschland) (StVO) and were able to formalize a subset of the traffic rules for overtaking, including the safety distance.

The theorem prover generates code that can be used to formally monitor compliance with traffic rules. Traffic rules are generated in Linear Temporal Logic (LTL), using boolean predicates in discrete time.[33] These are defined in Higher-Order Logic (HOL) by taking each predicate found in the law text as an atomic proposition. HOL is an extension of First-Order Logic (FOL).[34] Predicate symbols of HOL and FOL include among others terms (e.g., t_1, t_2, ...), *logical and* ($\wedge$), *logical or* ($\vee$) *logical negation* ($\neg$), *existential quantifier* ($\exists$), *all-quantifier* ($\forall$), and *functions* (e.g., p, which maps x to $p(x)$).[35] LTL additionally has temporal operators, such as *next*, *until*, *always*, and *eventually*. [36]

[30] Qin, X. et al.: Falsification with Dynamically Constrained Reinforcement Learning (2020).

[31] Wang, X. et al.: Falsification-Based Robust Adversarial Reinforcement Learning (2020).

[32] Rizaldi, A. et al.: Formalising and Monitoring Traffic Rules (2017).

[33] Donzé, A.: On Signal Temporal Logic (2014).

[34] Ahmed, A. et al.: Formal Asymptotic Analysis (2018).

[35] Ertel, W.: Grundkurs Künstliche Intelligenz (2016), p. 44.

[36] Esterle, K. et al.: Formalizing Traffic Rules for Machine Interpretability (2020).

The approach of Karimi and Duggirala[37] provides a simple mapping from California traffic rules for uncontrolled intersections to a formal model and from this model to its implementation. According to their description, in any traffic scenario, the correct behavior depends on three factors:

- Static features: e.g., lane shapes, lane types, lane overlaps, signs, etc.
- Dynamic features: e.g., arrival at an intersection, turn signals, etc.
- Traffic rules: Statement about the static and dynamic features of a scenario.

Based on Rizaldi et al.[38], Karimi and Duggirala[37] followed a top-down approach to formalize the rules. According to this, traffic rules are first studied to identify the logical form of the rules and the concepts they use. After that, each rule is formulated as a formal logical sentence using atomic predicates, and then each predicate is defined. Atomic predicates are modeled using spatio-temporal mathematical objects. According to the authors, the separation of logical form and mathematical concepts provides flexibility to incorporate various types of traffic rules in the driver handbook. To formalize an entire set of rules an iterative approach can be attempted, the authors say. At each iteration, additional rules are added to the context while maintaining consistency. To maintain consistency, it may be necessary to redefine an earlier rule.

Here, the logical form of a traffic rule is modeled as a formula of the FOL. A formula is a combination of predicates, logical links, and quantifiers ($\exists$ or $forall$). Predicates in the formula refer to static and dynamic characteristics of the traffic scenario. Static features are modeled by mathematical concepts such as quantities, functions, etc., while dynamic features are modeled by events. Therefore, each predicate representing an event has a time parameter. Subformulas are defined that have an intuitive meaning as auxiliary predicates. This serves to simplify the expression of long formulas.

Esterle et al.[39] also refer to Rizaldi et al.[38] in their paper on formalizing traffic rules for machine interpretability. Using LTL, they describe the actual state of the environment, as well as behavioral rules for TP. In their paper, they restrict themselves to StVO and to rules that are applicable on two-lane roads, such as freeways. There is no consideration here of rules that include pedestrians or cyclists.

According to their approach, the first step is to identify a rule and divide it into initial premise (*if*) and conclusion (*then*). If there is no premise, this rule starts with *always*. Second, all exceptions to the premise are identified and these exceptions are used in negated form to update the premise. Last, the premise and conclusion are decomposed into labels or new rules.

[37] Karimi, A.; Duggirala, P. S.: Formalizing traffic rules for uncontrolled intersections (2020).

[38] Rizaldi, A. et al.: Formalising and Monitoring Traffic Rules (2017).

[39] Esterle, K. et al.: Formalizing Traffic Rules for Machine Interpretability (2020).

4 Scenery Description with BSSD

After the literature review, BSSD was investigated in detail, a demo map was produced and converted to BSSD.

4.1 Explanation of BSSD Specification (Familiarization with BSSD)

This thesis refers to a description that is currently under development at FZD: The Behavior-Semantic Scenery Description (BSSD). It differs from other behavior descriptions in the fact that ways of a road network are segmented so that the prescribed behavior boundaries are homogeneous for each lane of a segment. Where ways intersect, there are nodes containing connecting ways, which connect the ways coming from the outside.

BSSD is limited to the modeling of behavioral specifications for motor vehicles. The development of the specification is based on StVO. The focus of the development is on the modeling of urban road networks and behavioral specifications there. As this thesis mainly refers to the BSSD specification[40], figure 4-1 shows the inheritance diagram from that document, which makes the connection of the elements of BSSD visible. The elements are described in more detail below.

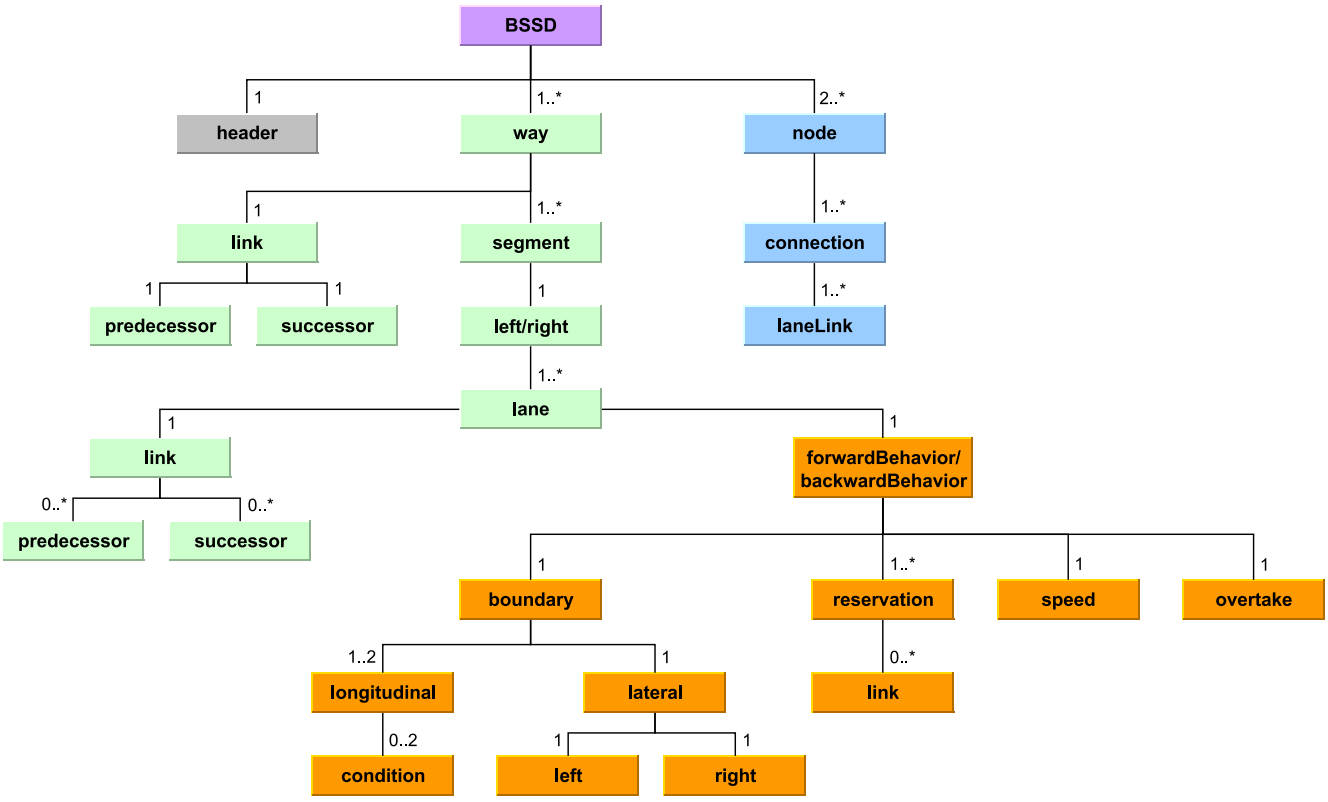


Figure 4-1: BSSD Inheritance Diagram.[40]

The BSSD of a road network is to be stored in XML format. Such a description consists of an element <header> with general information about the description as well as at least one element <way> and at

[40] Glatzki, F.; Lippert, M.: BSSD Specification (2021).

least two elements <node>. The assignment of Identification Number (ID) for elements in BSSD always start with "1" (with the exception of <lane>).

4.1.1 Way Element

Each <way> element has the following attributes.

- id specifies the ID of the way, unique for the whole document.
- from and to contain the ID of the nodes connected by this way. They also define the reference direction of the way. Forward in reference direction means from node from to node to. This definition is important for the further description.
- node describes whether this way is a connecting way. If no, then the value is "-1" and if yes, then the value is the ID of the node where the way is located.

A way has one <link> element with one <predecessor> and one <successor> element each, which specify the predecessor and successor elements with respect to the reference direction. These two children of <link> have the attributes elementType and elementId. In a normal way, the type of the successor and predecessor is a node (elementType="node"). For a connecting way, they are ways (elementType="way"). elementId accordingly specifies the ID of the way or node.

Likewise, a way contains at least one element. According to BSSD, a segment includes as attribute only an ID unique within the way and as child elements one or more lanes (<lane>), which however are grouped at first using the elements <left> and <right>. These two groups refer to the reference direction serve for better human comprehensibility.

The ID of an element <lane>, which is unique within a segment, also specifies how lanes are arranged within it. If the direction of travel of a lane is the same as the reference direction, then its id attribute has a negative sign. It starts with id="-1" and the numbers get smaller by one with each lane going outward starting from the center. The innermost lane leading against the reference direction has id="1". Each further lane in this direction has a number one higher the further out this lane is. If a segment contains only one lane, which is used in both directions, it has the id="-1". The value "0" for the id is not assigned.

A lane includes a lane link object (<link>) which may include none, one or more <predecessor> and <successor> elements that specify which lanes from the predecessor and successor ways this lane is linked to. Therefore these elements have the attribute laneId, which specifies the ID of the lane from the linked way.

In addition, a lane has one <forwardBehavior> and one <backwardBehavior> element. These describe the behavior limits that apply to this way segment lane unit, in the following called area, and for the corresponding direction of travel in relation to the reference direction.

<boundary> is one of the elements of the bahavior elements and contains the elements <longitudinal> and <lateral>, which describe the regulations for a longitudinal entry, i.e. the change from one segment to the next, and lateral entry, i.e. a lane change, into this area. <lateral> always exists once, while <longitudinal> can exist at least once, but even twice, if traffic lights regulate the entry.

The <longitudinal> element has the crossing attribute, which can have the values "allowed", "conditional", "prohibited" or "not possible", and the optional trafficLightActive attribute (default value is "no"). If the longitudinal entry to this area can be regulated by traffic lights in principle, two elements <longitudinal> exist within <boundary> and one has the value "yes" for the attribute trafficLightActive, and the other has "no". The elements describe the entry rules for the cases whether there are active traffic lights or not. In the case of active traffic lights, we do not consider further at this point which signal they show. Instead, any different specifications can be modeled using the <condition> child element. If there are no traffic lights, there is only one <longitudinal> element. Here, the trafficLightActive attribute can be omitted.

If element <longitudinal> has the attribute-value pair crossing="conditional", it must contain at least one and at most two <condition> elements. All conditions must be satisfied for a longitudinal entry to be legal. A conditional element has an attribute type, which can have the following values.

- "stop": The entrant must have stopped before the entry.
- "no stagnant traffic": The entry is only allowed if there is no stagnant traffic within this area.
- "no red light": The entry may only take place if the traffic lights, which regulate the longitudinal entry, do not show red.

The <lateral> element always contains two children: <left> and <right>. They specify the lateral regularities for leaving the area via the respective side, related to direction of travel i. e. behavior direction and not to reference direction of the way. <left> and <right> each have the attribute crossing, which can have the values "allowed", "prohibited" or "not possible".

In addition to <boundary>, a behavior element also includes at least one <reservation> element. The reservation elements describe which TP have priority in this area. These conditions must be met when entering as well as while remaining in the area. The reservation elements have the following attributes.

- Attribute id specifies an ID unique within <forwardBehavior> or <backwardBehavior>.
- Attribute type has one of the following values.
  - Value "own-reserved" means that the area is reserved for a TP that enters the area in the behavior direction – that is the ego vehicle.
  - Value "externally-reserved" means that the area is reserved for TP entering the area against behavior direction.
  - Value "equally-reserved" means that the area is equally reserved for TP entering the area from different directions.
- Attribute object restricts the reservation to a specific kind of TP. The possible values are: "pedestrian", "bicycle", "motor vehicle", "railed vehicle". This attribute is only available if "own-reserved" is not specified for type, since BSSD is restricted to motor vehicle as ego vehicle.

- Attribute `trafficLightActive` is specified if traffic lights specify the right of reservation. In this case there are at least two elements <`reservation`> with this attribute and once each the value `"yes"` and `"no"`.

If the `type` attribute of a <`reservation`> element has the value `"externally-reserved"` or `"equally-reserved"`, then this element must have at least one reservation link (<`link`>) as a child element. Reservation link has the following attributes.

- `id` specifies an ID unique within <`reservation`>.
- `linkedWay` specifies the ID of the way used by a TP entering this area, which has the reservation right for this area because it enters via linked way.
- The specification for `linkedSegment` works the same as described before, except that the segment is inside the way that is specified via `linkedWay`.
- `linkedArea` can specify the ID of the lane within the segment specified by `linkedSegment`, over which a TP enters this area, which has the reservation right for this area, because it enters over the linked lane. Alternatively, the value can be `"0"` if such a TP can enter via all lanes of linked segment. Or `linkedArea` has the value `"-inf"` ($-\infty$) or `"inf"` (∞) if such a TP can come outside the regular movement space (to the right or left of the way, respectively). This last case covers e.g. the modeling of crossing pedestrians who have priority at the crosswalk.
- `targetWay`, `targetSegment`, and `tartgetArea` are used similarly to `linkedWay`, `linkedSegment`, and `linkedArea`, but they are optional attributes and specify the area where the reservation receiving object moves to.

<`forwardBehavior`> and <`backwardBehavior`> each have a further child <`speed`>, which specifies the maximum speed allowed in this area in units of $\mathrm{km/h}$. The attributes are as follows.

- `max` generally specifies the maximum speed allowed.
- `timeMax` is optional and specifies another maximum allowed speed, which is valid in a certain time interval.
- `timeInterval` must be given if `timeMax` is specified, and defines the time condition for this additional speed specification.
- `wetMax` is optional and describes another allowed maximum speed, which must be kept if the road is wet.

The last element within the behavior elements is <`overtake`> with the attribute `permission`. The values `"yes"` and `"no"` define the permission to overtake other TP. ”It does not matter whether the two participants are in the same lane or not.”[41]

[41] Glatzki, F.; Lippert, M.: BSSD Specification (2021).

4.1.2 Node Element

As explained at the beginning, there are at least two <node> elements at the top level in a BSSD modeling in parallel to <way>. Such an element has the attribute `id` which indicates the ID unique in the entire document. The attribute `infinitesimal` with the possible values `"yes"` and `"no"` describes whether this is an infinitesimal node. Such a node is used if a way is divided without an actual node area, e.g. if different lanes continue in different directions, as turn lanes on the freeway do.

Element <node> contains child elements <connection> which establish correspondences between outer ways and connecting ways. The `id` attribute contains the unique ID of this connection within a node. The `incomingWay` and `outgoingWay` attributes specify the ID of the outer ways, and `connectingWay` specifies the number of the way within the node that connects the two outer ways. If the node is an infinitesimal node, then `connectingWay="-1"`.

Inside <connection> there is at least one node lane link (<laneLink>). It describes the links of lanes of the ways specified in the parent node. Accordingly, the attributes are `incomingLane`, `connectingLane` and `outgoingLane` and indicate the ID of the corresponding lanes.

4.1.3 General Behavior Rule

In addition to the XML structure of a scenery description, the BSSD specification defines two behavioral constraints that apply everywhere and are thus complementary to the requirements described within the <forwardBehavior> and <backwardBehavior> elements.

The first rule deals with collisions: ”A traffic participant shall avoid foreseeable collisions both within a reservation area [...] and when changing areas, regardless of the reservation type of the area.”[42] Reservation area means nothing else than a way-segment-lane unit, which is simply called area in this thesis.

The second rule deals with obstructions: ”When entering a new lane area, traffic participants with the same reservation right for this area shall not be obstructed.”[42] Lane area here also means a way-segment-lane unit. It does not matter whether the ego vehicle enters the new area longitudinally or laterally. By *same reservation right* is meant that of this area the same <reservation> element or several reservation elements, which can be assigned to the ego vehicle and the other TP, have the same values at the attribute `type`. TP with lower reservation right may be obstructed, but collisions must be avoided by the first rule. TP with higher reservation right must not be obstructed, but this is regulated by description of the reservation elements in the BSSD specification.

4.2 Designing a Demo Map

In order to be able to test the formalization of the behavior limits at a later stage and to create a RL behavior planner based on these criteria, which can learn a desired behavior in a simulation environment, suitable sceneries must be available. To this end, within the scope of this thesis, route sections are to be selected and then converted into the BSSD format.

[42] Glatzki, F.; Lippert, M.: BSSD Specification (2021), p. 6.

To achieve this goal, a synthetic map was created that covers the different behavioral specifications that BSSD can describe and, on the other hand, represents sceneries as realistic as possible. These specifications to be covered are:

- Longitudinal Boundary
  - Crossing: allowed, conditional, prohibited, not possible
  - Traffic Light: no, active, inactive
  - Condition: stop, no stagnant traffic, no red light
- Lateral Boundary
  - Crossing: allowed, prohibited, not possible
- Reservation
  - Type: own-reserved, externally-reserved, equally-reserved
  - Object: pedestrian, bicycle, motor vehicle, railed vehicle
  - Traffic Light: no, active, inactive
  - use of reservation links
  - Speed: $\mathbb{R}$
  - Overtake Permission: yes, no

For reasons of complexity reduction, the behavioral specification for maximum speed is reduced to one attribute. Weather and time-dependent speeds are not considered. (See also section 5.) Nevertheless, this attribute with its continuous values is a challenge to cover all possible values in the map. Therefore, only the values that frequently occur in urban sceneries were covered: 7km/h, 30km/h, 40km/h, 50km/h, 60km/h, 70km/h.

Figure 4-2 shows the resulting map. It resembles a lying eight and thus offers an intersection in the center of the figure. This figure eight is supplemented by another way and a T-junction on the right side of the figure. This choice as a road network allows for a few different routes despite having few ways. The map offers different numbers of lanes, between which a vehicle in the simulation can or in some cases must change in order to continue driving, for instance in the lower right of the figure, where an obstacle on the one-way street blocks one lane. The map also provides areas where other TP have a higher reservation right than motor vehicles or different types of TP have equal rights. These areas are the crosswalk (on the left in the figure), bicycle boulevard (on the right), traffic-calmed area (on the upper right) and also on the upper right a lane on which rail traffic has higher right of way depending on the traffic light signal.

In a simulation environment based on this synthetic map, routes through this road network are later created that define a sequence of segments along which the ego vehicle should move without fulfilling the falsification criteria. Moreover, the ego vehicle can later be placed at specific positions to test the different falsification criteria. However, in order to train a behavior planner to choose the best possible action in all possible scenarios, a training data set must be available that covers all these sceneries, i.e. all combinations of elements and attributes provided by BSSD. This is difficult to achieve with a manually

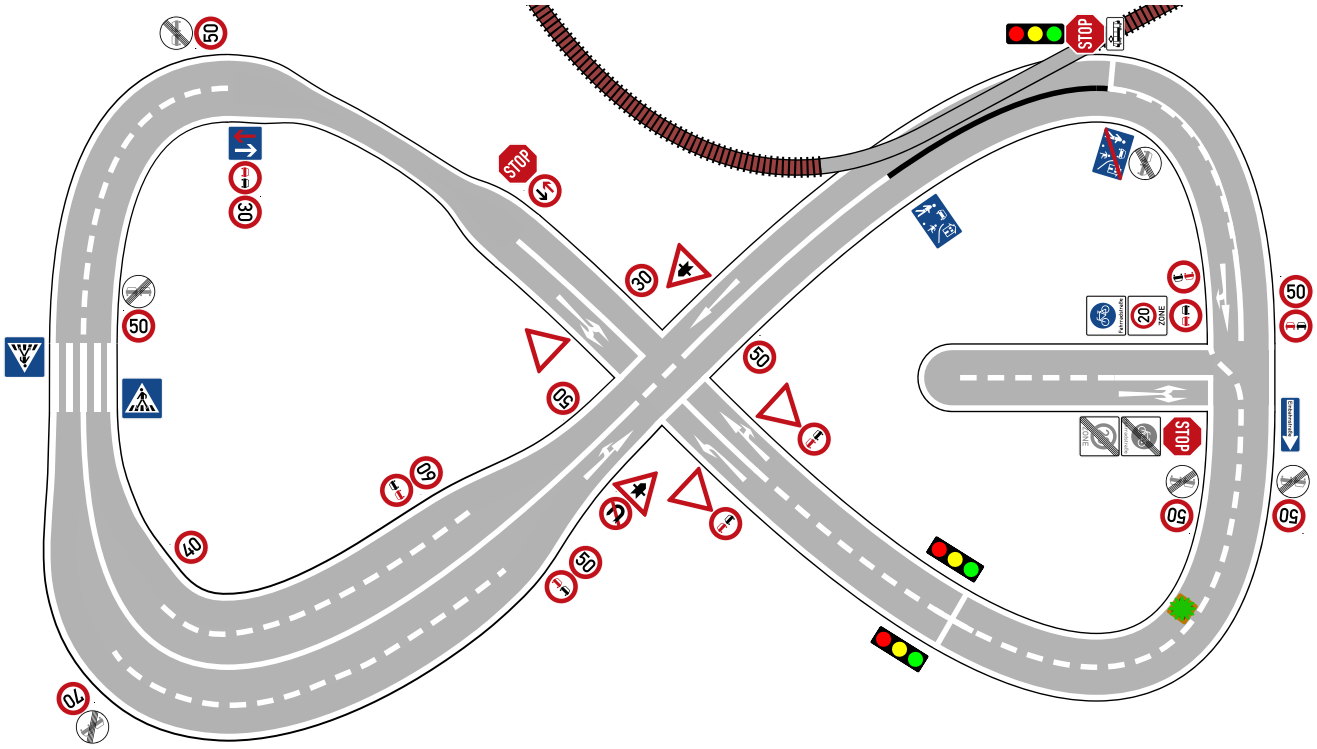


Figure 4-2: Demo Map.

created demo map. Therefore, algorithms could be developed that specifically create such sceneries. However, this is already beyond the scope of this thesis.

Since figure 4-2 may give the impression that a geometric arrangement of the ways and segments could be considered for modeling, it is mentioned here that BSSD does not model the position and spatial course of the ways. Not even the length of individual segments is modeled. However, at least the latter is necessary to create a reasonable simulation environment for this task. Therefore, the longitudinal dimensions of the individual segments were stored separately to BSSD.

4.3 Convert Demo Map to BSSD

After the demo map was first created graphically, it was converted to BSSD. BSSD specifies modeling in XML format. The generation of such a model by hand carries the risk that errors of various kinds may occur during the process. To limit this risk to some extent, a different way was chosen for BSSD conversion of the demo map.

Since at this point the choice of programming language for the implementation of the behavior planner and its simulation environment had already fallen on Python, the approach arose to write BSSD not as an XML file in a text editor, but as an initialization of an object of the class BSSD in a Python Integrated Development Environment (IDE).

For this purpose, a Python package named `pybssd` was created, which covers all information that BSSD can model with the help of subpackages and classes contained therein. `pybssd` uses with few exceptions the hierarchical structure of the original version of BSSD in XML format. An exception of this is for example the renouncement of the `<left>` and `<right>` elements within `<segment>` elements, since this

information is redundant to the ID of the lanes. (See subsubsection 4.1.1). To create an object in Python in the whole information depth of BSSD, it is possible to write a long initialization statement of the main class `BSSD`, in which also all necessary child objects of the classes like `Way`, `Node`, `Segment` and many more are initialized.

Using Python instead of XML offers several advantages: When using an IDE with appropriate functionality, it can offer completion suggestions to the author during writing. If a parameter is omitted or used incorrectly when initializing an object, the IDE can mark this as an error to alert the author. `pybssd` uses enum objects rather than strings for attributes that allow only a fixed list of values, to prevent typographical mistakes. In addition, an error message is generated when executing the initialization if the specified ID are not unique in the prescribed ranges. Text Highlighting and Auto Indent are other useful features, but some text editors also support them with XML.

In `pybssd` the method `check_integrity()` was implemented, which checks after initialization of the object in addition some information whether these are consistent in itself. Thus e.g. it is examined whether the nodes indicated for the ways are existing and vice versa. Only certain errors are checked, which were estimated as probable during the conversion of the demo map and whose examination could be implemented with appropriate expenditure. This method could certainly be extended outside the scope of this thesis.

If a BSSD element contains several child elements of the same type, they are kept in `pybssd` in a dictionary, i.e. as key-value pairs. The keys are the ID of the objects. During the conversion of the demo map it proved to be very helpful repeatedly with Python in the terminal to load the at that time still incomplete, but executable version of the map and to let output relevant parts for this moment. Therefore a method was developed to output the whole object or parts of it as text in the terminal. As an example, figure 4-3 shows a terminal where in an interactive python session the scenery description outputs the lane with the ID -1 from the first segment of the third way using just this line of code: `bssd.ways[3].segments[1].lanes[-1]`.

However, since the BSSD specification requires an XML version, `pybssd` can be used to export a created object as an XML file. During the export, exact compliance with the specification is again ensured, for example by grouping lanes using the `<left>` and `<right>` elements. In addition, `pybssd` can import XML files. The import and export function was tested by exporting, importing and re-exporting the converted demo map and checking the two Python objects as well as the resulting XML files for equality.

During the conversion of the demo map, a question became visible that BSSD specification leaves unanswered so far: How should the direction for forward/backward behavior be defined, if a way leads from a node back to that node? This problem exists for the way shown in Figure 4-2 on the left. To solve it, an auxiliary node was introduced which splits this way into two ways. Thus, no way leads directly back to its starting node. This auxiliary node is a good example of the use of an infinitesimal node, since it is not a real intersection and thus is also area-less. Accordingly, the connecting way would have no length. However, because this would have required more development effort due to appropriate exception handling, the node was not represented as infinitesimal and the connecting way has the length of $5\,\mathrm{m}$.

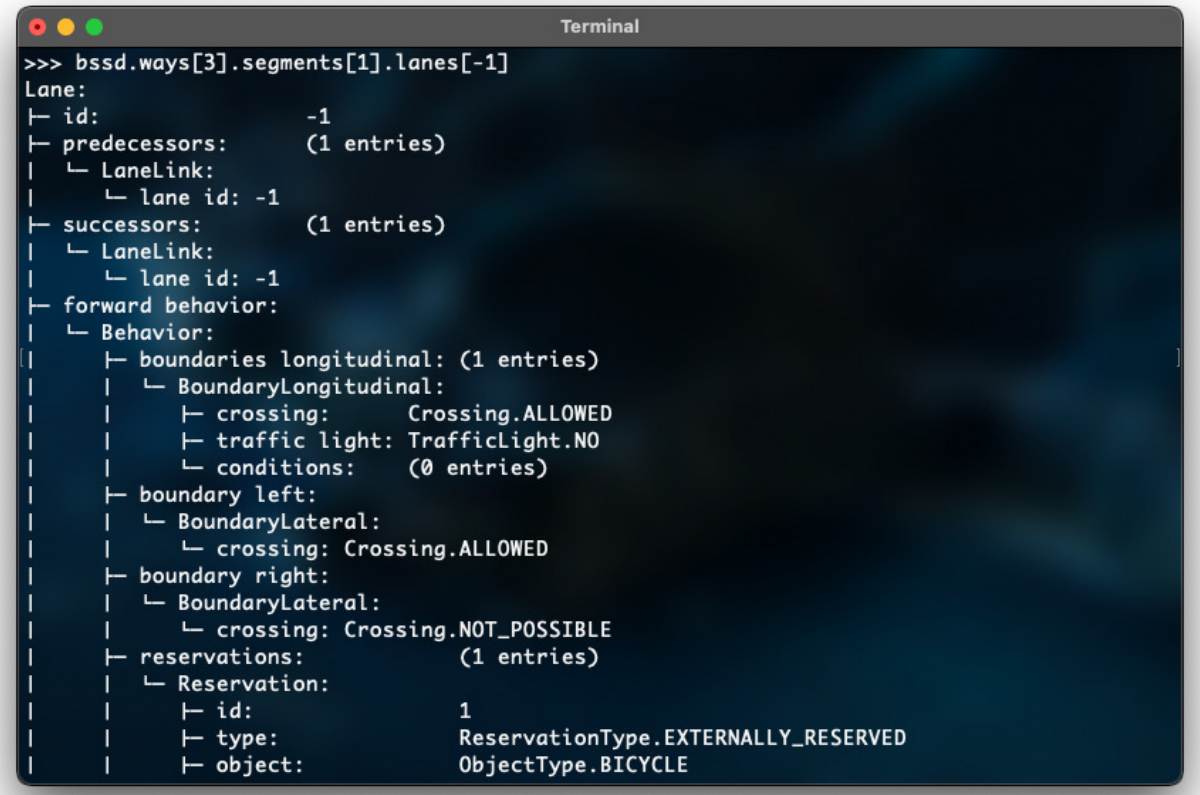


Figure 4-3: Example of using pybssd: Output of the scenery description of a particular lane in a terminal.

Figure 4-4 shows the 5 main ways and 4 nodes of the demo map with their assigned ID, so also the auxiliary node with ID 4. Only 2 of the 16 connecting ways within the nodes are represented their ID. Main ways are labeled in yellow, connecting ways in blue, segments in red, lanes in black und nodes in white.

At the time the map was created and converted, the BSSD specification for reservation link did not yet include the optional `target` attributes. These have not yet been included in the demo map. However, `pybssd` can already include this information. Also the demo map does not contain any infinitesimal nodes, even if e.g. the auxiliary node would be predestined for this. The root cause is that with the implementation of the behavior planner for nodes without connecting ways an exception handling would have to take place, which does not stand in the expenditure benefit relationship and this attribute for nodes has no effect on the formalization of the falsification criteria.

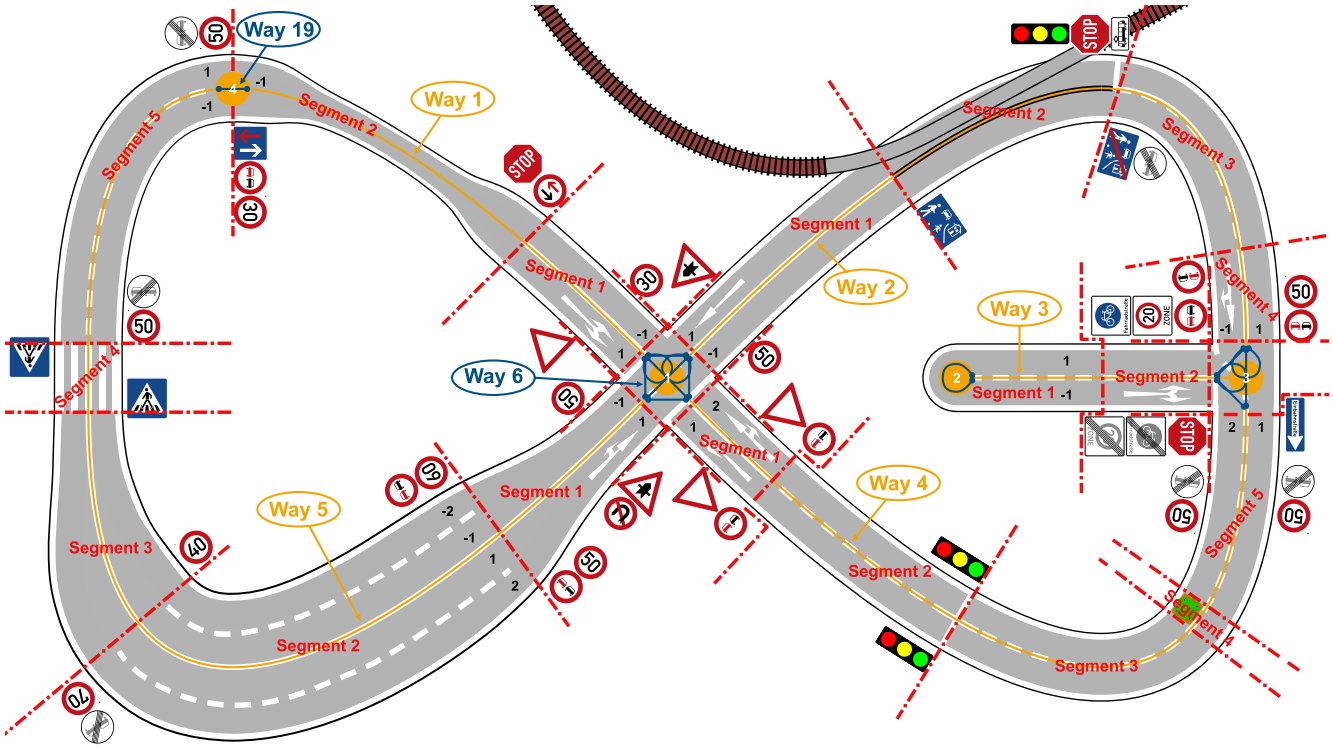


Figure 4-4: Demo map with annotations shows identifies for ways, segments, lanes and nodes, separators for segments and indications for connecting ways within nodes.

5 Complexity Reduction

To simplify the formalizations and implementation, the assumption is made that the ego vehicle moves only along a given sequence of segments. It includes all segments of all ways ego must travel along the route to the destination. The foundation for this simplification is the reality-based reasoning that ego follows a route given by the navigation system. Within segments, ego must move accordingly on the lanes that will take it to the next segment in the sequence. If it leaves a segment via a wrong lane, e.g. would turn wrong at an intersection, this is evaluated as if it had violated a boundary regularity. Regarding to the lane change detection in definition 6.17 the assumption is made that no vehicle is wider than an area.

The ego vehicle can only move forward on its segment sequence in the longitudinal direction, i.e. toward the destination. Reversing or turning is not possible. Thus, for each segment of the sequence, it is clear whether the ego vehicle will traverse it forward or backward in the reference direction, and which of the two direction-specific behavioral specifications applies to the ego vehicle.

For the formalizations, time is used as a discrete variable. The reason for this is that according to the current state of the art such NN and thus also corresponding simulation environments work with time steps, so also the software project created here.

BSSD includes the possibility of a detailed description of the maximum permitted speed depending on weather conditions, day and time. This will not be considered further in this paper. In the following, only one maximum permitted speed will be used. As already mentioned in subsection 4.3, the BSSD specification was extended by the optional target attributes for reservation link only during the processing of the thesis. Therefore, no further attention was paid to these in the further process of the elaboration. Special behavioral requirements to be followed when an area is equally-reserved for multiple TP, like zipper merge, should not receive further attention in this work.

5.1 Complexity Reduction for Implementation

For the implementation, further effort is made to reduce the complexity. For example, the resolution of the space in lateral extent is limited to the lanes. Thus, within a lane, the lateral position of an ego vehicle cannot be further determined. Two TP cannot be next to each other within a lane. In addition, vertical expansion was omitted. Accordingly, there are no bridges or tunnels.

Following Hoel et al.[43] the action space, which includes the longitudinal and lateral guidance of the ego vehicle, is limited to a reasonable extent. (See also subsection 3.2.) To this end, functionalities of certain driver assistance systems are used, which then take over the lower level operational decision making. For longitudinal guidance, the functionality of a Cruise Control (CC) is simulated. Thus, the behavior planner only specifies a target speed, which is maintained by CC. For lateral guidance of the ego vehicle, the functionalities of active assistants for keeping and changing lanes are used. Thus the selection of possible lateral actions is limited to change to the left, change to the right or no change. These three actions are also sufficient to drive through intersections as specified. By selecting these restrictions in the action space, a reality-based complexity reduction is made, since it is quite conceivable that AV will make use of assistance systems that are already in use today and thus well tested.

[43] Hoel, C.-J. et al.: Planning and Deep Reinforcement Learning (2020).

6 Formalizing of Falsification Criteria

A distinctive feature of BSSD is that ways of the road network are segmented in such a way that the required behavior is homogeneous in each segment. Just as a way is divided longitudinally into segments, a segment is divided laterally into lanes. To detect violations of the traffic rules that are in effect at the location of the ego vehicle, the action selected by the behavior planner is checked at runtime at each time step using the falsification criteria. For this purpose, the behavioral semantic descriptions of the segment lane subunit, also called area, on which the ego vehicle is located are analyzed. However, the ego vehicle may be on more than one segment and/or lane at a time. Accordingly, the rules of all areas on which the ego vehicle is located must be observed. In addition, the behavioral specifications must also be checked for compliance by the areas that the ego vehicle has traveled since the last time step, even if it is no longer on them at the current time step.

Unlike the papers presented in subsection 3.3, in this thesis the formalization of falsification criteria as behavioral boundaries for automated vehicles is primarily not based on the set of rules of a particular country. In this case, the falsification criteria are derived from the behavioral specifications as they can be described using BSSD. Thus, the BSSD specification[44] is the first source of reference. However, as mentioned in subsection 4.1, the BSSD should describe the constraints coming from StVO. At the points where further clarification is required, the StVO is therefore consulted in the second step and then, if necessary, further sources will be included.

As described in section 5, the ego vehicle moves along a segment sequence to its destination. For the following formalizations, only the direction-specific behavioral constraints that apply to the ego vehicle are considered, respectively to other TP if this consideration is necessary for a falsification criterion of the ego vehicle.

In the theoretical case that ego's route is straight, i.e. without curves, a geometric point

$$ {}^{\mathrm{w}}P = \begin{pmatrix} {}^{\mathrm{w}}x \\ {}^{\mathrm{w}}y \\ {}^{\mathrm{w}}z \end{pmatrix} $$

with respect to the world coordinate system S_{w} can be transformed into a point

$$ {}^{\mathrm{q}}P = \begin{pmatrix} {}^{\mathrm{q}}x \\ {}^{\mathrm{q}}y \end{pmatrix} $$

with respect to ego's segment sequence coordinate system S_{q} using transformation matrix ${}^{\mathrm{w}}T_{\mathrm{q}}$.[45]

$$ \underbrace{{}^{\mathrm{w}}\hat{P}}_{\in\mathbb{R}^4} = \begin{pmatrix} {}^{\mathrm{w}}P \\ \hline 1 \end{pmatrix} = \underbrace{{}^{\mathrm{w}}T_{\mathrm{q}}}_{\in\mathbb{R}^{4\times 3}} \cdot \begin{pmatrix} {}^{\mathrm{q}}P \\ \hline 1 \end{pmatrix} = {}^{\mathrm{w}}T_{\mathrm{q}} \cdot \underbrace{{}^{\mathrm{q}}\hat{P}}_{\in\mathbb{R}^3} \tag{6-1} $$

[44] Glatzki, F.; Lippert, M.: BSSD Specification (2021).

[45] Stryk, O. v.: Grundlagen der Robotik (2018), p. 24.

The ego vehicle $\mathfrak{e}$ has a speed ${}^{\mathrm{q}}v_{\mathfrak{e}} > 0$ with respect to S_{q} if it follows the route (travels in the direction of the destination). Another TP $\mathfrak{p}$ on ego vehicle's segment sequence has a speed ${}^{\mathrm{q}}v_{\mathfrak{p}} > 0$ with respect to S_{q} if it travels in the same direction as the ego vehicle does and ${}^{\mathrm{q}}v_{\mathfrak{p}} < 0$ if it travels in the opposite direction.

BSSD provides a thematic breakdown of behavioral specifications through the various XML elements within `<forwardBehavior>` or `<backwardBehavior>` (see subsubsection 4.1.1). This structure is predominantly adhered to in the formalization of falsification criteria. The falsification criteria collision and obstruction, which are derived from the general behavior rules (see subsubsection 4.1.3), are placed in front.

In the following subsections the individual falsification criteria are presented in a strict form – as definitions and mostly as formulas. For the criteria that are very complex and could not be implemented in this way either, a full presentation using formulas was omitted. To derive individual falsification criteria, the top-down approach of Karimi and Duggirala[46] was followed. Thus, each criterion was extracted from the BSSD specification, its logical form identified, and finally formalized in HOL. Later in the thesis, all falsification criteria were formalized again in the programming language Python in a way that fits the RL behavior planner implemented here (see also subsection 7.2).

6.1 Falsification Criterion for Collision

According to the first general behavior rule of the BSSD specification, a TP should avoid foreseeable collisions both within an area and during the change between areas, independent of the reservation type of the area.[47] (See also subsubsection 4.1.3.) This requires knowledge about the shapes and sizes of objects, including the TP, and their spatial positions. If all these details are available, it is possible to check at any time whether the ego vehicle has collided with another object.

For collision detection, simulative solutions are often used in which the state of motion of the TP is predicted in discrete time steps.[48a] Among others, geometric approaches are applied, which take into account the spatial extent of the vehicles, for example in the form of a rectangle, and aim at detecting overlaps between vehicles.[48b] These rectangles are also called bounding boxes. This principle can also be applied in three-dimensional space using cuboids.

However, whether it is a foreseeable collision is much more difficult to determine, partly because the BSSD specification does not define what exactly is meant by *foreseeable*. The approach described here relies on the use of probabilistics. Here, a lower limit for foreseeability is determined, i.e. a value for the probability that a collision will happen. If the probability is higher than the threshold, then such a collision should have been prevented. Furthermore, a time specification is important, because this is essential for the predictability. For example, a few fractions of a second before a collision will happen, the probability of this event will always be very high. Accordingly, a specification is required for the minimum time gap to the collision at which the predictability value is compared with the threshold value.

[46] Karimi, A.; Duggirala, P. S.: Formalizing traffic rules for uncontrolled intersections (2020).

[47] Glatzki, F.; Lippert, M.: BSSD Specification (2021), p. 6.

[48] Winner, H. et al.: Handbuch Fahrerassistenzsysteme (2015). a: pp. 705-706; b: p. 706.

To determine this falsification criterion, reference is made to Time To Critical Collision Probability (TTCCP), which Schreier [49a] introduces in his dissertation to estimate the progression of the traffic as well as its criticality. „TTCCP criticality metric is calculated as the necessary prediction time until the probability of this collision event exceeds a certain value. It is therefore a time metric analogous to [Time To Collision], which, however, takes many possible future traffic scene evolutions into account.“[49b] Unlike Schreier, here not several movements per time step have to be considered, but only the one that led to the state of the next, already known time step. Practically, this could be solved in such a way that after a collision, time step by time step for a defined period of time, the probability of the movements of both objects that they have made in this time step is calculated.

Definition 6.1 (Collision – Part 1 of 4)**.** The falsification criterion for collision $F_{\mathrm{c}}\left(t\right)$ is fulfilled at time t if at this time a collision $C\left(\mathfrak{e},\mathfrak{o},t\right)$ of the ego vehicle $\mathfrak{e}$ with an object $\mathfrak{o}$ of all other objects in the simulation environment $\mathfrak{O}\backslash\{\mathfrak{e}\}$ occurs and the foreseeability of this collision $p\left(C\left(\mathfrak{e},\mathfrak{o},t\right)\middle|\Theta_{\mathfrak{e}}(t-\delta_{\mathrm{c,last}})\right)$ at time $t-\delta_{\mathrm{c,last}}$, which is by the duration $\delta_{\mathrm{c,last}}$ before the collision time t, was greater than $p_{\mathrm{c,last,min}}$[50].

$$F_{\mathrm{c}}\left(t\right) \leftarrow \bigvee_{\mathfrak{o}\in\mathfrak{O}\backslash\{\mathfrak{e}\}} \left(C\left(\mathfrak{e},\mathfrak{o},t\right) \wedge p\left(C\left(\mathfrak{e},\mathfrak{o},t\right)\middle|\Theta_{\mathfrak{e}}(t-\delta_{\mathrm{c,last}})\right) > p_{\mathrm{c,last,min}}\right) \tag{6-2}$$

To determine a collision, the principle of bounding boxes is used for simplification.

Definition 6.2 (Collision – Part 2 of 4)**.** $C\left(\mathfrak{e},\mathfrak{o},t\right)$ is fulfilled at time t, when the bounding boxes of $\mathfrak{e}$ and $\mathfrak{o}$ touch or intersect. At time t a cubic bounding box of a TP extends from

$${}^{\mathrm{w}}P_{\mathrm{min}}(t) = \begin{pmatrix} {}^{\mathrm{w}}x_{\mathrm{min}}(t) \\ {}^{\mathrm{w}}y_{\mathrm{min}}(t) \\ {}^{\mathrm{w}}z_{\mathrm{min}}(t) \end{pmatrix}$$

to

$${}^{\mathrm{w}}P_{\mathrm{max}}(t) = \begin{pmatrix} {}^{\mathrm{w}}x_{\mathrm{max}}(t) \\ {}^{\mathrm{w}}y_{\mathrm{max}}(t) \\ {}^{\mathrm{w}}z_{\mathrm{max}}(t) \end{pmatrix}$$

with respect to the world coordinate system S_{w}.

The formula 6-3 shows how a collision detection can be solved if the bounding boxes of the two TP are aligned parallel to coordinate axes.

$$\begin{aligned} C\left(\mathfrak{e},\mathfrak{o},t\right) \leftarrow\ & {}^{\mathrm{w}}x_{\mathfrak{e},\mathrm{min}}(t) \leq {}^{\mathrm{w}}x_{\mathfrak{o},\mathrm{max}}(t) \wedge {}^{\mathrm{w}}x_{\mathfrak{e},\mathrm{max}}(t) \geq {}^{\mathrm{w}}x_{\mathfrak{o},\mathrm{min}}(t) \wedge \\ & {}^{\mathrm{w}}y_{\mathfrak{e},\mathrm{min}}(t) \leq {}^{\mathrm{w}}y_{\mathfrak{o},\mathrm{max}}(t) \wedge {}^{\mathrm{w}}y_{\mathfrak{e},\mathrm{max}}(t) \geq {}^{\mathrm{w}}y_{\mathfrak{o},\mathrm{min}}(t) \wedge \\ & {}^{\mathrm{w}}z_{\mathfrak{e},\mathrm{min}}(t) \leq {}^{\mathrm{w}}z_{\mathfrak{o},\mathrm{max}}(t) \wedge {}^{\mathrm{w}}z_{\mathfrak{e},\mathrm{max}}(t) \geq {}^{\mathrm{w}}z_{\mathfrak{o},\mathrm{min}}(t) \end{aligned} \tag{6-3}$$

[49] Schreier, M.: Diss., Environment Representation, Prediction, Criticality Assessment (2015). a: -; b: pp. 150-151.

[50] This is explained later in this subsection.

In terms of the world coordinate system, this is not a perfect assumption, since the bounding box is proportionally large if its TP lies diagonally in it. A better algorithm can be used here in the future. However, since for the following implementation of the falsification criteria in the scope of this thesis a collision is only detected within segments and the TP are always aligned parallel to the axes of the segment sequence coordinate system, the formula 6-4 is used here.

$$\begin{aligned} C\left(\mathfrak{e},\mathfrak{o},t\right) \leftarrow\ & {}^{\mathrm{q}}x_{\mathfrak{e},\min}(t) \leq {}^{\mathrm{q}}x_{\mathfrak{o},\max}(t) \wedge {}^{\mathrm{q}}x_{\mathfrak{e},\max}(t) \geq {}^{\mathrm{q}}x_{\mathfrak{o},\min}(t) \wedge \\ & {}^{\mathrm{q}}y_{\mathfrak{e},\min}(t) \leq {}^{\mathrm{q}}y_{\mathfrak{o},\max}(t) \wedge {}^{\mathrm{q}}y_{\mathfrak{e},\max}(t) \geq {}^{\mathrm{q}}y_{\mathfrak{o},\min}(t) \end{aligned} \tag{6-4}$$

Definition 6.3 (Collision – Part 3 of 4)**.** $p\left(C\left(\mathfrak{e},\mathfrak{o},t\right)\middle|\Theta_{\mathfrak{e}}(t-\delta_{\mathrm{c,last}})\right)$ is the total probability that starting from $t-\delta_{\mathrm{c,last}}$ in each time step $\tilde{t}$ until t the movements occurred as they actually did. $\Theta_{\mathfrak{e}}(t-\delta_{\mathrm{c,last}})$ is the probability model of $\mathfrak{e}$ for predicting its environment, made based on information from time span $\left[0, t-\delta_{\mathrm{c,last}}\right]$. $t-\delta_{\mathrm{c,last}}$ is the latest time when $\mathfrak{e}$ could have performed a maneuver that would have prevented this collision.

$t-\delta_{\mathrm{c,last}}$ was chosen to keep the observation period small. A later time at which the foreseeability is to be compared is not useful, because then the collision is inevitable anyway. Following the requirements for a human driver, to determine the foreseeability, all information available for the ego vehicle up to $t-\delta_{\mathrm{c,last}}$ must be added to $\Theta_{\mathfrak{e}}(t-\delta_{\mathrm{c,last}})$, including the weather, which has an influence on the condition of the road surface, the estimation of the driving behavior of the collision participant and further more. As a lower predictability limit $p_{\mathrm{c,last,min}}$, the value was initially set at $50\,\%$. At a lower value, the event can no longer be considered *foreseeable*. For a more precise definition of the limit, further research must be carried out or it must even be determined in an empirical perception study at what point a collision is generally classified as foreseeable.

In this manner, the falsification criterion is formalized to fit the requirement from the BSSD specification. However, in this way it is still not sufficient for the application of a RL behavior planner, since the planner would also be penalized if the collision was foreseeable, but the ego vehicle had no possibility to escape the collision. Example: the ego vehicle is at the end of a traffic jam, recognizes that another TP approaches from behind with high speed and a collision is probable, but ego vehicle has no more chance to do something about it. In this case, a penalty would be unwarranted.

The Responsibility-Sensitive Safety developed by the company Mobileye includes formalizations for safe driving of AV. Regarding accidents, they prescribe: ”If you can avoid an accident without causing another one, you must do it”[51] Following this approach, the definition of the falsification criterion is extended.

Definition 6.4 (Collision – Part 4 of 4)**.** The falsification criterion for collision $F_{\mathrm{c}}\left(t\right)$ is only fulfilled if in addition $\mathfrak{e}$ had the possibility to avoid $C\left(\mathfrak{e},\mathfrak{o},t\right)$ without creating another collision $C\left(\overline{\mathfrak{e}},\cdot,\cdot\right)$.

Note: Since foreseeability and avoidability are not considered further in the implementation work of this thesis and because in this respect the BSSD specification will probably change again, the definitions and formulas are not extended further in this sense.

[51] Shalev-Shwartz, S. et al.: Formal Model of Safe Self-driving Cars (2018), p. 15.

6.2 Falsification Criterion for Obstruction

In two places in the BSSD specification, depending on the relations of the reservation rights of the TP, it is described in which cases an obstruction is not permitted. The reservation right of a TP for an area is described on the basis of the <reservation> elements of this area. The first place is the second general behavior rule, which states that when entering a new area, TP with the same reservation right for that area must not be obstructed.[52] (See also subsubsection 4.1.3.) Second, TP that have a higher reservation right for that area must not be obstructed at any time within this area.

Definition 6.5 (Obstruction – Part 1 of 4)**.** The falsification criterion for obstruction $F_{\mathrm{o}}(t)$ is fulfilled at time t if the ego vehicle $\mathfrak{e}$ at that time is an obstruction $O(\mathfrak{e}, \mathfrak{p}, t)$ for at least one TP $\mathfrak{p}$ of all other TP in the simulation environment $\mathfrak{P} \backslash \{\mathfrak{e}\}$ that has

- a higher reservation right $r(\mathfrak{p}, \mathfrak{a}) > r(\mathfrak{e}, \mathfrak{a})$ ($r(\cdot, \cdot) \in \{0, 1\}$) for at least one area $\mathfrak{a}$ of all areas $\mathfrak{A}_{\mathfrak{e}}(t)$ on which $\mathfrak{e}$ is completely or partially located at time t or was located during the movement from the previous time step $t-1$ to t or
- a equal reservation right $r(\mathfrak{p}, \mathfrak{a}) = r(\mathfrak{e}, \mathfrak{a})$ and $\mathfrak{e}$ enters $\mathfrak{a}$ at this time ($\mathfrak{a} \notin \mathfrak{A}_{\mathfrak{e}}(t-1)$).

$$F_{\mathrm{o}}(t) \leftarrow \bigvee_{\mathfrak{a} \in \mathfrak{A}_{\mathfrak{e}}(t)} \bigvee_{\mathfrak{p} \in \mathfrak{P} \backslash \{\mathfrak{e}\}} \Big(O(\mathfrak{e}, \mathfrak{p}, t) \wedge \Big(r(\mathfrak{p}, \mathfrak{a}) > r(\mathfrak{e}, \mathfrak{a}) \vee \big(r(\mathfrak{p}, \mathfrak{a}) = r(\mathfrak{e}, \mathfrak{a}) \wedge \mathfrak{a} \notin \mathfrak{A}_{\mathfrak{e}}(t-1) \big) \Big) \Big) \tag{6-5}$$

Definition 6.6 (Obstruction – Part 2 of 4)**.** A TP $\mathfrak{p}$ has no reservation right ($r(\mathfrak{p}, \mathfrak{a}) = 0$) for $\mathfrak{a}$ if

- there is no <reservation> element $\mathfrak{r}_{\mathfrak{p}}$ of all <reservation> elements $\mathfrak{R}_{\mathfrak{a}}$ of $\mathfrak{a}$ that matches $\mathfrak{p}$ ($\mathfrak{r}_{\mathfrak{p}} \notin \mathfrak{R}_{\mathfrak{a}}$) with respect to the attributes `object`, `trafficLightActive` and one of possibly several <link> elements of <reservation> or
- there is one ($\mathfrak{r}_{\mathfrak{p}} \in \mathfrak{R}_{\mathfrak{a}}$) but it has the attribute value pair `type="externally-reserved"`.

$\mathfrak{p}$ has a reservation right ($r(\mathfrak{p}, \mathfrak{a}) = 1$) for $\mathfrak{a}$ if there is a <reservation> element ($\mathfrak{r}_{\mathfrak{p}} \in \mathfrak{R}_{\mathfrak{a}}$) of $\mathfrak{a}$ that matches $\mathfrak{p}$ and it has the value `"own-reserved"` or `"equally-reserved"` for attribute `type`.

$$r(\mathfrak{p}, \mathfrak{a}) := \begin{cases} 0, & \text{if } \mathfrak{r}_{\mathfrak{p}} \notin \mathfrak{R}_{\mathfrak{a}} \vee \mathrm{type}(\mathfrak{r}_x) = \texttt{"externally-reserved"} \\ 1, & \text{otherwise} \end{cases} \tag{6-6}$$

But neither the BSSD specification nor the StVO define what obstruction is. Police Specialist Handbook (ger.: Polizei-Fach-Handbuch) (PolFHa) provides police officers in training and in daily practice descriptions of legal texts, so it also offers a definition of obstruction in road traffic. ”Obstruction is any interference with traffic participation that is within the scope of public use, whether by interfering with movement (forcing vehicles to swerve, brake, or stop), or by preventing or impeding further movement

[52] Glatzki, F.; Lippert, M.: BSSD Specification (2021), p. 6.

(immobilizing or blocking vehicles)."[53] It further states, however, that only avoidable obstructions are prohibited. "Avoidable is an obstruction […] if it cannot be expected of the person concerned according to the circumstances. To decide this, it is necessary to weigh the interests of the person obstructing against those of the person obstructed."[53] The BSSD, however, does not provide for such a restriction at the current time. Therefore, and because consideration of it would significantly increase the complexity of the formalization work, this restriction will not be pursued further within this thesis.

In addition to the scenarios that can be easily classified as obstruction, there are also those in which the obstruction classification depends on how an affected TP assesses the further behavior of other TP. Thus, there is also the obstruction type of deception or irritation. The Uniform Federal Catalogue of Offences for Road Traffic Offences (ger.: Bundeseinheitlicher Tatbestandskatalog für Straßenverkehrsordnungswidrigkeiten) (BT-KAT-OWI) includes the defined offenses in road traffic, which are derived from the regulations such as the StVO. The offense with the number $108\,100$ describes a scenario in which a TP does not approach the right-of-way road at a moderate speed and thereby irritates the person with the right of way without actually intersecting its route.[54]

Likewise, an obstruction can already occur, if a TP is forced to act by an announcement of intention by another TP. Example: TP a is driving on the leftmost lane of a multi-lane freeway. TP b is driving in front of him on the lane to the right at a significantly lower speed and suddenly activates the left turn signal. a assesses that in such a way that b immediately changes into a's lane and, due to the small distance, feels compelled to reduce its speed in order to properly deal with a potential danger that has become probable simply because b announced its intention without actually carrying it out so far.

These two cases show that obstruction also has a subjective component. This must be taken into account when defining obstruction. In the following, however, only obstructions that arise due to driving behavior will be considered. Obstructions caused by intentions, dazzling with headlights or noise are initially excluded for the simulation.

Consistent with the definition of obstruction previously cited from [53], it is detected by a deviation from the original trajectory. Therefore, two trajectories are calculated for each TP at each time step.

Definition 6.7 (Obstruction – Part 3 of 4)**.** The *theoretical trajectory* $\left\{ P_{\mathfrak{p}}(\tilde{t}) \in \mathbb{R}^3 : t \leq \tilde{t} \leq t_{\mathrm{f}} \right\}$ (t_{f}: final time)[55] describes how $\mathfrak{p}$ would move in the future if it only pay attention to other TP to which it must give priority. The *actual trajectory* $\left\{ P'_{\mathfrak{p}}(\tilde{t}) \in \mathbb{R}^3 : t \leq \tilde{t} \leq t_{\mathrm{f}} \right\}$ takes all TP into account. The *actual trajectory* is derived from *theoretical trajectory* due to a reaction of $\mathfrak{p}$ because in $\mathfrak{p}$'s estimation based on $\Theta_{\mathfrak{p}}(t)$ a collision $C\left(\mathfrak{e}, \mathfrak{p}, t + \delta_{\mathrm{c}}\right)$ of $\mathfrak{e}$ and $\mathfrak{p}$ at time $t + \delta_{\mathrm{c}}$, which is by the duration δ_{c} in the future, will otherwise occur with probability $p\left(C\left(\mathfrak{e}, \mathfrak{p}, t + \delta_{\mathrm{c}}\right) \middle| \Theta_{\mathfrak{p}}(t)\right) > p_{\mathrm{c,min}}$.[56]

Using both trajectories to detect an obstruction due to deviation from each other is poorly implementable on real roads with human TP, but quite implementable in a simulation for an RL behavior planner, since the simulation environment can be designed for this use case. It is important that the cause for a deviation can

[53] Verlag Deutsche Polizeiliteratur GmbH: Erläuterungen zur Straßenverkehrsordnung (2019), p. 5.

[54] Kraftfahrt-Bundesamt: BT-KAT-OWI (2020), p. 73/0.

[55] Stryk, O. v.: Grundlagen der Robotik (2018), p. 56.

[56] See also definitions 6.1, 6.2 and 6.3. It may be necessary that $p_{\mathrm{c,min}} \neq p_{\mathrm{c,last,min}}$.

be named, since the falsification criterion may only be fulfilled if the ego vehicle is the cause. Example: If another TP with a higher reservation right than the ego vehicle brakes in close proximity to it, but the cause of the braking is a soccer ball rolling onto the road, the planner shall not be penalized for this.

However, when considering real traffic scenarios, provoking a slight, short delay, for example, cannot yet be considered an obstruction. The StVO also only mentions a *substantial* obstruction, which must be avoided as a TP giving right of way.[57] The PolFHa adds: "It is not sufficient for a violation of the right of way if the participant giving the right of way only had to take his foot off the accelerator for a short time"[58] and refers to a court decision[59]. Otherwise, provoking a slight but longer braking is definitely an obstruction. Therefore, a measure of the severity of an obstruction must be found. This should be independent of whether the disabled TP brakes early and lightly or late and heavily. Furthermore, a threshold value of this measure is wanted, above which a non-legitimate obstruction by the ego vehicle leads to a penalty.

One such measure is the spatial deviation of two points on theoretical and realistic trajectories at the same time in the future. Figure 6-1 shows three scenarios. On the one hand, the green vehicle $\mathfrak{p}$ on the left lane is not obstructed, in the other two scenarios it is obstructed by the ego vehicle $\mathfrak{e}$ (red) changing from the right to the left lane. The trajectories of the TP are drawn with the positions of the vehicle origins at different times. This emphasizes the deviation from the *theoretical trajectory* of $\mathfrak{p}$ without obstruction and the *actual trajectories* with obstruction. In the middle scenario $\mathfrak{p}$ reacts early with a slight delay, and in the right scenario $\mathfrak{p}$ reacts late with a strong delay.

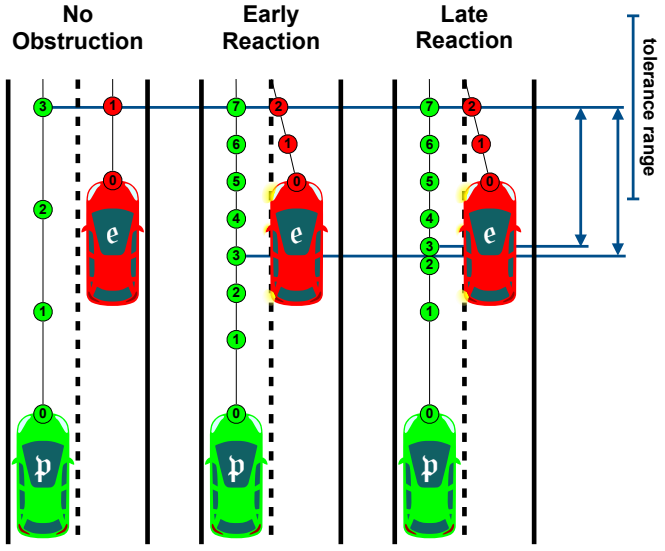


Figure 6-1: Comparison of three scenarios: no obstruction (1), obstruction by ego vehicle $\mathfrak{e}$ with early response of the other TP $\mathfrak{p}$ (2), and late response (3). Also the longitudinal tolerance range is plotted here as well as trajectories of $\mathfrak{e}$ and $\mathfrak{p}$ using position marks at $t = 0$, $t = 1$, ...

Definition 6.8 (Obstruction – Part 4 of 4)**.** $O(\mathfrak{e}, \mathfrak{p}, t)$ is fulfilled at time t if at $t + \delta_\mathrm{o}$ the position $P'_\mathfrak{p}(t + \delta_\mathrm{o})$ of $\mathfrak{p}$ deviates from $P_\mathfrak{p}(t + \delta_\mathrm{o})$ more than $d_{\mathrm{o,lon}}$ in longitudinal direction or $d_{\mathrm{o,lat}}$ in lateral direction.

[57] Bundesministerium der Justiz und für Verbraucherschutz: Straßenverkehrs-Ordnung (2013), § 8 (2).

[58] Verlag Deutsche Polizeiliteratur GmbH: Erläuterungen zur Straßenverkehrsordnung (2019), p. 28.

[59] Oberlandesgericht Braunschweig: 2 U 34/90 (1990).

Which value for δ_o is specifically considered as a good basis for calculating the deviation of to positions cannot yet be determined in this thesis. For this purpose, further research or perhaps even empirical perception studies are needed that reveal at what point in time an obstruction is already perceived as such. However, δ_o should not be too small, otherwise a late, severe delay would not be evaluated. For example in the scenes for $t = 1$ in figure 6-1 there is no deviation between *No Obscrution* and *Late Reaction*. Meanwhile, the scenes of $t = 3$ show large differences. The author estimates that this duration must be in the dimension of $5...20\,\mathrm{s}$. Using $t - \delta_o$ as a basis for the calculation of the deviation has the advantage that thereby also the velocities are considered. At high speed, the comparison positions are farther away from the current position than at a low speed. This reflects the obstruction perception in reality: at a high own speed, for example on the freeway, the behavior of other TP is judged and checked for a possible obstruction, which are even farther away along the planned trajectory than at a lower speed. The reason for this is that more distance is covered during the reaction time and a collision avoidance maneuver.

In addition to the point in time for the comparison, the tolerance range within which an obstruction is still acceptable must also be determined. This is also not yet clarified with the completion of this thesis and must also be determined by further research or empirical perception studies. It is possible to select the tolerance range differently in lateral and longitudinal direction, as shown in figure 6-2. The tolerance range must also be speed-dependent, since at higher speeds a deviation from theoretical and actual trajectory is to be provoked with a lower degree of obstruction than at lower speeds.

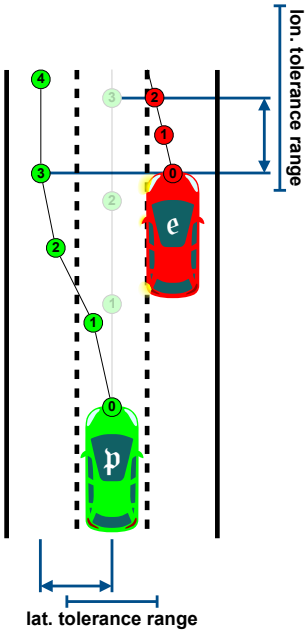


Figure 6-2: Example of an obstruction scenario with braking and evasive reaction. Also the lateral and longitudinal tolerance range is plotted here.

Note: Since such an approach far exceeds the feasibility within this thesis and obstruction had to be implemented in a different way, these definitions and the formulas are not further extended here.

6.3 Falsification Criterion for Longitudinal Boundary Regulation

Definition 6.9 (Lon. Boundary Regulation – Part 1 of 3)**.** The falsification criterion for longitudinal boundary regulation $F_{\text{lon}}(t)$ is fulfilled at time t if ego vehicle $\mathfrak{e}$ longitudinally enters at that time at least one area $\mathfrak{a}$ of all areas $\mathfrak{A}_{\mathfrak{e}}(t)$ on which $\mathfrak{e}$ is completely or partially located at time t or was located during the movement from the previous time step $t-1$ to t and fulfills a longitudinal boundary violation $B_{\text{lon}}(\mathfrak{a}, \mathfrak{b}_{\text{lon}}, t))$ for the suitable ($\text{fit}(\mathfrak{a}, \mathfrak{b}_{\text{lon}}, t)$) longitudinal boundary element (<longitudinal>) $\mathfrak{b}_{\text{lon}}$ of possibly several longitudinal boundary elements $\mathfrak{B}_{\text{lon},\mathfrak{a}}$ of $\mathfrak{a}$. *Longitudinal entering* means that $\mathfrak{a}$ did not belong to any segment $\mathfrak{s}$ of all segments $\mathfrak{S}_{\mathfrak{e}}(t-1)$ on which $\mathfrak{e}$ was completely or partially located at time $t-1$ or during the movement from $t-2$ to $t-1$.

$$F_{\text{lon}}(t) \leftarrow \bigvee_{\mathfrak{a} \in \mathfrak{A}_{\mathfrak{e}}(t)} \left(\begin{array}{ll} \left(\bigwedge_{\mathfrak{s} \in \mathfrak{S}_{\mathfrak{e}}(t-1)} \mathfrak{a} \notin \mathfrak{A}_{\mathfrak{s}} \right) \wedge \\ \bigvee_{\mathfrak{b}_{\text{lon}} \in \mathfrak{B}_{\text{lon},\mathfrak{a}}} \left(\text{fit}(\mathfrak{a}, \mathfrak{b}_{\text{lon}}, t) \wedge B_{\text{lon}}(\mathfrak{a}, \mathfrak{b}_{\text{lon}}, t) \right) \end{array} \right) \tag{6-7}$$

Definition 6.10 (Lon. Boundary Regulation – Part 2 of 3)**.** $\text{fit}(\mathfrak{a}, \mathfrak{b}_{\text{lon}}, t)$ is fulfilled if

- $\mathfrak{a}$ has an active traffic light system and
  $\mathfrak{b}_{\text{lon}}$ has the attribute value pair `trafficLightActive="yes"` or
- $\mathfrak{a}$ has no active traffic light system (it is inactive or there are no traffic lights) and
  $\mathfrak{b}_{\text{lon}}$ has the attribute-value pair `trafficLightActive="no"` (or `trafficLightActive` is missing – `"no"` is the default value).

$$\text{fit}(\mathfrak{a}, \mathfrak{b}_{\text{lon}}, t) \leftarrow \text{hasActiveTrafficLight}(\mathfrak{a}, t) = \text{isTrafficLightActive}(\mathfrak{b}_{\text{lon}}) \tag{6-8}$$

$\text{hasActiveTrafficLight}(\cdot, \cdot)$ and $\text{isTrafficLightActive}(\cdot)$ have boolean return values, e.g. $\{0, 1\}$. The information about whether an area has a traffic light system or not and if so, whether it is active or inactive, is not encoded by the BSSD of a road network, but has to be shared separately. For example, as it was done during the implementation work in this thesis. See subsection 7.1 for this.

Definition 6.11 (Lon. Boundary Regulation – Part 3 of 3)**.** $B_{\text{lon}}(\mathfrak{a}, \mathfrak{b}_{\text{lon}}, t)$ is fulfilled if

- attribute `crossing` of $\mathfrak{b}_{\text{lon}}$ has the value `"prohibited"` or `"not possible"` or
- attribute `crossing` of $\mathfrak{b}_{\text{lon}}$ has the value `"conditional"` and fulfills at least one of the three sub-falsification criteria $F_{\text{stop}}(t)$, $F_{\text{NST}}(t)$ or $F_{\text{NRL}}(t)$ for minimum one <condition> element $\mathfrak{c}$ of all these elements $\mathfrak{C}_{\mathfrak{b}_{\text{lon}}}$ of $\mathfrak{b}_{\text{lon}}$.

$$B_{\text{lon}}(\mathfrak{a}, \mathfrak{b}_{\text{lon}}, t) \leftarrow \text{crossing}(\mathfrak{b}_{\text{lon}}) \in \{\texttt{"prohibited"}, \texttt{"not possible"}\} \vee \left(\begin{array}{l} \text{crossing}(\mathfrak{b}_{\text{lon}}) = \texttt{"prohibited"} \wedge \\ \bigvee\limits_{\mathfrak{c} \in \mathfrak{C}_{\mathfrak{b}_{\text{lon}}}} \left(F_{\text{stop}}(\mathfrak{a}, \mathfrak{c}, t) \vee F_{\text{NST}}(\mathfrak{a}, \mathfrak{c}, t) \vee F_{\text{NRL}}(\mathfrak{a}, \mathfrak{c}, t) \right) \end{array} \right) \tag{6-9}$$

The sub-falsification criteria for the conditions are explained in the following subsubsections. The value `"allowed"` of attribute `crossing` has no influence on the falsification criterion.

6.3.1 Stop Condition

If the ego vehicle enters the new area with a <condition> element of the type `"stop"`, the ego vehicle must have stopped with its nose (with the front bumper) in the previous time step within a stop zone. The stop zone is located in the previous area (viewed from the direction of travel), extends laterally across the entire width of the area and has a certain longitudinal dimension. A value close to reality of $30\,\text{cm}$ was defined here. When viewed from the direction of travel, the zone ends with the end of the area, so that it lies directly in front of the beginning of the new area. Thus the following definition of this sub-falsification criterion arises.

Definition 6.12 (Stop Condition)**.** The sub-falsification criterion for the stop condition $F_{\text{stop}}(\mathfrak{a}, \mathfrak{c}, t)$ is fulfilled in time step t if attribute `type` of condition $\mathfrak{c}$ has the value `"stop"` and in time step $t-1$ either

- the longitudinal position ${}^{\text{q}}x_{\mathfrak{e}}(t-1)$ of $\mathfrak{e}$' front bumper with respect to the segment sequence coordinate system S_{q} was not in stop zone range ($\neg \left({}^{\text{q}}x_{\mathfrak{a},\text{min}} - d_{\text{stop}} \leq {}^{\text{q}}x_{\mathfrak{e}}(t-1) \leq {}^{\text{q}}x_{\mathfrak{a},\text{min}} \right)$), where d_{stop} is the longitudinal expansion of the stop zone and ${}^{\text{q}}x_{\mathfrak{a},\text{min}}$ is the longitudinal position where $\mathfrak{a}$ begins in direction of travel or
- $\mathfrak{e}$'s speed $v_{\mathfrak{e}}(t-1) \neq 0$.

$$F_{\text{stop}}(\mathfrak{a}, \mathfrak{c}, t) \leftarrow \text{type}(\mathfrak{c}) = \texttt{"stop"} \wedge \left(\begin{array}{l} \neg \left({}^{\text{q}}x_{\mathfrak{a},\text{min}} - d_{\text{stop}} \leq {}^{\text{q}}x_{\mathfrak{e}}(t-1) \leq {}^{\text{q}}x_{\mathfrak{a},\text{min}} \right) \vee \\ v_{\mathfrak{e}}(t-1) \neq 0 \end{array} \right) \tag{6-10}$$

6.3.2 No Stagnant Traffic Condition

A <condition> element with attribute value pair `type="no stagnant traffic"` means that an entry is only allowed if the traffic flow within this element does not stagnate.

Neither in the BSSD specification nor in the StVO there is a precise definition of stagnant traffic. According to the Swiss Federal Office for Roads, it is a case of stagnant traffic "if the greatly reduced vehicle speed out of town is below $30\,\text{km/h}$ for at least $1\,\text{min}$ and/or there is a short standstill in some areas."[60] Europe's largest traffic club[61], the General German Automobile Club (ger.: Allgemeiner Deutscher Automobil-Club

[60] Bundesamt für Strassen: Definitionen (2018).

[61] Karg, J.: ADAC-Chef August Markl (2019).

e. V.) (ADAC) defines stagnant traffic as several vehicles driving for at least $5\,\mathrm{min}$ and $1\,\mathrm{km}$ with an average speed in the range of $20...40\,\mathrm{km/h}$.[62] Neither of these definitions can easily be generalized to apply meaningfully to areas within cities, especially since the no stagnant traffic condition applies mainly at intersections or at crosswalks, i.e., to relatively short segments. In a very strict interpretation, the sub-falsification criterion can be defined as follows.

Definition 6.13 (No Stagnant Traffic Condition – Part 1/2)**.** The sub-falsification criterion for the no stagnant traffic condition $F_{\mathrm{NST}}(\mathfrak{a}, \mathfrak{c}, t)$ is fulfilled at time step t if attribute `type` of condition $\mathfrak{c}$ has the value `"no stagnant traffic"` and at least one TP $\mathfrak{p}$ of all other TP in the simulation environment $\mathfrak{P}\backslash\{\mathfrak{e}\}$ is also completely or partially located at t or was located during the movement from $t-1$ to t on this area $\mathfrak{a}$.

A behavior that would be forced by this interpretation of the criterion would be rather untypical in practice. In order to formalize the criterion a bit more realistically, the researched speed limits for stopped traffic out of town can be put in relation to the usual permissible speed limits there. In Germany, the maximum allowed speed out of town, if not further restricted, for passenger cars is $100\,\mathrm{km/h}$ if the road is not a freeway or structurally similar.[63] Less than $6\,\%$ of the interurban roads in Germany are freeways.[64] Therefore, the assumption is made that the $100\,\mathrm{km/h}$ limit applies to most roads outside towns. With such a maximum speed limit and stagnant traffic with an average speed of $40\,\mathrm{km/h}$, only $40\,\%$ of the permitted speed could be used. In Switzerland, the general speed limit out of town is $80\,\mathrm{km/h}$ (except on freeways)[65]. In case of traffic stuck at $30\,\mathrm{km/h}$, this would be $38\,\%$ – i.e. a similar value. With this consideration, the previously very strict sub-falsification criterion can be improved.

Definition 6.14 (No Stagnant Traffic Condition – Part 2/2)**.** $\mathfrak{p}$ is considered only if its speed $v_{\mathfrak{p}}(t) < 0.4 \cdot v_{\mathfrak{p},\max}$, where $v_{\mathfrak{a},\max}$ is the maximum allowed speed at $\mathfrak{a}$.

$$F_{\mathrm{NST}}(\mathfrak{a}, \mathfrak{c}, t) \leftarrow \mathrm{type}(\mathfrak{c}) = \texttt{"no stagnant traffic"} \wedge \bigvee_{\mathfrak{p} \in \mathfrak{P}\backslash\{\mathfrak{e}\}} \left(v_{\mathfrak{p}}(t) < 0,4 \cdot v_{\mathfrak{a},\max} \wedge \bigvee_{\mathfrak{a}_{\mathfrak{p}} \in \mathfrak{A}_{\mathfrak{p}}(t)} \mathfrak{a}_{\mathfrak{p}} = \mathfrak{a} \right) \tag{6-11}$$

Time or length specifications of a slow movement as a further necessary criterion for stagnant traffic, as shown in the research results, are not used here, since they are difficult to transfer to short areas inside towns.

It should be mentioned at this point that the no stagnant traffic condition only specifies that no stagnant traffic may be present at entry. Whether the ego vehicle itself causes stagnation of the traffic flow after entry is irrelevant here.

[62] Abrahamczyk, M.: Ab wann ist ein Stau ein Stau? (2020).

[63] Bundesministerium der Justiz und für Verbraucherschutz: Straßenverkehrs-Ordnung (2013), § 3 (3).

[64] Bundesministerium für Verkehr und digitale Infrastruktur: Längenstatistik Straßen überörtlicher Verkehr (2020), p. 1.

[65] Schweizerischer Bundesrat: Verkehrsregelnverordnung (2021), Art. 4a (1).

6.3.3 No Red Light Condition

Definition 6.15 (No Red Light Condition)**.** The sub-falsification criterion for the no red light condition $F_{\mathrm{NRL}}(\mathfrak{a}, \mathfrak{c}, t)$ is fulfilled in time step t if attribute `type` of condition $\mathfrak{c}$ has the value `"no red light"` and $\mathfrak{a}$ is controlled by a traffic light that displays the signal red at time t.

$$F_{\mathrm{NRL}}(\mathfrak{a}, \mathfrak{c}, t) \leftarrow \mathrm{type}(\mathfrak{c}) = \texttt{"no red light"} \wedge \mathrm{hasRedTrafficLight}(\mathfrak{a}, t) \tag{6-12}$$

6.4 Falsification Criterion for Lateral Boundary Regulation

According to the BSSD specification, the regularities for leaving an area in lateral direction are controlled by the <lateral> element with the <left> and <right> children. Each child specifies a regularium for a lane change in the respective direction.

Definition 6.16 (Lat. Boundary Regulation – Part 1/5)**.** The falsification criterion for lateral boundary regulation $F_{\mathrm{lat}}(t)$ is fulfilled at time t if ego vehicle $\mathfrak{e}$ is in the state of a lane change $l_{\mathfrak{e}}(t) \neq 0$ and fulfills the lateral boundary violation $B_{\mathrm{lat}}\left(\mathfrak{b}_{\mathrm{lat},\mathfrak{a}}\left(l_{\mathfrak{e}}(t)\right), t\right)$ for the suitable lateral boundary element $\mathfrak{b}_{\mathrm{lat},\mathfrak{a}}\left(l_{\mathfrak{e}}(t)\right)$ of at least one area $\mathfrak{a}$ of all the original areas $\mathfrak{A}_{\mathrm{origin},\mathfrak{e}}(t)$.

$$F_{\mathrm{lat}}(t) \leftarrow l_{\mathfrak{e}}(t) \neq 0 \wedge \bigvee_{\mathfrak{a} \in \mathfrak{A}_{\mathrm{origin},\mathfrak{e}}(t)} B_{\mathrm{lat}}\left(\mathfrak{b}_{\mathrm{lat},\mathfrak{a}}\left(l_{\mathfrak{e}}(t)\right)\right) \tag{6-13}$$

Definition 6.17 (Lat. Boundary Regulation – Part 2/5)**.** $\mathfrak{e}$ changes its lane change state from *no lane change* ($l_{\mathfrak{e}}(t_{\mathrm{s}} - 1) := 0$) to *lane change to left* ($l_{\mathfrak{e}}(t_{\mathrm{s}}) := 1$) when it crosses the left boundary of at least one area $\mathfrak{a}$ of all the areas $\mathfrak{A}_{\mathfrak{e}}(t_{\mathrm{s}})$ on which $\mathfrak{e}$ is completely or partially located in t_{s} or was located during the movement from $t_{\mathrm{s}} - 1$ to t_{s}. Accordingly, it changes from $l_{\mathfrak{e}}(t_{\mathrm{s}} - 1) := 0$ to $l_{\mathfrak{e}}(t_{\mathrm{s}}) := -1$ (*lane change to the right*) when $\mathfrak{e}$ crosses the right boundary of $\mathfrak{a}$. It changes to $l_{\mathfrak{e}}(t_{\mathrm{e}}) := 0$ when $\mathfrak{e}$ is only on one area in lateral direction. (t_{s} and t_{e} are start and end time of lane change.)

Contrary to the usual definition of lane change, the lane change state $l_{\mathfrak{e}}(\cdot)$ refers not only to a completed change process, but also to a scenario in which the vehicle only partially or temporarily enters another lane.

Note: The lane change state is not further defined and formalized at this point, since this is not relevant for the continuation of this thesis. In the following implementation, this state does not have to be detected for the ego vehicle, but is set by an action of the behavior planner.

Definition 6.18 (Lat. Boundary Regulation – Part 3/5)**.** If $\mathfrak{e}$ is in a lange change to the left, the original areas $\mathfrak{A}_{\mathrm{origin},\mathfrak{e}}(t)$ are all areas $\mathfrak{A}_{\mathfrak{e}}(t)$ on which $\mathfrak{e}$ is completely or partially located in t or was located during the movement from $t - 1$ to t with the exception of the leftmost areas $\mathfrak{A}_{\mathrm{left},\mathfrak{e}}(t)$, and if $\mathfrak{e}$ is in a lange change to the right, then $\mathfrak{A}_{\mathrm{origin},\mathfrak{e}}(t)$ are $\mathfrak{A}_{\mathfrak{e}}(t)$ without the rightmost areas $\mathfrak{A}_{\mathrm{right},\mathfrak{e}}(t)$.

$$\mathfrak{A}_{\text{origin},\mathfrak{e}}(t) := \begin{cases} \mathfrak{A}_{\mathfrak{e}}(t) \notin \mathfrak{A}_{\text{left},\mathfrak{e}}(t), & \text{if } l_{\mathfrak{e}}(t) = 1 \\ \mathfrak{A}_{\mathfrak{e}}(t) \notin \mathfrak{A}_{\text{right},\mathfrak{e}}(t), & \text{if } l_{\mathfrak{e}}(t) = -1 \\ \mathfrak{A}_{\mathfrak{e}}(t), & \text{otherwise} \end{cases} \tag{6-14}$$

If the ego vehicle is located within one segment during the change, only the regularity of the one original area must be considered. If the ego vehicle is on several segments at the same time or has passed through several segments since the last time step, all these original areas must be checked accordingly.

Definition 6.19 (Lat. Boundary Regulation – Part 4/5)**.** If $\mathfrak{e}$ is in a lange change to the left, the suitable lateral boundary element $\mathfrak{b}_{\text{lat},\mathfrak{a}}\left(l_{\mathfrak{e}}(t)\right)$ is the <left> element $\mathfrak{b}_{\text{left},\mathfrak{a}}$ and its the <right> element $\mathfrak{b}_{\text{right},\mathfrak{a}}$, if $\mathfrak{e}$ is in a lange change to the right. If no lane change is performed, then there is no matching element.

$$\mathfrak{b}_{\text{lat},\mathfrak{a}}\left(l_{\mathfrak{e}}(t)\right) := \begin{cases} \mathfrak{b}_{\text{left},\mathfrak{a}}, & \text{if } l_{\mathfrak{e}}(t) = 1 \\ \mathfrak{b}_{\text{right},\mathfrak{a}}, & \text{if } l_{\mathfrak{e}}(t) = -1 \\ \varnothing, & \text{otherwise} \end{cases} \tag{6-15}$$

Definition 6.20 (Lat. Boundary Regulation – Part 5/5)**.** $B_{\text{lat}}\left(\mathfrak{b}_{\text{lat},\mathfrak{a}}\left(l_{\mathfrak{e}}(t)\right)\right)$ is fulfilled if attribute `crossing` of $\mathfrak{b}_{\text{lat},\mathfrak{a}}\left(l_{\mathfrak{e}}(t)\right)$ has the value "`prohibited`" or "`not possible`".

$$B_{\text{lat}}\left(\mathfrak{b}_{\text{lat},\mathfrak{a}}\left(l_{\mathfrak{e}}(t)\right)\right) \leftarrow \text{crossing}\left(\mathfrak{b}_{\text{lat},\mathfrak{a}}\left(l_{\mathfrak{e}}(t)\right)\right) \in \{\texttt{"prohibited"}, \texttt{"not possible"}\} \tag{6-16}$$

The value "`allowed`" has no influence on the falsification criterion.

6.5 Falsification Criterion for Speed Limit

Definition 6.21 (Speed Limit)**.** The falsification criterion for maximum allowed speed $F_{\text{speed}}(t)$ is fulfilled at time t if the current speed $v_{\mathfrak{e}}(t)$ of the ego vehicle $\mathfrak{e}$ is greater than the maximum allowed speed $v_{\max,\mathfrak{a}}$ of at least one area $\mathfrak{a}$ of all the areas $\mathfrak{A}_{\mathfrak{e}}(t)$ on which $\mathfrak{e}$ is completely or partially located at time t or was located during the movement from the previous time step $t-1$ to t.

$$F_{\text{speed}}(t) \leftarrow \bigvee_{\mathfrak{a} \in \mathfrak{A}_{\mathfrak{e}}(t)} v_{\mathfrak{e}}(t) > v_{\max,\mathfrak{a}} \tag{6-17}$$

6.6 Falsification Criterion for Overtaking Regulation

The first paper about BSSD said: „Overtaking takes place when two traffic participants are traveling in the same direction on the same roadway with different speeds and the faster vehicle passes the slower vehicle.“[66] (See also subsubsection 4.1.1.) From this the falsification criterion is derived as follows.

Definition 6.22 (Overtaking Regulation)**.** The falsification criterion for overtaking regulation $F_{\text{overtake}}(t)$ is fulfilled at time t if

- one segment $\mathfrak{s}_\mathfrak{e}$ of all segments $\mathfrak{S}_\mathfrak{e}(t)$ on which the ego vehicle $\mathfrak{e}$ is completely or partially located in t or was located during the movement from $t-1$ to t is the same segment $\mathfrak{s}_\mathfrak{p}$ of all segments $\mathfrak{S}_\mathfrak{p}(t)$ there at least one TP $\mathfrak{p}$ of all other TP in the simulation environment $\mathfrak{P}\backslash\{\mathfrak{e}\}$ is also completely or partially located at t or was located during the movement from $t-1$ to t and
- $\mathfrak{e}$ and $\mathfrak{p}$ are traveling in the same direction and speed ${}^{\mathrm{q}}v_\mathfrak{e}(t)$ of $\mathfrak{e}$ is higher than speed ${}^{\mathrm{q}}v_\mathfrak{p}(t)$ of $\mathfrak{p}$ (${}^{\mathrm{q}}v_\mathfrak{e}(t) > v_\mathfrak{p}(t) \geq 0$)[67] with respect to the ego vehicle's segment sequence coordinate system S_q and
- $\mathfrak{e}$ and $\mathfrak{p}$ overlap longitudinally but not laterally[68] and
- at least one area $\mathfrak{a}_\mathfrak{e}$ of all areas $\mathfrak{A}_{\mathfrak{s}_\mathfrak{e}}(t)$ of segment $\mathfrak{s}_\mathfrak{e}$ has an element <`overtake`> with the attribute value pair `permission="no"`.

$$
F_{\text{overtake}}(t) \leftarrow \bigvee_{\mathfrak{p}\in\mathfrak{P}\backslash\{\mathfrak{e}\}} \left(\begin{array}{l} {}^{\mathrm{q}}v_\mathfrak{e}(t) > {}^{\mathrm{q}}v_\mathfrak{p}(t) \geq 0 \wedge \\ {}^{\mathrm{q}}x_{\mathfrak{e},\min}(t) \leq {}^{\mathrm{q}}x_{\mathfrak{o},\max}(t) \wedge {}^{\mathrm{q}}x_{\mathfrak{e},\max}(t) \geq {}^{\mathrm{q}}x_{\mathfrak{o},\min}(t) \wedge \\ \neg \left({}^{\mathrm{q}}y_{\mathfrak{e},\min}(t) \leq {}^{\mathrm{q}}y_{\mathfrak{o},\max}(t) \wedge {}^{\mathrm{q}}y_{\mathfrak{e},\max}(t) \geq {}^{\mathrm{q}}y_{\mathfrak{o},\min}(t) \right) \wedge \\ \bigvee\limits_{\mathfrak{s}_\mathfrak{e}\in\mathfrak{S}_\mathfrak{e}(t)} \left(\begin{array}{l} \bigvee\limits_{\mathfrak{s}_\mathfrak{p}\in\mathfrak{S}_\mathfrak{p}(t)} \left(\mathfrak{s}_\mathfrak{e} = \mathfrak{s}_\mathfrak{p} \right) \wedge \\ \bigvee\limits_{\mathfrak{a}_\mathfrak{e}\in\mathfrak{A}_{\mathfrak{s}_\mathfrak{e}}(t)} \neg\, \mathrm{overtakePermission}(\mathfrak{a}_\mathfrak{e}) \end{array} \right) \end{array} \right) \tag{6-18}
$$

The two TP must not overlap laterally, otherwise it would not be an overtake, but a collision. And since this lateral overlap must not be fulfilled explicitly, it must be ensured for checking the falsification criterion that these two TP are on the same segment.

It should be noted again that according to the BSSD specification, the overtake permission refers to overtakes that happen both in the same area or on different lanes.

[66] Glatzki, F. et al.: Behavioral Attributes for a BSSD (2021).

[67] If $\mathfrak{p}$ travels in the opposite direction of $\mathfrak{e}$, than ${}^{\mathrm{q}}v_\mathfrak{p}(t) < 0$.

[68] Bounding boxes are also used here. See also formula 6-3

7 Implementation and Training of Behavior Planner

The RL behavior planner developed as part of this thesis follows the approaches of the behavior planners presented in subsection 3.2 where a long road is simulated on which the ego vehicle moves. In this thesis, this road is the segment sequence as presented in section 5. However, unlike in the presented projects, a set of behavioral rules should be followed here, while the presented behavioral planners were only concerned with avoiding collisions.

For the realization, a software project was created in the programming language Python and thematically divided into modules. Figure 7-1 shows the corresponding data flow diagram. Modules are symbolized as stadiums, while rectangles represent stored states. In the following subsections the individual modules and their connections are described in more detail.

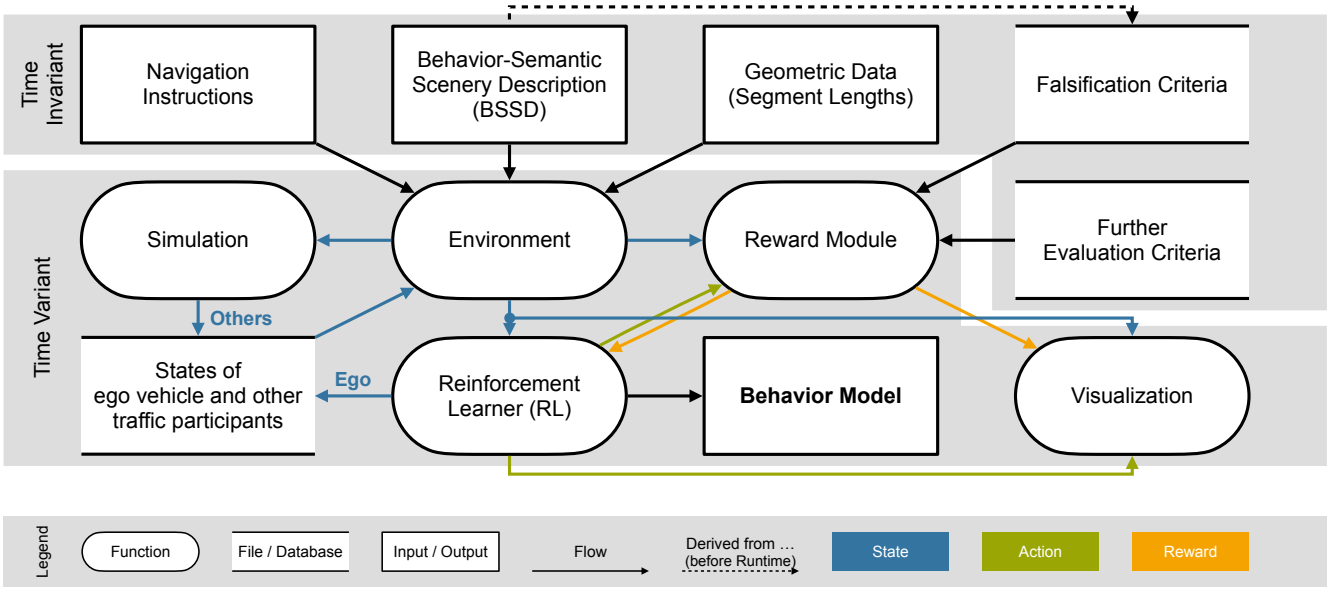


Figure 7-1: Data Flow Diagram of Implementation.

Due to the complexity of this task it is important to develop the individual modules and functions with the focus on high correctness and usability from the beginning. Thus, among other things, for almost all methods and functions optional type hints were used in Python to declare the types of arguments and return values.[69] This prevented incorrect usage later in the development. In addition, the correctness of critical program parts was ensured by writing and applying tests.

7.1 Environment

The module *environment* is a central component of the behavior planner. It prepares information, merges it, makes it available to other modules, stores state changes and maintains a history.

With the initialization of the environment it receives the BSSD of the demo map as a `pybssd` object, which was created before as described in section 4. In addition, one-dimensional geometry data for each segment of the road map is passed to the `Environment` object in the form of a Python dictionary as well

[69] Rossum, G. v. et al.: PEP 484 (Type Hints) (2014).

as another dictionary covering the traffic signals. Moreover, the environment receives all TP that were previously initialized as objects, including the ego vehicle. TP whose behaviors have to be simulated are created within simulation objects, which are then passed to the environment. (See subsection 7.3).

An element of type `TrafficParticipant` has mainly the following information passed to it for its initialization:

- a boolean statement about whether it is the ego vehicle or not (i.e., this TP controlled by the behavior planner to learn the target behavior or not)
- the type of the vehicle analogous to the `object` attribute in BSSD,
- the initial position, i.e. the ID of the used way, segment and lanes, and a logitudinal offset in reference direction in meters, which describes the position within a segment,
- a route this TP has to follow.

So far, the environment is not designed as a multi agent system, which means that only one TP can have the value `True` for property `ego`.

An element of type `TrafficParticipant` can include some further properties, such as the current speed, the target speed level and the state whether the TP is currently in a lane change operation. For this purpose the property `lane_change` can comprise the enumeration object `LaneChange`. ”An enumeration is a set of symbolic names bound to unique, constant values.”[70] The symbolic names of `LaneChange` are `NO`, `LEFT`, `RIGHT`. A history is created for these dynamic properties of a TP, since some falsification criteria require access to values from past time steps.

The background of the route approach is that a vehicle to be controlled by the behavior planner can only move along a sequence of segments specified by a navigation system. An object of the type `Route`, is initialized by a list of instructions, i.e. of nodes, which are to be driven along in order to reach the desired destination. If the specification of a node does not unambiguously describe which route is to be used there, the instruction must be extended by specifying the way. If the direction of travel is not unambiguous for a way, this must also be specified. This allows the user to specify routes comparatively easily. By the method `get_completed_route_using_bssd()` a route can be completed under specification of the BSSD of this road network. This is done by adding any missing values for way and direction to each instruction and inserting more instructions for connecting ways in nodes. A route can also represent a circuit, so that, for example, the ego vehicle can be trained on an endless segment sequence.

After the `Environment` object has been initialized, it can either be used to train a behavior using RL (see subsection 7.5) or to start the visualization to provide an overview to a human supervisor (see subsection 7.4). In both cases, the `Environment` object is repeatedly asked for the state description from the ego vehicle’s point of view. Such a state for an TP includes the properties of ego vehicle itself and the other TP and the segment sequence of the ego vehicle.

Depending on the used architecture of the NN of the RL behavior planner, it may be necessary that the input vector, which is derived from ego vehicle’s state, must always have the same size. To ensure this, the state has to be restricted to a certain FoV. It then contains only at most as many other TP, as

[70] Warsaw, B. et al.: PEP 435 (Enum) (2013).

many segments in the sequence, as many lanes in the segments, as many reservations in the lanes and as many reservation links in the reservations, as is specified when initializing the ego vehicle via the class `FieldOfView`. It can be specified separately to how many segments in front of and behind the ego vehicle the sequence should be limited. The purpose of including segments behind the ego vehicle is to allow it to react to other TP approaching it from behind.

Each segment of this sequence is based on a segment as described by the BSSD. However, since ways are omitted as parent elements, each object of this new type `Segment` also contains the information of its parent element, plus the following additional information:

- the position of this segment in the sequence, where a value of 0 means that it is the segment on which the (front bumper of the) TP is currently located, negative values mean that the segments are located behind the TP and in front of it for positive values,
- the direction in which this segment is traveling with respect to the reference direction,
- the length of this segment in meters and
- the main lane, i.e. the lane on which the TP is located at sequence position 0 or the lane that follows if the position of the TP is continued forward (for segments with position > 0) or backward (for segments with position < 0).

Similar to the new class `Segment` there is also a new class `Lane`. It no longer includes two behavior descriptions, but only the description for the direction in which it is to be driven. Instead, it stores whether a traffic signal regulates this area and if so, which traffic signal is currently shown. For this there is an enumeration class `TrafficLight` with the possible states `NO` (no traffic light available), `INACTIVE` (no traffic light is shown), `GREEN`, `YELLOW`, `RED` and `RED_YELLOW` (red and yellow at the same time). The method `is_equivalent_to_bssd_traffic_light()` of `TrafficLight` returns a boolean value that indicates whether the state of the traffic light object from `pybssd` with the states `NO`, `INACTIVE` and `ACTIVE` matches the state of this `TrafficLight` object.

Besides getting the state for a TP, performing an action of the ego vehicle is the other important function of `Environment`, which is done via the `perform()` method. Performing an action takes the simulation environment to the next time step. The simulated time that elapses between two steps is adjustable via a variable. It was set to $0{,}5\,\mathrm{s}$ in order to be able to represent follow maneuvers, as they certainly occur in practice. In Germany, the legal recommendation of a time gap of $2\,\mathrm{s}$[71a] is very often undercut. An exemplary evaluation of the distances actually driven on a German freeways shows that the time gap is $1{,}4\,\mathrm{s}$ on average[71a]. The law states that vehicles driven by humans must not fall below the time gap of $0{,}9\,\mathrm{s}$[71a]. But even this is sometimes undercut in practice. With a time step length of $0{,}5\,\mathrm{s}$, $85\,\%$ of the subsequent trips are covered, as they appeared in the evaluation.[71b]

The selected action of the ego vehicle is mapped as enumeration object `Action` and can have one of the following values.

- `KEEP`: Neither the target speed is to be changed, nor a lane change is to be performed.
- `FASTER`: The target speed is to be increased by one step.

[71] Maurer, M. et al.: Autonomes Fahren (2015). a: p. 314; b: p. 337.

- `SLOWER`: The target speed is to be decreased by one step.
- `STOP`: The target speed is to be set to $0\,\mathrm{km/h}$.
- `LEFT`: A lane change is to be made to the left.
- `RIGHT`: A lane change is to be made to the right.

The speed levels are as follows.

$$\left\{\begin{array}{lllll} 0\,\mathrm{km/h}, & 2\,\mathrm{km/h}, & 7\,\mathrm{km/h}, & 10\,\mathrm{km/h}, & 20\,\mathrm{km/h}, \\ 30\,\mathrm{km/h}, & 40\,\mathrm{km/h}, & 50\,\mathrm{km/h}, & 60\,\mathrm{km/h}, & 70\,\mathrm{km/h}, \\ 80\,\mathrm{km/h}, & 90\,\mathrm{km/h}, & 100\,\mathrm{km/h}, & 120\,\mathrm{km/h}, & 130\,\mathrm{km/h} \end{array}\right\}$$

The levels cover the usual speed limits in the range from $30\,\mathrm{km/h}$ up to the recommended speed of $130\,\mathrm{km/h}$ applicable on German freeways. In Germany, only walking speed is allowed in a low-traffic zone (ger.: verkehrsberuhigter Bereich). Several courts determined that this has the value of $7\,\mathrm{km/h}$. [72,73,74]

The levels $10\,\mathrm{km/h}$ and $20\,\mathrm{km/h}$ provide the option of adapting the speed to critical traffic scenarios. The level of $2\,\mathrm{km/h}$ is necessary in order to be able to hit the predefined zone of $30\,\mathrm{cm}$ at a time step duration of $0{,}5\,\mathrm{s}$, in which the ego vehicle must stop for a longitudinal entry condition of the type `"stop"` in order to meet this condition. (See also 6.3.1.)

For the ego vehicle, the acceleration values for reaching the next target speed level can be set via variables. The chosen values are based on practical experience: A positive acceleration occurs with the value $2\,\mathrm{m/s^2}$ [75a] and a deceleration with the value $5\,\mathrm{m/s^2}$. The action `STOP` represents an emergency braking, therefore the target speed is set to $0\,\mathrm{km/h}$ and the deceleration in each time step in which `STOP` is selected occurs with the value of $10\,\mathrm{m/s^2}$ [75b].

Since the lateral resolution of the simulation environment is limited to lanes, a lane change of the ego vehicle is simulated as a jump between two lanes after an appropriate action has been selected for a certain number of time steps. An analysis by Sporrer et al.[76] showed that the duration of a normal lane change depends on traffic density but not on the selected speed. They considered 100 change maneuvers each in the city and on freeways and 50 on rural roads and found that the duration was in the range of $3{,}1...6{,}5\,\mathrm{s}$. For the ego vehicle, a change duration of $4\,\mathrm{s}$ was set because cities are expected to have higher traffic density than the overall average and the focus of this thesis is on urban traffic scenarios.

The changes of position, speed and the state of a lane change from the ego vehicle are performed in the Reward Module together with the evaluation of the individual actions and returned to `Environment`. During development, this approach has proven to be very efficient in terms of development effort and computational effort per time step, even if it seems counterintuitive at first to assign this task to the Reward Module as well.

[72] Oberlandesgericht Köln: Ss 782/84 (1985).

[73] Oberlandesgericht Karlsruhe: 1 Ss 159/03 (2004).

[74] Oberlandesgericht Brandenburg: 1 Ss (OWi) 86 B/05 (2005).

[75] Burg, H.; Moser, A.: Handbuch Verkehrsunfallrekonstruktion (2017). a: p. 464; b: p. 471.

[76] Sporrer, A. et al.: Realsimulation von Spurwechselvorgängen im Straßenverkehr (1998).

After the action of the ego vehicle has been evaluated and performed, the last thing to be done in the `perform()` method of `Environment` is to run the simulations of other TP for this time step. The method returns a boolean value which indicates whether this is the last time step of the simulation for the ego vehicle. A simulation run is finished, if the ego vehicle has reached its destination, is in a state due to an action, whereby the simulation has to be aborted (e.g. leaving the road due to a lane change) or if time out has been reached. time out is reached if a simulation runs longer than $5\,\mathrm{min}$ (in simulation time) or if the ego vehicle has not moved for $30\,\mathrm{s}$. The second value may have to be increased for training in other scenarios, e.g. if a longer waiting time in front of a traffic light or a railroad crossing is to be assumed.

7.2 Reward Module

As mentioned before, the task of the reward module is not only to generate the score for each action of the behavior planner at each time step, but it also generates the new state of the ego vehicle resulting from the action. The `get_reward()` function is executed with `state` and `action`. In addition, the BSSD and the geometry data of the entire road map are also passed, since these have to be used by some falsification criteria. A list of `Reward` objects is returned, and values describing the new state of the ego vehicle, such as position, current and target speed, direction of travel, and lane change. For each falsification criterion that was met in this time step, the reward list contains an entry. Such a `Reward` element has at least three properties:

- a value of this reward or the penalty (negative reward) with respect to a criterion
- a boolean information about whether the rund can be continued or must be aborted due to this violation
- a human-readable reason for this entry

It would be possible to assign different values to violations. Thus, actions that lead to more serious violations could be given more weight with even smaller values in order to force the avoidance of these actions in particular during the training. However, since every action that leads to a violation should be avoided at the end of the training, the value -1 was set equally for all falsification criteria. If several rules are violated by one action at the same time, the sum is formed over all individual reward values, all abort statements are linked using bitwise *or* and all justifications are output. Care was taken to ensure that an action that led to the fulfillment of a falsification criterion on several areas at the same time is still only evaluated once.

Non-compliance with certain falsification criteria must lead to an abort or restart of the simulation in some scenes, in fact whenever an action causes the ego vehicle to enter a state that cannot be represented and/or the ego vehicle cannot get out of it. Example: The ego vehicle is on the rightmost lane and makes a lane change to the right. The resulting condition is not mappable in the simulation and would not be useful for further training runs. If the reward list contains such an entry, the environment is reinitialized during the training.

In order to be able to monitor the formalization and implementation of the falsification criteria and the training process, it is very useful to be able to view a listing of the violations committed by the ego vehicle at each time step. Such a justification for the evaluation of an action chosen by the planner should be

passed on to the visualization accordingly. Therefore, a human-readable justification can be added to each entry of the reward list.

The falsification criteria only represent the boundaries of the space which describes a legal behavior. A vehicle operating within it will obey all the rules, but is not yet required to have a sensible driving style. Without further evaluation of an action, the behavior planner will learn very quickly that doing nothing is the most effective way to obtain the highest reward, namely 0. Therefore, further entry is added to the reward list with a positive value if no falsification criterion is fulfilled and the ego vehicle has a speed $v_{\mathfrak{e}}(t) > 0$. The value increases linearly with speed and ends in 1 when the maximum allowed speed $v_{\max,\mathfrak{a}}$ for area $\mathfrak{a}$ is reached. If the speed increases further, the falsification criterion for overspeed is fulfilled and the value decreases again to -1. Figure 7-2 illustrates this with an example in which driving up to a maximum speed of $30\,\mathrm{km/h}$ is allowed.

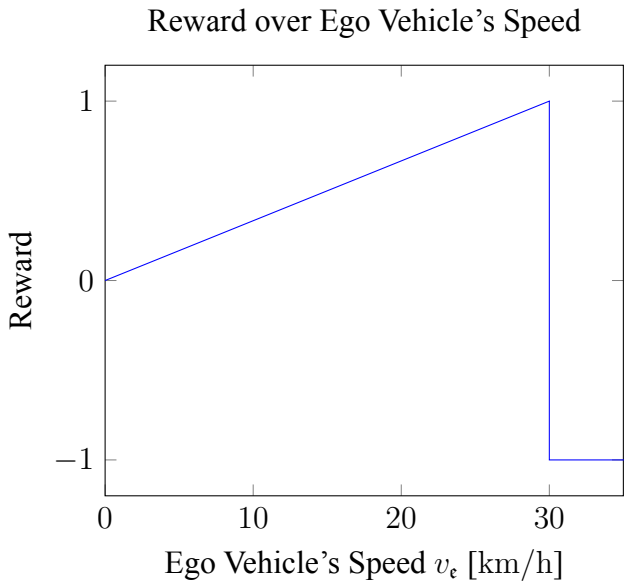


Figure 7-2: Exemplary reward progression over ego vehicle's speed $v_{\mathfrak{e}}$ for driving in area $\mathfrak{a}$ with maximum allowed speed $v_{\max,\mathfrak{a}} = 30\,\mathrm{km/h}$.

Within the function `get_reward()` first the new target and current speed is calculated based on the selected action. For this the time step length, the steps for target speed and the acceleration values are used, which are specified in the environment module. Also the new lane change state is created or the lane position is actually changed. This is done in parallel with checking the falsification criterion for lateral boundary regulation, as described in subsubsection 7.2.4. Then, the ego vehicle is propagated forward along the segment sequence to the new position and, at the same time, with respect to the segments that the ego vehicle has now newly entered or that it first entered during propagation, the falsification criteria related to entering areas are checked. Then, the remaining falsification criteria that are independent of entering an area are tested. In the following subsubsections the implementations of the individual falsification criteria are described.

7.2.1 Falsification Criterion for Collision

Since only the longitudinal dimensions of segments are available as geometric data in this simulation environment, collisions can be determined within one way, but if the colliding vehicles are on different ways, collision detection is not possible. However, a workaround has been found for this. These two cases are treated separately in the following, like they are also treated separately in the reward module.

Collision Within Same Ways

The falsification criterion regarding a collision of the ego vehicle with another TP is checked at each time step. For this purpose, it is first iterated over all areas on which the ego vehicle is completely or partially located or was located during the movement since the last time step. Within this iteration there are two more iterations: over all other TP and over the areas, on which such an TP is completely or partially located or was located during the movement since the last time step by using the method `falsification_criteria.collision_within_same_ways()`.

Within this triple nested iteration it is checked if the areas of the partial positions of the ego vehicle and of another TP are the same, but also if for this TP a collision is already registered in the reward list, so that despite possibly several overlapping partial positions a collision with an TP is weighted only once. For each TP that shares or shared an area with the ego vehicle, a check is now made to see if the two vehicles overlap or have overlapped since the last time step. Between two time steps no interpolation of the positions of both TP is done, because the benefit and the necessary programming and computational effort would not be in proportion.

Instead, for each area that the two TP share or shared, the zones within the area that each of them traveled since the last time step are determined. For this purpose, the lengths of both TP are taken into account. This is similar to bounding boxes, which are often used to simplify collision determinations. However, the bounding boxes in this case are one-dimensional, since the simulation environment does not provide further lateral resolution within an area.

Such a comparison of the traveled zones in an area is necessary at still further places in code, so the function `helpers.get_offset_min_max()` was implemented, to get the longitudinal limits of the traveled zone of a TP. If the *1D bounding boxes* overlap in at least one area, then there is a collision of the ego vehicle with this TP. In this case, a `Reward` element is added to the reward list.

Checking the foreseeability of a collision, as required by the BSSD specification, was not implemented, as this would have been beyond the scope of this thesis.

In subsection 6.1 it was described that besides foreseeability also avoidability is an important criterion for the penalty in case of a collision. However, according to the scope of this thesis, the addition from definition 6.4 was not implemented either. Nevertheless, it is important that the behavior planner is not penalized for the misbehavior of others in order to avoid training an overcautious behavior.[77] Therefore, the implementation of the falsification criterion was extended to ensure that a penalty only occurs, if the ego vehicle is at fault.

[77] Qin, X. et al.: Falsification with Dynamically Constrained Reinforcement Learning (2020).

Of course, a detailed algorithm for determining fault cannot be given at this point. In order to approximate the fault clarification, three cases were defined in which the ego vehicle is at least partially at fault for the collision and receives a negative reward for it:

- The directions of travel of both TP were opposite and the ego vehicle had a speed $v_e > 0$.
- The directions of travel were in the same direction and the ego vehicle had a higher speed $v_e > v_o$ than the other TP.
- The ego vehicle was in a lane change operation in the direction of the other TP.

Inter Way Collision Prevention

In this simulation environment, it is unknown whether two ways intersect and thus whether two vehicles collide that are on different ways. However, since BSSD specifies that intersecting ways must be grouped into nodes, it is at least possible to determine whether two TP are within a node, even if they are on different ways. To prevent collisions in nodes, a TP should only enter a node if there is no other TP in that node on all other ways. Therefore, using `falsification_criteria.collision_prevention_in_nodes()` method, a negative reward is given, if the ego vehicle enters an area that is within a node and there is at least one other TP completely or partially on another way within that node or was there during its movement since the last time step. On the other hand, if both TP were on the same way within that node, a collision again can be determined more precisely, as described above, so at this point the penalty is only applied if the ways differ.

7.2.2 Falsification Criterion for Obstruction

The falsification criterion for obstruction as described in subsection 6.2 was not be realized according to the scope of this thesis. Instead, a different approach was used to detect obstruction. This approach checks whether another TP, to which the ego vehicle must give priority, would collide with the ego vehicle if it took a way leading to the same position where the ego vehicle is located then. For a TP that have a higher reservation right for the area where the ego vehicle is located, the implementation is even so strict that the falsification criterion is already fulfilled if the ego vehicle cannot leave the area until it is entered by the other TP. This is based on a strict interpretation of the description in the BSSD specification regarding the <reservation> element: ”Specifies the conditions that need to be fulfilled to enter and/or remain in this area.”[78,79].

In the following two minisections the implementations for the two cases of equal and unequal reservation right are described in more detail. In the third minisection the tree search is described, which is used for both cases.

Obstruction of Equally Qualified Participants

If the ego vehicle enters a new area, within `falsification_criteria.obstruction_equal()` method it is checked whether another TP with the same reservation right, which would already be on the area, would have to brake so strongly because of the ego vehicle in order to avoid a collision that a certain limit

[78] Glatzki, F.; Lippert, M.: BSSD Specification (2021), p. 8.

[79] Lippert, M. et al.: BSSD of Road Networks for Automated Driving (2022).

value would be undercut. This obstruction deceleration value was set to $7\,\mathrm{m/s^2}$ and thus lies between the values for normal braking and hazard braking.

Two cases are considered for this purpose: If both TP were driving in opposite directions, a frontal crash would occur or in the obstruction case the other TP would drive into the back of the ego vehicle, if they were driving in the same direction. For both cases, the minimum distances which have to be ensured can be determined when the ego vehicle enters this area.

For the lengths of the two distances, using the tree search described bellow, the road network around the ego vehicle is searched for other TP which are candidates for obstruction. If at least one such TP is found, the falsification criterion for obstruction is fulfilled.

Obstruction of Participants with Higher Reservation Right

With regard to another TP with a higher reservation right, it is not sufficient to consider only the areas to which the reseravation link refers. The reservation link has the task to describe, where a TP must come from, so that it can receive that reservation right. However, therefore a TP does not have to be exactly on the linked area to have this right. If such a linked area is very short and a TP comming from this direction would be obstructed, which is still on the neighboring area, then the falsification criterion must still be fulfilled. In the opposite case, the linked area could be very long. Then, the behavior planner must not be penalized for having a TP on the linked area, which is still far enough away that the ego vehicle could still leave the reserved area without affecting the other TP. To solve this, at each time step in the method `falsification_criteria.obstruction_higher()` it is considered which time the ego vehicle needs to leave this area completely. This is then compared to the time it would take another TP to reach the area where the ego vehicle is. This is illustrated in figure 7-3 (but with distances instead of travel times).

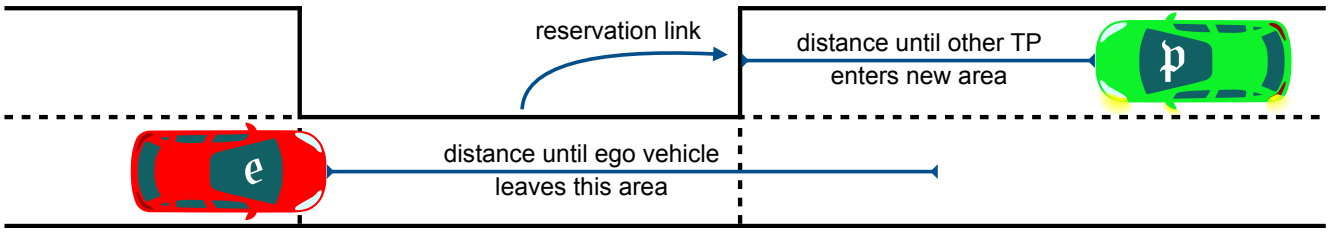


Figure 7-3: Example of an obstruction of TP (p) by ego vehicle (e) although the TP as a higher reservation right than ego vehicle for the area where ego vehicle is located.

However, if the reservation link refers to an area that is in the same segment, i.e. if another TP on another lane of this segment has a higher reservation right, then this examination is not carried out, since using this method it cannot be determined when the other TP enters the area on which the ego vehicle is located. In this case, the falsification criterion is fulfilled without further examination. Accordingly, this also applies to pedestrians who come from outside the road at the level of this segment and have a higher right of reservation, e.g. at a crosswalk.

To determine the time required by the other TP to reach the reserved area, speed and acceleration values must be assumed. For the other TP, the worst case assumption is made that it always travels as fast as is permitted for it in the respective areas. A step change between the speeds is assumed. If the TP would

drive faster than allowed and thus reach the reserved area earlier, the falsification criterion would not be fulfilled (false negative). On the other hand, the TP itself would violate the rules to be there so fast. For the ego vehicle, its current speed is assumed, and an acceleration with the previously defined value of $2\,\mathrm{m/s^2}$ up to the maximum speed allowed for that segment, which it would then maintain until it left the segment completely. So, the falsification criterion would be fulfilled, if the ego vehicle could not leave the area in time even if it would accelerate to the allowed maximum speed to leave the area even faster.

This time is used to search the road network around the ego vehicle for other TP that are candidates for such obstruction using the tree search described in the next minisection. If at least one such TP was found, the falsification criterion for Obstruction is fulfilled.

Tree Search

For checking both cases of obstruction, a tree search over the areas is performed in the `helpers.get_list_of_obstructed_tp()` method. Starting with the area where the ego vehicle is located, the neighboring areas are searched and then the areas neighboring them and so on. The depth of the search depends on the case. In the obstruction search of equal TP, the minimum distances are used to limit the depth of the search. These limits vary and depend on whether the part of the tree being searched is ahead of or behind the ego vehicle in the direction of travel. For the obstruction search of a TP that has a higher right, the time the ego vehicle would need to leave the area is used to limit the search.

Therefore, the depth is determined in each area that is covered during the search. Either simply the length of the area is considered (same reservation right) or the length divided by the permitted maximum speed to determine the travel time in this area is considered (higher reservation right). Care is taken that during the tree search only those areas are expanded a further time, which receive a more critical depth value when viewed again via another search path and expands the search space at this point. This is the case when such an area is visited again via a shorter search path. Thus, it is possible that the search space must be extended at this point.

Before the list of recognized TP the ego vehicle has to give priority is returned to the previous explained methods, TP which cannot be obstructed even if they are in the search space are filtered out. This is the case if the ego vehicle and the other TP are driving in the same direction, but the other TP is behind the ego vehicle and driving slower.

Figure 7-4 and 7-5 illustrate two tree searches for the ego vehicle located within way 1 segment 1 lane -1 (see figure 4-4). In figure 7-4 the depth is regulated by the distance. In the direction of travel the search is for $50\,\mathrm{m}$ and against the direction of travel for $30\,\mathrm{m}$. After searching the area where the ego vehicle is located, $47\,\mathrm{m}$ respectively $28\,\mathrm{m}$ remain. In figure 7-5 the depth of the search corresponds to the travel time of $5\,\mathrm{s}$. After the expansion of the first area, $4{,}64\,\mathrm{s}$ of travel time remain. In both cases, the last nodes of the trees have negative values for the remaining depth. This means that the corresponding areas no longer have to be searched completely, but only up to the respective length.

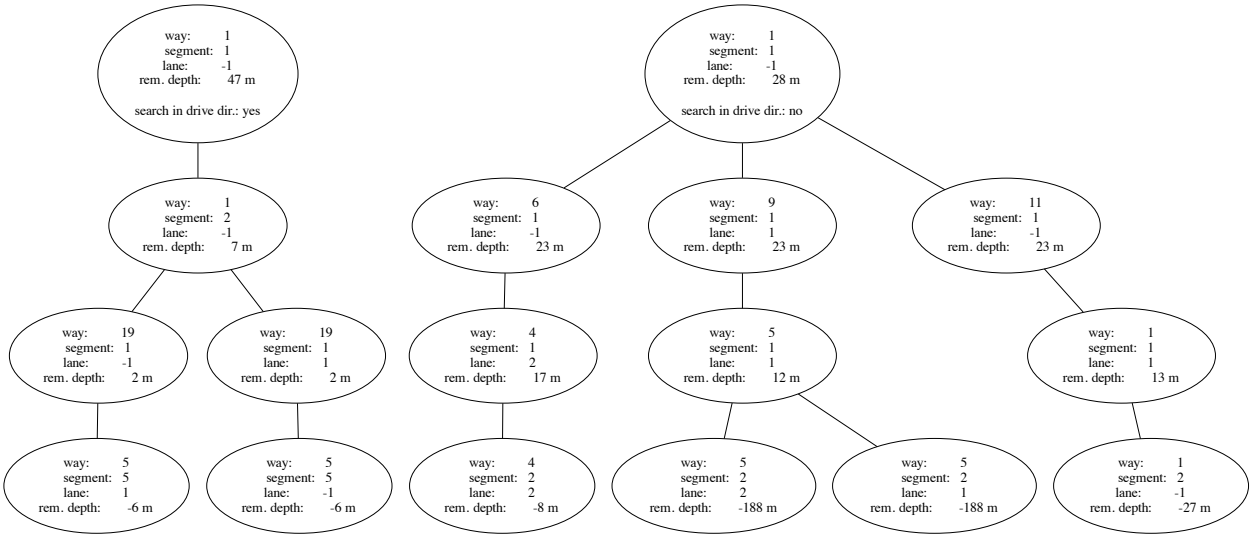


Figure 7-4: The figure illustrates a tree search with a search depth of $50\,\mathrm{m}$ in direction of travel and $30\,\mathrm{m}$ against direction of travel to detect obstructions of TP with the same reservation right as the ego vehicle.

7.2.3 Falsification Criterion for Longitudinal Boundary Regulation

The check of the falsification criterion regarding the longitudinal entry of an area is performed per time step for all areas that the ego vehicle has entered since the last time step using the method `falsification_criteria.boundary_longitudinal()`. This is usually one or no area, but can also be several areas if the ego vehicle is traveling at a correspondingly high speed or the areas are comparatively short.

For each new area entered in this time step, an iteration is performed over the longitudinal boundary specifications that match the state of that area relative to a traffic signal. This is done using the `is_equivalent_to_bssd_traffic_light()` method as described in subsection 7.1. If the crossing property states, that entry is prohibited or not possible, then a corresponding `Reward` element is added to the reward list. If entry is not possible, then this element has the property `abort = True`, which causes that this run is aborted, because the ego vehicle is in a state that can not be represented in the simulation environment. If the crossing property is conditional, then within a nested second iteration all conditions of this longitudinal boundary constraint are checked. In this, depending on the type of the conditions, the corresponding check of the sub-falsification criteria is executed.

The implementation of sub-falsification criteria was done exactly as described in the subsubsections 6.3.1, 6.3.2 and 6.3.3. If a sub-falsification criterion is fulfilled, then a corresponding `Reward` element is added to the reward list. In all three cases, it is not necessary to abort the simulation.

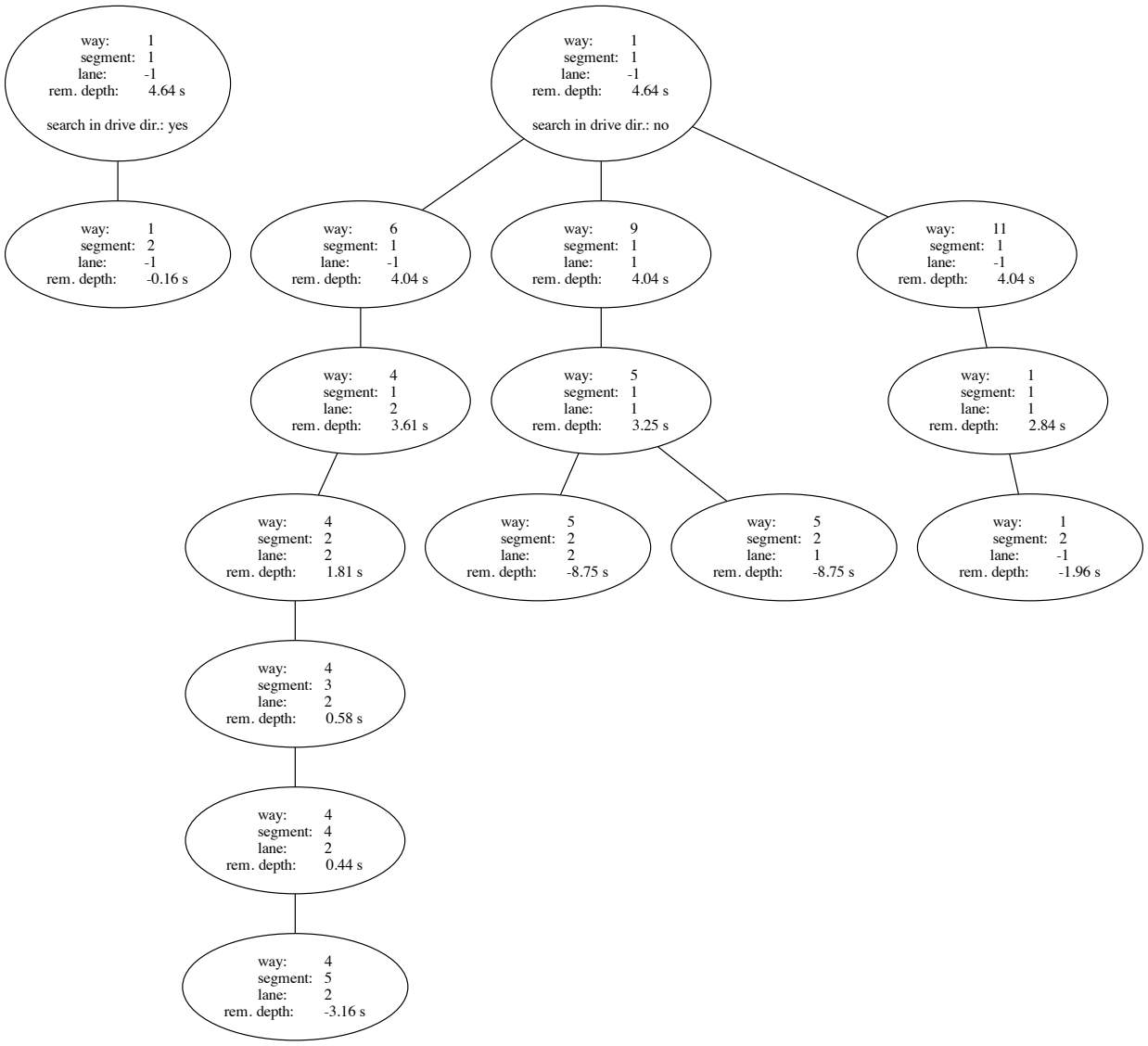


Figure 7-5: The figure illustrates a tree search with a search depth of $5\,\mathrm{s}$ travel time to detect obstructions of TP with higher reservation right than the ego vehicle.

7.2.4 Falsification Criterion for Lateral Boundary Regulation

Due to the lack of lateral resolution within an area, a lane change is performed as a jump if the `lane_change` property of a TP (object of class `TrafficParticipant`) has the value `LaneChange.LEFT` or `LaneChange.RIGHT` for a defined time. As described in subsection 7.1, the lane change duration was set to $4\,\mathrm{s}$. The state transition diagram in figure 7-6 shows which combination of actions result in what kind of lane change state. Here, as an example, the TP is in lane -1 before the lane change is completed. If a TP, in this case the ego vehicle, is in the `LaneChange.NO` state, it can be moved to the `LaneChange.LEFT` or `LaneChange.RIGHT` state using the `Action.LEFT` or `Action.RIGHT` action, respectively. To successfully complete the lane change operation, the RL behavior planner must maintain this state for the defined lane change time. To do this, the same action must be selected in

the following time steps, i.e. `Action.LEFT` in the `LaneChange.LEFT` state and `Action.RIGHT` in the `LaneChange.RIGHT` state. Alternatively, `Action.SLOWER` or `Action.FASTER` can be selected to change the speed during the change process.

By `Action.KEEP`, but also if `Action.RIGHT` is selected in the `LaneChange.LEFT` state and vice versa, the change process is aborted and the lane change state goes back to its initial value `LaneChange.NO`. The idea behind `Action.STOP` is the possibility to perform an emergency stop. Ego receives the target speed $0\,\mathrm{km/h}$ and can reach it with a higher deceleration value. In addition, `lane_change = LaneChange.NO` is also set here to abort a possible lane change at the same time. If the lane change state has the value `LaneChange.LEFT` or `LaneChange.RIGHT` for so many time steps, that the lane change duration is reached, then the position of the ego vehicle is changed accordingly and the state goes back to `LangeChange.NO`.

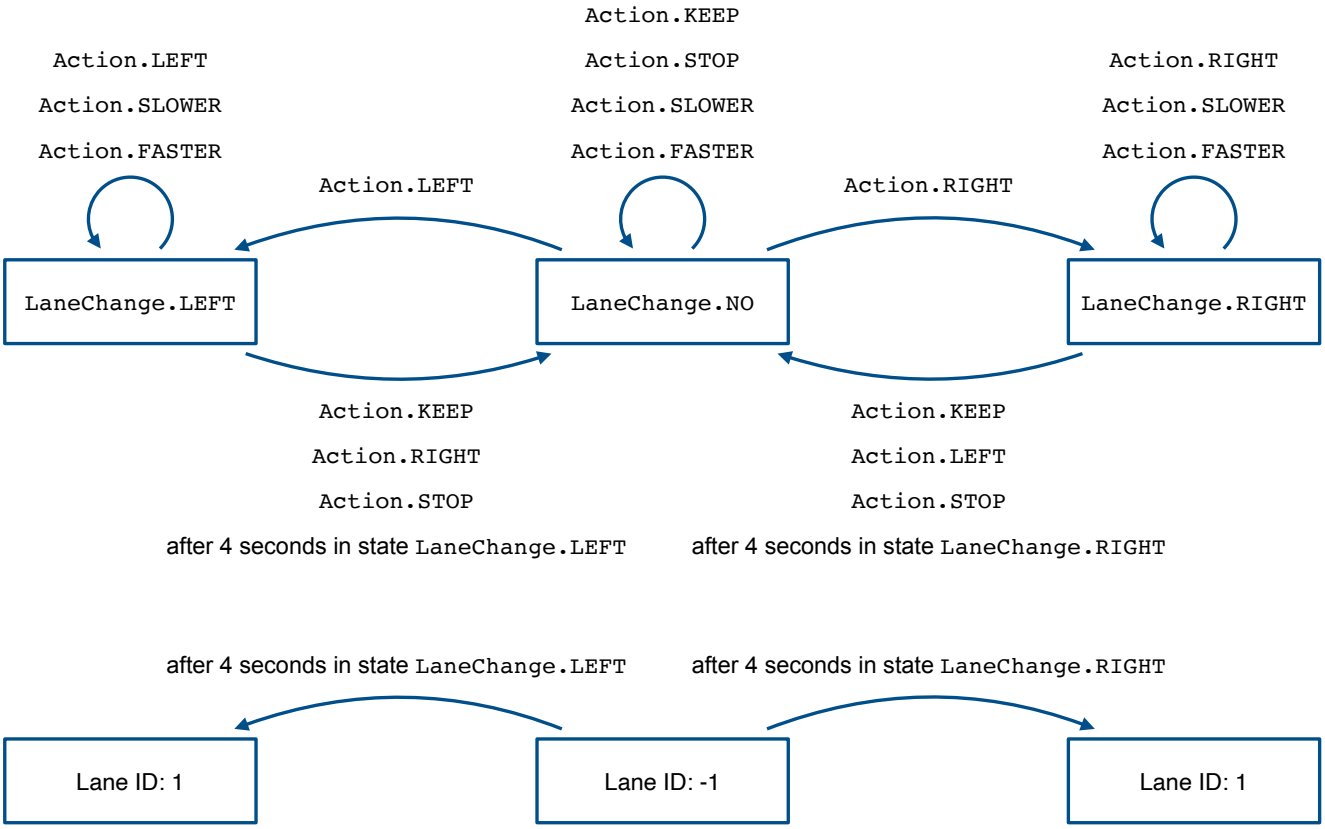


Figure 7-6: State transition diagram for lane change. As shown here, as an example, the TP is in lane -1 before the lane change is completed.

During the entire lane change operation, the ego vehicle must comply with all other falsification criteria on both lanes. During this time, the position of the ego vehicle includes the new lane in addition to the original lane, so that all other falsification checks can include all areas on which the ego vehicle is partially located. However, when testing the falsification criterion for lateral boundary regulation, only the original lane is considered, since only the behavioral specification on these areas are relevant.

If the behavioral regulation for lateral leaving such an origin area is not followed, a corresponding `Reward` element is added to the reward list. If the lateral boundary regulation says that leaving is not only forbidden,

but even not possible, then this element has the property `abort = True`, which causes that this run is aborted, because the ego vehicle is in a state that can not be represented in the simulation environment. Both the testing of this falsification criterion and the handling of the lane change state of the ego vehicle is done in method `falsification_criteria.boundary_lateral()`.

7.2.5 Falsification Criterion for Speed Limit

The test of the falsification criterion regarding the exceeding of the maximum permissible speed is performed at each time step using the method `falsification_criteria.speed()`. For this purpose, an iteration is performed over all areas on which the ego vehicle is completely or partially located or was located during the movement since the last time step. If the maximum speed of the ego vehicle was exceeded for at least one area, then a corresponding >Reward< object is added to the reward list.

7.2.6 Falsification Criterion for Overtake Regulation

The check of the falsification criterion concerning the overtaking of other TP is done at each time step using the method `falsification_criteria.overtake()`. However, since there is no further lateral resolution within an area in the chosen implementation of the simulation environment and the TP always completely fills an area laterally, it is excluded that an overtaking process happens within an area, because that would be a collision.

In order to find a not allowed overtaking, it is first iterated over all areas on which the ego vehicle is completely or partially located or was located during the movement since the last time step. Only those areas are considered further for which overtaking is prohibited. Now, within this iteration, a second iteration is performed over all other TP and a third iteration is performed over the areas on which such a TP is completely or partially located or was located during the movement since the last time step.

Within this threefold nested iteration it is checked if the segments of these two areas of the ego vehicle and the other TP are the same, but the areas themselves differ. Different from the collision detection and also different from the concept for this falsification criterion, for the realization the case is excluded in which the two TP share or shared the same area, because otherwise it is necessarily a collision, which is covered by the other falsification criterion.

Thus, if the ego vehicle shares or shared the same segment but not the same area with at least one other TP, and both TP are traveling in the same direction and the ego vehicle is faster than the other TP and the traveled zones of both TP longitudinally overlap, then a corresponding `Reward` object is added to the reward list. To determine the traveled zones the function `helpers.get_offset_min_max()` is used, which output the longitudinal borders of such a zone, as described in subsubsection 7.2.1.

7.3 Simulation of Other Traffic Participants

In addition to the ego vehicle, other TP can also be present in the simulation environment. If these should not be static, their behavior must be simulated. Each such TP is initialized in a corresponding simulation object for this purpose. A list of the simulations is passed as initialization parameter to the environment. After within the method `perform()` of the `Environment` object the ego vehicle was set into its new state, for the same time step each TP which has to be simulated is also set into its new state using its simulation. So far, only one type of simulator has been realized, namely for pedestrians at crosswalks.

A TP which is simulated using `PedestrianCrossingSimulator` appears after a random time of absence in the range of $0...10\,\mathrm{s}$ at one of the two roadsides of the given segment. At which roadside of the segment the pedestrian appears is also decided randomly. The TP remains there for $3\,\mathrm{s}$ before stepping onto the roadway to allow the behavior planner to react to it. However, it would already be a reservation violation if the ego vehicle would enter the crosswalk at this time. After the waiting period, the pedestrian begins the road side change at a random lateral speed in the range of $1...5\,\mathrm{s}$ per area. Due to the lack of lateral resolution within areas, the side change is also simulated only by jumps from area to area. After the pedestrian has reached the other side, it disappears and the process starts again.

7.4 Visualization

For testing the functionality of the whole project, especially for validating the implementation of the falsification criteria, but also for interpreting the result of the training process, a visualization is essential. For this purpose, a web application was created, which can be used to display the ego vehicle within its segment sequence and other TP within it. Furthermore, the ego vehicle can be controlled via the web application and the reward list can be displayed at the respective time. For visualization, a specific web page is accessed in a browser on the same computer. Figures 7-7 and 7-8 show the visualization of two scenes.

The visualization is divided into three areas. At the top left is the *Control Pane*, which consists of three tabs. In the *Action* tab (see figure 7-7), the user can manually control the ego vehicle using the known actions (blue buttons) or let the RL behavior planner choose an action for this time step (red button). After clicking one of the buttons, the updated segment sequence and reward list will be displayed without the need to reload the page in the browser. In the *Segment Sequence* tab, the view of the segment sequence can be customized. The *Traffic Light* tab can be used to change the states of the traffic signals on the map (see figure 7-8).

At the top right is the *Info Pane*. The *Reward* tab is particularly important. There, the reward list of the respective time step is visualized. In the scene shown in Figure 7-7, the ego vehicle fulfills three falsification criteria simultaneously. Accordingly, the reward for the learner in this time step is -3. The run in this example can continue in the time step shown because none of the list entries require termination. In the scene shown in Figure 7-8, the ego vehicle is within the legal behavior space and receives a positive reward depending on its speed.

In the bottom pane of the visualization, the segment sequence of the ego vehicle is shown, as well as the ego vehicle itself, all other TP located in areas of the segment sequence and the traffic lights assigned to the areas. The sequence has the length as set by the FoV parameter of the ego vehicle. The segments can be displayed in a compact view where they all have the same length (as shown in figure 7-7) or to scale in their respective extents (as shown in figure 7-8), depending on the user's preference.

For each area, the required behavior is represented by lines and symbols. Thus, a white border of an area means that longitudinal entry or lateral exit is prohibited via this side. A black line means that it is not possible. A white dashed line means that longitudinal entry is regulated via one of the three conditions. If this is the case, a corresponding symbol in the box above the respective area indicates which conditions are involved.

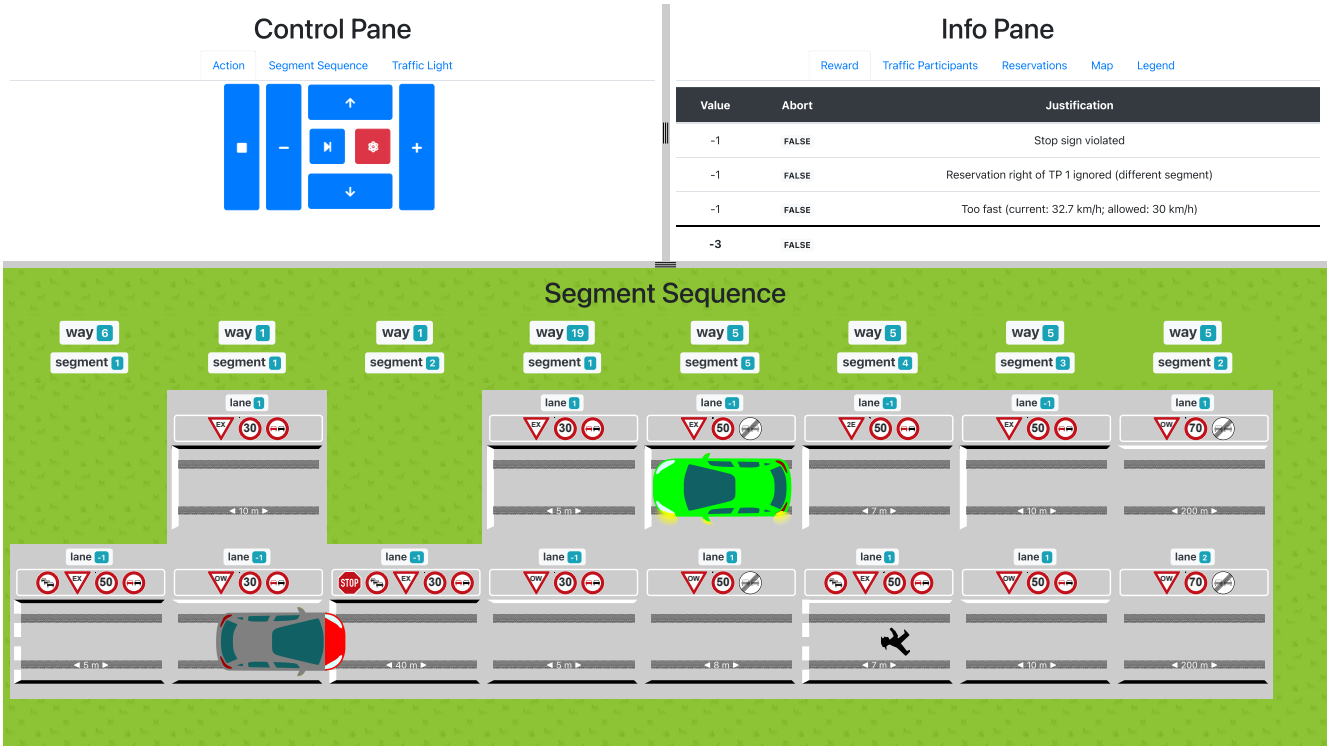


Figure 7-7: Visualization Example 1: The figure shows the *Action* tab in the *Control Pane* and a segment sequence of a scene with multiple TP.

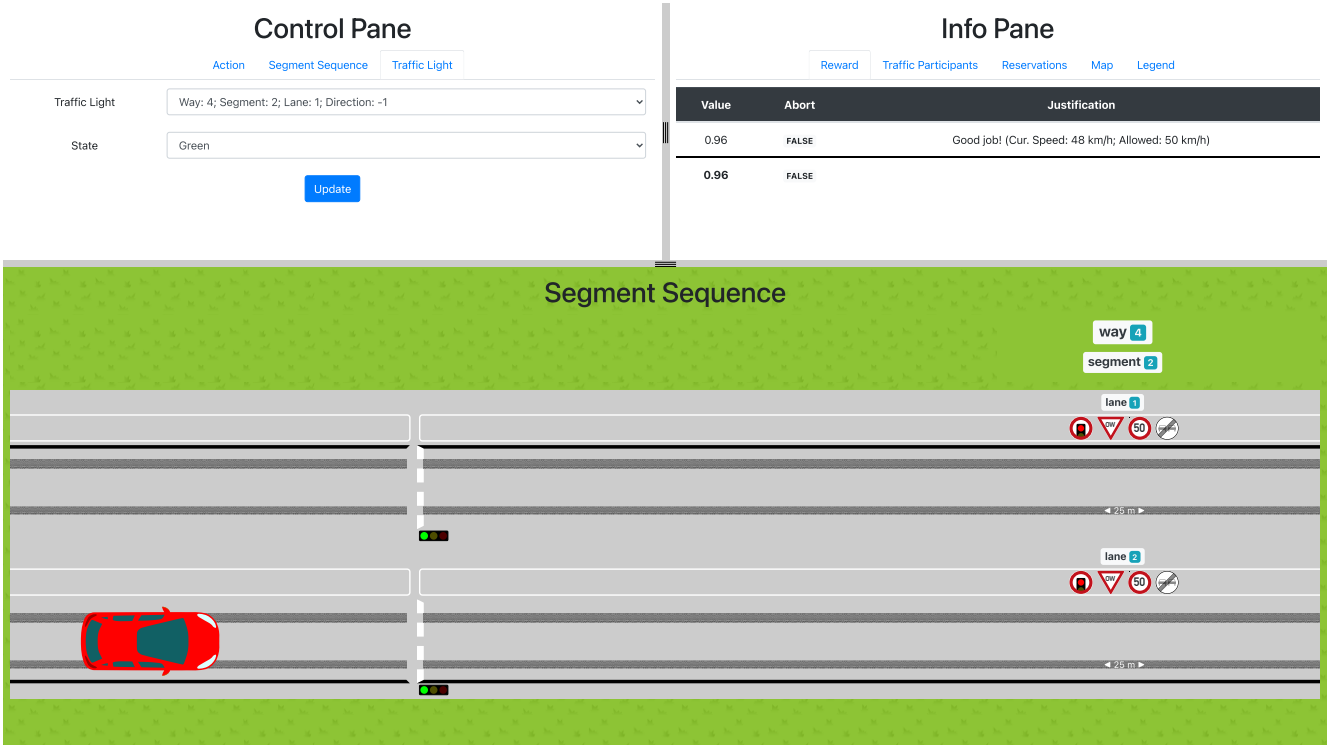


Figure 7-8: Visualization Example 2: The figure shows the *Traffic Light* tab in the *Control Pane* and a segment sequence of a scene with two traffic lights.

- Stop Condition: See the first symbol of way 1 segment 2 lane -1 in figure 7-7.
- No Stagnant Traffic Condition: See the first symbol of way 6 segment 1 lane -1 in figure 7-7.
- No Red Light Condition: See the first symbol of way 4 segment 2 lane 1 (and 2) in figure 7-8

In addition, the reservation right, the maximum permitted speed and the permission or prohibition to overtake are displayed there. The following abbreviations are used to indicate the right of reservation:

- OW: own reserved
- EX: externally reserved
- EQ: equally reserved
- nE with $n \in \mathbb{N}$: There are several (externally or equally) reservation elements

Figure 7-7 shows three TP. The ego vehicle in red, another vehicle in green, and a pedestrian. However, the ego vehicle is on two segments at the time shown. Since the front bumper is the reference for positioning vehicles, the foremost part of the ego vehicle is shown in the proper color, while the part that is still on the segments behind it is shown in gray. If a vehicle is in a lane change operation, this is symbolized by direction indicators of the vehicle.

7.5 Behavior Planner

The RL behavior planner was implemented as DDQN because it is possible to achieve results better than average compared with other RL methods.[80,81] The implementation of the training process is contained in the `Learner` class, that of the actual behavior planner in the `Agent` subclass.

Note: The code used to implement the DDQN and the training process was originally taken from a tutorial by Tabor[82,83] and has been modified and extended for the purpose of this project. Before adaptation for the BSSD behavior planner, the functionality of the code was tested on a more simple example: OpenAI's *Lunar Lander* simulation.[84]

For the initialization of `Learner` the environment is passed and an object of the subclass `Agent` is created. `Learner` then provides two methods: `train()` to train the agent, and `apply()` to apply the learned behavior in the visualization.

The behavior planner receives all important parameters for the initialization as an object of the class `Agent`, which is need for training and applying. These parameters are the size of the input vector, the number of possible actions, the batch size and the learning rate. (For an explanation of these terms, see subsection 2.2.) Also, this determines the discount factor γ and the maximum, minimum and decay values of ϵ, which describes the trade-off between exploration and exploitation. (For an explanation of these terms,

[80] Wang, Z. et al.: Dueling Deep Reinforcement Learning (2016).

[81] Hessel, M. et al.: Rainbow (2018).

[82] Tabor, P.: main_keras_dueling_dqn_lunar_lander.py (2020).

[83] Tabor, P.: dueling_dqn_keras.py (2020).

[84] OpenAI: Gym (2016).

see subsection 2.3.) Like DeepTraffic[85], the behavior planner uses an experience replay (as described in subsection 3.2). The size of this replay buffer, as well as the replace value for a target network, is also passed here during initialization.

A target network is a complementary way to the experience replay to avoid oscillating or diverging values for the weights of the neurons in the NN. Therefore, `Agent` includes two DDQN with the same architecture: an evaluation network and a target network. "The predicted Q values of [the target network] are used to backpropagate through and train the main Q-network [(evaluation network)]. It is important to highlight that the target network's parameters are not trained, but they are periodically synchronized with the parameters of the main Q-network. The idea is that using the target network's Q values to train the main Q-network will improve the stability of the training."[86]

The architecture of the DDQN looks like this: Before the network is split into V and A stream, it has five fully connected layers. The first two layers have twice as many neurons as the input vector is long, so that there is no shortage of opportunities for the NN to solve the task. The following three layers then have as many neurons as the length of the input vector. The output vector on the A stream represents the action space, where each entry in the vector represents an action. The action with the highest value in the vector is used to control the ego vehicle at that time step. The activation function used everywhere is ReLU.

If in class `Learner` the method `train()` is called, the training process for the behavior planner is executed for the specified number of iterations. A training run takes as many time steps as the ego vehicle is in the environment until either it performs an action that must lead to a termination of the simulation or until time out is reached (time out criteria are explained in subsection 7.1). After each training run the environment is reset and the ego vehicle starts again. Likewise, ϵ is then reduced by the specified decay value. During the training the weights of DDQN as well as the reached values of a run like reward, distance and steps are stored, so the DDQN can be tested later in the visualization and the data can be evaluated later. As optimization function ADAM is used.

Which procedure was chosen to find good hyperparameters for the behavior planner is described in subsubsection 7.5.2. But first, the description of how the input vector is generated from the state of the ego vehicle (including its segment sequence) follows.

7.5.1 Input Vector Generation

The input of the DDQN is the state of the ego vehicle and its environment. However, to feed the state into the NN, it must be put into the form of a number vector. For a NN as used here, this vector must always have the same length. Therefore, when initializing the ego vehicle, a FoV was defined, which limits the segment sequence and the number of TP to be considered.

The method used to create the input vector is based on the method used by Fridman et al.[85]. In this paper, the ego vehicle and the environment covered by the FoV are viewed and gridded in the top-down view. In this, the ego vehicle has a fixed position. Each cell is as wide as the lane, has a length of for example $20\,\mathrm{cm}$, and is described by the same number of properties, such as whether it is occupied by a vehicle

[85] Fridman, L. et al.: DeepTraffic (2018).

[86] Torres, J.: Deep Q-Network (DQN)-II (2020).

and the speed of a vehicle in that cell. All cells and their properties are then strung together to create the vector.

However, since the behavior planner in this thesis is supposed to consider multiple behavior requirements, many properties describing a single cell are required, e.g., the maximum allowed speed, the boundary regulations, and many more. This would make the vector extremely long, so the training process would take a very long time. Thereby, the vector would contain a lot of redundant information. To reduce the vector length, the constraint of the BSSD specification is used, that the behavior requirements within an area are homogeneous. This way the length of the input vector for the NN can be reduced significantly.

To create the vector, the `get_input_vector()` method first iterates over all segments in the segment sequence reduced by the FoV. Within this iteration, way ID, segment ID, the direction in which the ego vehicle traverses the segment with respect to the reference direction, and the segment length are added as individual entries to the vector. Also, within this iteration, all areas of the segment are iterated over. Within the second, nested iteration, the lane `ID`, the traffic light state, and the requested behavior are added.

In order to represent the behavior requirements of all areas within the vector, several vector entries per area are necessary:

- 8 entries for longitudinal boundary, since up to 2 boundary elements are possible (with 2 entries per element for the attributes `crossing` and `traffic_light`) with up to 2 condition elements (with 1 entry per element for the attribute `type`) ($2 \cdot (2 + 2 \cdot 1) = 8$),
- 2 entries for lateral boundary,
- multiple entries for reservation, depending on the number of reservation and reservation link elements specified by FoV,
- 1 entry for maximum allowed speed, and
- 1 entry for overtake.

After the segment sequence has been transferred to the vector, the TP must still be appended. These are the ego vehicle and as many other TP as FoV specifies. The ego vehicle is always the first TP in the vector, so that the behavior planner can recognize the ego vehicle by its position in the vector. For each TP, one entry is added to the vector for each of the following properties:

- type of TP,
- way ID of its position,
- segment ID of its position,
- lane ID of its position,
- longitudinal offset within the segment,
- speed,
- lane change state and

- target speed (only for ego vehicle).

The entries with the ID of way, segment and lane serves to allow the NN to make a cognitive connection between the TP and the areas. To achieve good results in the training process and to speed up this process, the input data must be normalized beforehand.[87] Therefore, the properties with numerical values, such as velocity or ID, are normalized to the range $[0, 1[$ and the remaining properties are also mapped to numerical values in this range. Since the input vector must always have the same length, the vector is padded with the number 1 at the appropriate places if, for example, the number of areas in a segment of the sequence is less than the maximum defined by FoV.

7.5.2 Procedure for Training

The success of a training procedure is largely dependent on the selection of the right hyperparameters. These must be found experimentally. For this purpose, grid search was used, i.e. all combinations in a certain range were tested. Two hyperparameters were examined here, the learning rate and the discount factor, since these most likely have the greatest influence on the result. Other possible hyperparameters are the replace value and the size of the experience replay buffer. However, these were not examined in more detail, because this would have exceeded the scope of this thesis. For replace the value 100 was set permanently, which means that every 100 time steps in a run the target network is replaced by the evaluation network. The buffer can contain the fixed number of $100\,000$ state action reward combinations.

During training, it turned out that the input vector and thus the whole NN, which is large enough to cover as many reservations and reservation links per area as can occur in the demo map, is too large to produce meaningful training results in the remaining time. Therefore, no reservations were considered for training and thus the corresponding falsification criterion for obstruction was also not considered further.

The other TP used was the simulated pedestrian crossing the road at the crosswalk. However, since no reservation rights were considered, the purpose of the pedestrian was only to learn collision avoidance.

First, combinations of hyperparameters were tested in a wide range to find out to which smaller range the search can be restricted. For learning rate, the values

$$\left\{\begin{matrix} 5 \cdot 10^{-7}, 1 \cdot 10^{-6}, 5 \cdot 10^{-6}, 1 \cdot 10^{-5}, 5 \cdot 10^{-5}, 1 \cdot 10^{-4}, \\ 5 \cdot 10^{-4}, 1 \cdot 10^{-3}, 5 \cdot 10^{-3}, 1 \cdot 10^{-2}, 5 \cdot 10^{-2}, 1 \cdot 10^{-1} \end{matrix}\right\}$$

and for discount factor, the values

$$\{0,01\,, \quad 0,1\,, \quad 0,2\,, \quad 0,3\,, \quad 0,4\,, \quad 0,5\,, \quad 0,6\,, \quad 0,7\,, \quad 0,8\,, \quad 0,9\,, \quad 0,99\}$$

were tested. Because of the large number of combinations, these were only tested with 500 runs per combination to see where the trend is going in the right direction. During each training run ϵ decreases from the initial $1{,}0$ so that after $80\,\%$ of the runs the value is $0{,}01$ and remains so until the end of the training.

The route of the ego vehicle for training was initially chosen like this: It always starts on way 1 near node 1 then proceeds along ways 5, 2, 3, 3, and 4 back to way 1 (see Figure 4-4). Along this route, the behavior

[87] Sola, J.; Sevilla, J.: Importance of input data normalization (1997).

planner must maintain various maximum speeds, stop for a time step at two stop signs, avoid a collision with the pedestrian, and perform a lane change at one point to avoid hitting an obstacle. In addition, in most places the ego vehicle must be careful not to leave the lane by making a lane change.

After the large-scale search, the search space for the learning rate was narrowed down to a range from $1,1 \cdot 10^{-5}$ to $1,5 \cdot 10^{-5}$, since the best results were obtained in this area. For the discount factor, only the extreme values $0{,}01$ and $0{,}99$ were rejected. Beyond that, no clear trend was identified for the discount factor so far. With the restricted search space, further training units were carried out with a run number of 1500 and then selected hyperparameter combinations were trained again with $10\,000$ runs. For this, however, a shorter route was chosen to reduce the complexity of the task for the behavior planner with the idea that this would allow faster success of the training to be recognized. As a result, the ego vehicle traveled only the left part of the map as shown in Figure 4-4. There is no need for a lane change at any point.

In the following section the result of the training is evaluated.

8 Evaluation

Subsection 7.5.2 describes the procedure followed to find a combination of hyperparameters that leads to the best training result in the described search space. Finally, it turned out that a learning rate of $1.4 \cdot 10^{-5}$ and discount factor of $0{,}7$ is such a combination.

Figure 8-1 shows in orange the progression of the rewards achieved over the $10\,000$ runs with the mentioned hyperparameters. The orange curve is averaged over 100 runs using moving average algorithm. The same graph shows in blue how the exploration exploitation trade-off value ϵ gets smaller over time. Figure 8-2 is similar to figure 8-1, but here the reward per time step is shown. Figure 8-3 shows in orange the distance traveled by the ego vehicle. The black dashed lines symbolize the longitudinal position at which the crosswalk is located on this round trip. This position is $62\,\mathrm{m}$ measured from the start position of the ego vehicle. The round trip is $296\,\mathrm{m}$ long. The reward per step curve in figure 8-2 and the distance curve in figure 8-3 are also averaged 100 runs.

At the beginning of the training process ϵ has the value 1. This means that every action was chosen randomly. Therefore, in the beginning often the ego vehicle leaves the lane due to a lane change and the training run is aborted. Thus, the behavior planner can hardly gain reward per lap. Thus, the reward per step is initially still in the negative range. But in the range where ϵ decreases, reward per step increases almost linearly.

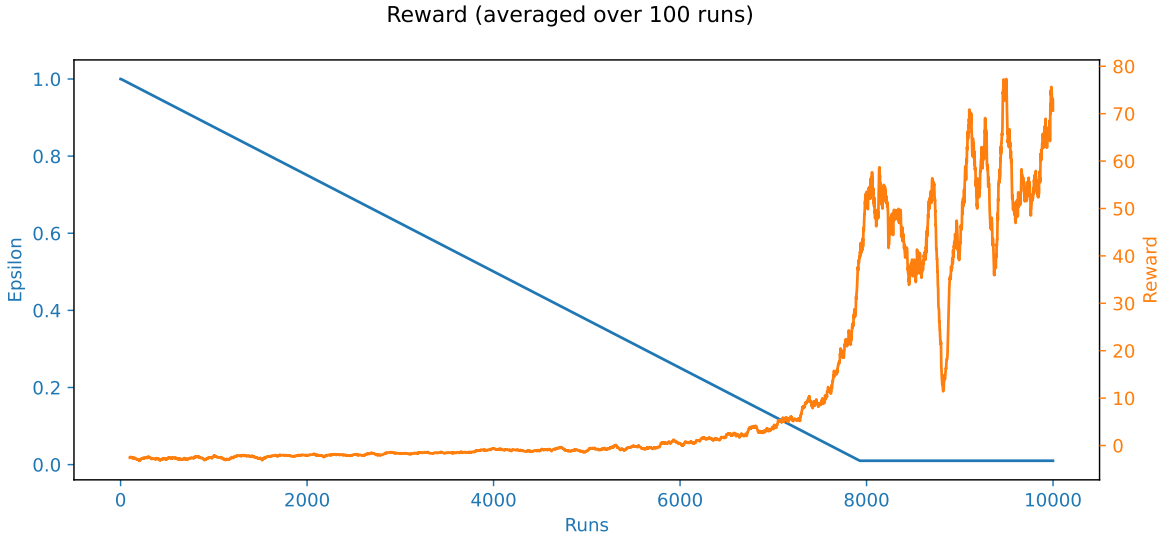


Figure 8-1: Progression of reward during training over $10\,000$ runs.

In the first approx. 7500 runs, the ego vehicle only moves rarely beyond the position of the crosswalk. (In figure 8-3 this is not visible at all, because here the values are averaged over 100 runs, but this was examined without smoothing). After that, the distance and reward jumps up. This suggests that from this moment on, the behavior planner has learned, to a first approximation, to avoid a collision with the randomly appearing pedestrian. However, this cannot be said with certainty at this point, since other factors may also have an influence on the rise. The assumption must be checked later in the visualization.

Such another factor, but one that definitely has an influence on the distance, is ϵ, which has the value $0{,}01$ starting from 8000 runs and keeps it since then. Which means that starting from this moment, every

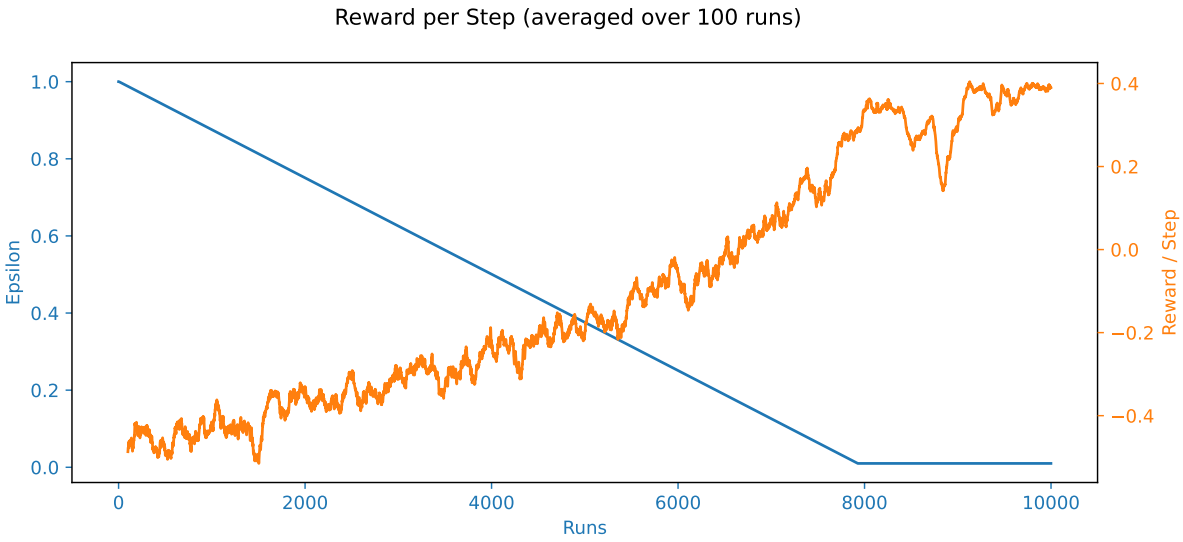


Figure 8-2: Progression of reward per step during training over 10 000 runs.

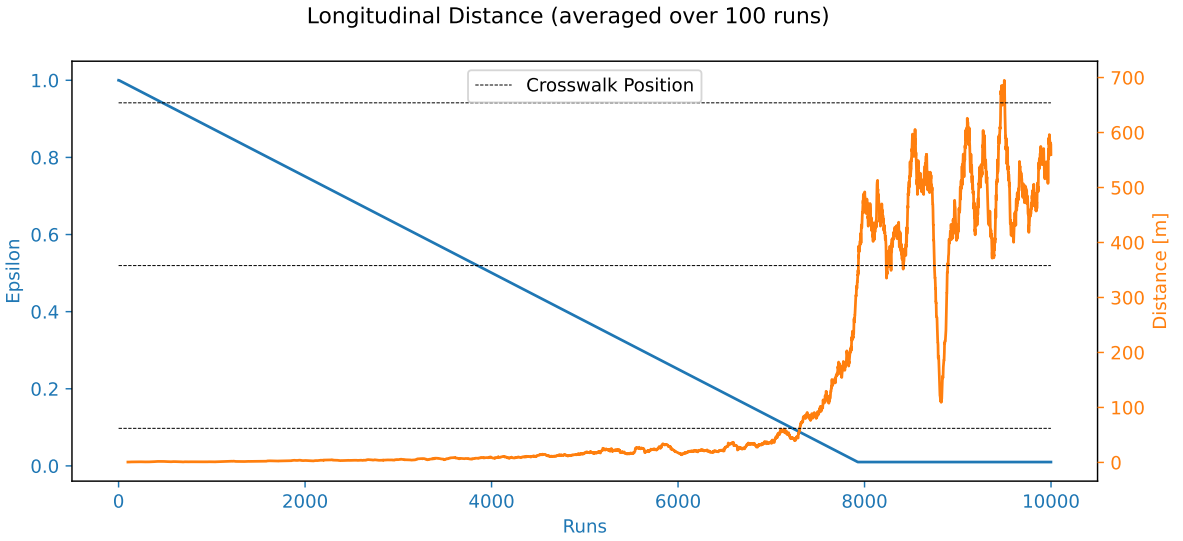


Figure 8-3: Progression of traveled distance during training over 10 000 runs.

hundredth action will be randomly selected. Reward per step continues to increase. After $10\,000$ runs, it reaches the value $0{,}4$ in the mean of the last hundred runs. So on average, significantly more steps are faultless. An upward trend can also be seen in the other two graphs, even if the values are exposed to large fluctuations.

This behavior model was further trained for 7500 runs to see how the reward evolves at $\epsilon = 0.01$. In fact, the trend continues, although not quite as strongly as before, as can be seen in figure 8-4, 8-5 and 8-6. A moving average of 1000 was chosen for these figures to highlight the trend behind the fluctuations.

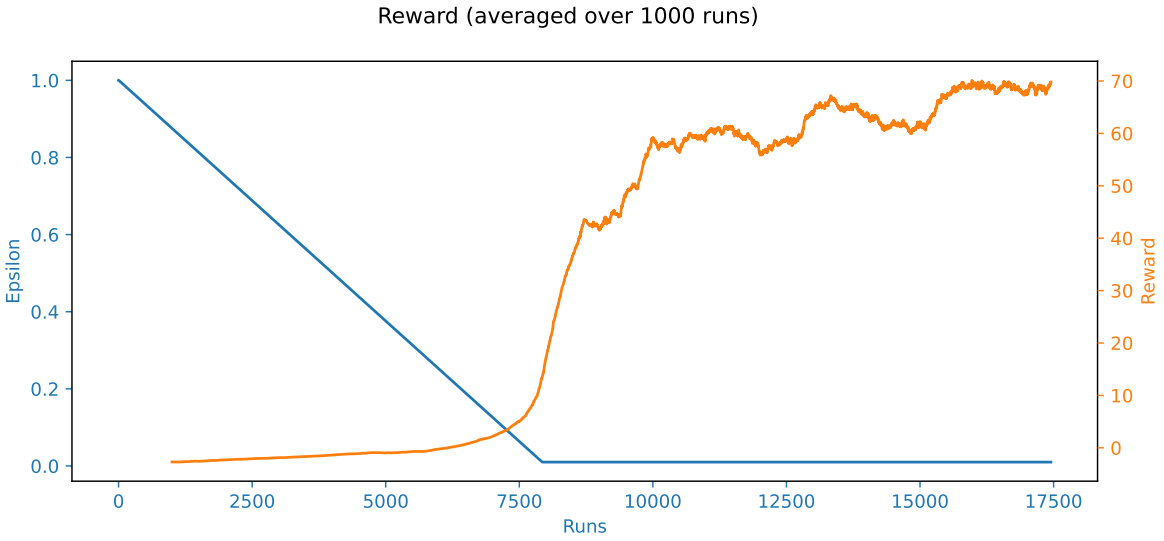


Figure 8-4: Progression of reward during training over $17\,500$ runs.

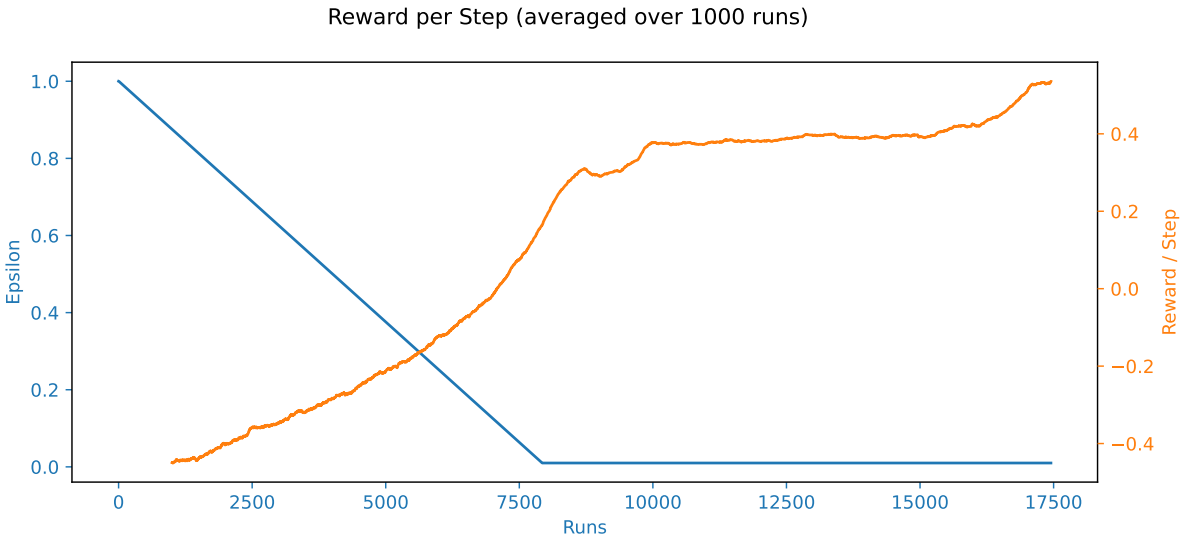


Figure 8-5: Progression of reward per step during training over $17\,500$ runs.

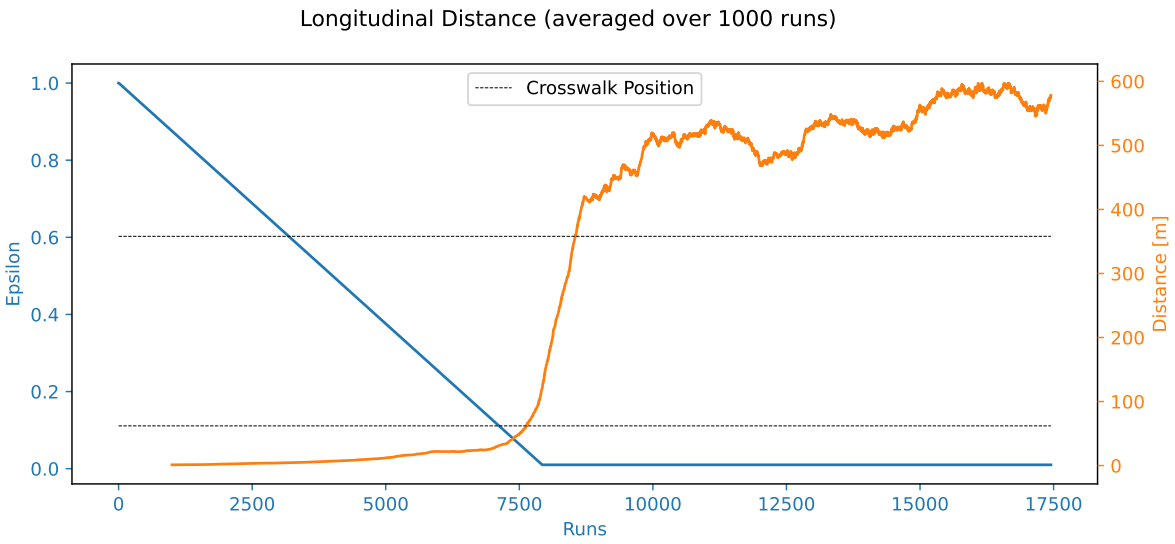


Figure 8-6: Progression of traveled distance during training over 17 500 runs.

At this point it should be noted that the evaluation of the behavior planner is done on the same route that is used for training. This creates a major risk of overfitting. As an example for this use case, overfitting could lead the behavior planner not to use the entry in the input vector for a stop condition to stop shortly before entering the area, but it uses the way, segment and lane ID of the area to identify a stop condition. If the behavior planner were then applied to a route that it doesn't know from its training, it would no longer be able to meet this behavior limit. Overfitting is a problem especially if the architecture of the NN is to be evaluated. Here, however, falsification criteria are to be evaluated. Even without separation between training and evaluation set, it can be said that the behavior planner learns from the reward derived from the falsification criteria and gets better over time.

Subsequent to the evaluation on the basis of the diagrams, the behavior model, which had been trained over 17 500 runs, was tested in the visualization and thus evaluated even further.

It turned out that the result regarding collision avoidance is insufficient. It is not evident that the ego vehicle reacts to the appearance of the pedestrian. This contradicts the above assumption that the behavior planner would have learned to avoid a collision. The ego vehicle can only pass the crosswalk if, due to the random components in the pedestrian simulator, it is not on the ego vehicle's trajectory. This surprisingly negative finding was reason to check the method of generating the input vector one more time especially regarding the mapping of TP. Also the transmission of the reward value to the behavior planner was checked again. No deviations from the target functionalities were noticed.

Also, a corresponding behavior for the stop condition of longitudinal entry regulation was not learned even after 17 500 runs. Here, this is even easier for the behavior planner than collision avoidance, since no random component such as a traffic light influences the reward for area entry. Also in this respect, the re-examination of the method of generating the input vector did not reveal any abnormality. Also the normally unwanted option of overfitting to other parameters, such as the way, segment and lane ID of the area with stop condition did not help the behavior planner.

During the whole test in the visualization, the behavior planner did no lane change. Obviously, it understood that this leads on the chosen route to a negative reward in most scenes, but does not provide any advantage

in any scene. In this case, complying with the falsification criterion is particularly easy due to this fact. If a lane change is initialized manually for the ego vehicle and then the behavior planner is asked for the next actions again, then the change process is aborted before completion in most tested cases. If the ego vehicle steers towards a scene that would lead to the fulfillment of the falsification criterion for lane change restrictions by a manually initialized lane change, the behavior planner chooses the stop action more often than otherwise. Obviously, therefore, a strategy was developed to avoid such a negative reward.

The execution of the model in the visualization also shows that the speed limits are observed at all times. The ego vehicle accelerates when the maximum allowed speed is increased and brakes in time before entering a new area where it is no longer allowed to drive as fast as before. On some sections of the route, however, the speed of the vehicle fluctuates as it switches between the highest permitted speed level and one level below each time step. On other route sections, however, it drives at exactly the maximum allowed speed for several time steps and thus receives the highest reward value. If the stop or slower action is manually selected in a time step, the behavior planer accelerates again to increase the reward. On the other hand, the planner reduces the speed if it was increased manually.

8.1 Discussion

After the evaluation of the training results the discussion follows. The starting point of it is the learned behavior in certain scenes. This is followed by a discussion of selected previous work steps that led to this result.

The previously described poor result regarding collision possibly comes from the fact that due to the random component of the pedestrian simulator, learning a correct reaction is more difficult and takes longer than previously trained. In comparison, ϵ drops too quickly. With a pedestrian only randomly occurring and an average of only one out of one hundred random actions at the end, there is probably too little search for better behaviors for this use case. Nevertheless, such a random component should not be omitted, since this avoids overfitting and makes the behavior planner more robust to different scenes. Also a probably insufficient architecture of the NN can have a bad influence on it. The behavior planner of Hoel et al.[88] uses a much more complex architecture.

Another issue that probably complicates collision avoidance is the way other TP are mapped in the input vector. Hoel et al.[88] and Fridman et al.[89] both describe the positions of other TP in relation to the ego vehicle. In the implementation of the BSSD behavior planner, the positions of all TP are described using way, segment, and lane ID as well as offset within an area. This requires the NN to first establish an awareness of the spatial context. This has apparently not happened conclusively in the $17\,500$ runs and may not happen at all with the simple architecture of the NN.

With the help of visualization, by using debugging tools for the program code and also by evaluating the meta data stored during the training process, it was ensured that the falsification criterion for collision is fulfilled in the right scenes and provides a negative contribution to the overall reward. But because of the poor result in the training of a collision avoidance, the question arises whether instead of a boolean collision reward a score depending on the severity of the accident would be more purposeful. The same

[88] Hoel, C.-J. et al.: Planning and Deep Reinforcement Learning (2020).

[89] Fridman, L. et al.: DeepTraffic (2018).

question arises from the consistent non-compliance with the stop condition. The sub-falsification criterion of longitudinal boundary regulation is fulfilled in the appropriate scenes, but it does not lead to the desired behavior. Training success could probably be accelerated by adjusting the Reward Module. On the other hand, a positive example is the choice of the appropriate speed by the behavior planner. The combination of the positive reward due to the vehicle speed in relation to the permissible speed and the negative reward for over-speeding leads to a comparatively very pleasing result.

An even more valid reason for not obeying the stop condition and not avoiding the collision is that stopping is not profitable in total. Stop condition example: In the previous area, the maximum allowed speed is $30\,\mathrm{km/h}$ and also in the area of this condition. As shown in table 8-1, for the braking and acceleration maneuver, the ego vehicle needs 11 time steps in the best case. Thereby the behavior planner would only get a reward of $5{,}32$ in 11 time steps to avoid a reward of -1 for violating stop condition. On the other hand, if it drives for 11 time steps with maximum allowed speed and disobeys the stop condition, the reward is $10 - 1 = 9$. Thus, stopping does not pay off at all.

Table 8-1: Reward progress for observing the stop condition based on the scenery used in the training process.

Time Step	Allowed Speed [$\mathrm{km/h}$]	Action	Current Speed [$\mathrm{km/h}$]	Reward
-1	30		30	
0	30	STOP	12	0,4
1	(Stop Condition)	STOP	0	0
2	30	FASTER	2	0,07
3	30	FASTER	5,6	0,19
4	30	FASTER	9,2	0,31
5	30	FASTER	12,8	0,43
6	30	FASTER	16,4	0,55
7	30	KEEP	20	0,67
8	30	KEEP	23,6	0,79
9	30	KEEP	27,2	0,91
10	30	KEEP	30	1
Sum				5,32

With the selected training route, it was only able to cover a small section of the formalized and implemented falsification criteria. Therefore the criteria for obstruction, longitudinal boundary regulation including the remaining subfalsification criteria for no stagnant traffic and not red light, as well as the criterion for overtake regulation was not yet be tested in the direct application by the training procedure. In addition to a more complex route, there is also a lack of other simulated TP, such as other motor vehicles, but also simulations of traffic lights. This would also imply that the NN increases and the training process has to be extended to deal with the input vector, which then has to include several other TP as well as reservation and reservation links.

In retrospect, the question also arises whether it makes sense to provide the behavior planner besides the slower action also the stop action. Without the stop action there would be less combinations of state and action and makes the search space and the training cost smaller. This would deprive the ego vehicle of the

option of *emergency braking*, but with careful driving the stop action can probably also be covered by the slow action.

In order to put the road around the ego vehicle into a form that can be processed by a NN, the method with the predefinition of a route and the segment sequence created by this has proven to be practicable. The segment sequence can be easily converted into an input vector thanks to its hierarchical structure (way – segment – lane) and thanks to the requirement of BSSD specification that the required behavior must be the same within an area. Nevertheless, the segment sequence also has its weaknesses. As described earlier, the use of way, segment, and lane ID probably makes it difficult to generate a spatial understanding between TP located on the specified areas. In addition, the input vector currently includes only one position per TP, even if it is located on multiple areas at the same time. Furthermore, if a TP that the ego vehicle must give priority to is not on ego's segment sequence and also not on the area specified in reservation link, the behavior planner has no chance to find a meaning behind an upcoming penalty. For example, this would be the case if the linked area is so short that the ego vehicle must also give priority to a TP that is only on this area in the near future. But as it is implemented at this moment, the behavior planner does not know how to place such a TP spatially.

Another problem regarding the input vector and the used NN architecture is that the input vector must always have the same size. The number of TP, segments in the sequence, areas per segment, reservations per area and links per reservation cannot dynamically adapt to individual scenes. The maximum number of such elements that can be included in the vector must be determined before training begins. Elements exceeding this limit will be discarded. If fewer elements are needed in a scene, then the corresponding places in the vector are filled with the placeholder number 1. It is difficult to find a good trade off between correct FoV and a vector that is not too large.

The falsification criteria which cannot be used directly in the training process were tested manually. For this purpose, the visualization with the illustration of the segment sequence and reward list and the control function of the ego vehicle was proven to be very helpful. The use of test routines was also very beneficial in order to be able to check over the entire duration of the project work whether the desired results are still achieved after changes have been made.

Even though the implementation of the falsification criterion for obstraction was not used in the direct application in the training process, it became apparent during implementation and testing that the variant using a tree search (see subsubsection 7.2.2) works to a certain extent, but is not yet fully functional. Some assumptions had to be made that are not practical. For example, the behavior planner is already penalized, if a TP might have a collision with the ego vehicle based on its current position and speed without checking how likely it is, that the TP will choose that route anyway. When checking obstruction of TP with higher right, it is unfortunate that the ego vehicle has to keep the whole area clear to escape a penalty. If the area is very long, it is quite possible that both TP are in it without the ego vehicle obstructing the other TP. Furthermore, currently the falsification criterion is incorrectly not fulfilled if the TP is already on the area where it has higher reservation right. So far, only reservation links are used to search for TP that are on the way to this area. Additionally, the algorithm for determining TP with the same reservation right as the ego vehicle is not fully correct.

Also for the implementation of the no stagnant traffic condition of the longitudinal boundary regulation it was only able to test it manually. Nevertheless, it turned out that the threshold of $40\,\%$ driving speed

in relation to the maximum permissible speed of this area, which was already introduced during the formalization (see subsubsection 6.3.2), is less useful. Areas with such a condition are usually found in nodes. There, however, TP often drive at a lower speed, e.g. because they have just accelerated in front of a traffic light. By the used definition, the traffic would almost always be stagnant as soon as another TP is in the area.

During the implementation, the synthetic road map created in the beginning was helpful for testing the falsification criteria as well as functions of other components of the software project at different locations in the map. For the training of the behavior planner, however, only a section of the map was chosen in the end in order to reduce complexity. Thanks to the intersections and thus the possibility to use different routes, it was possible to quickly choose a round course in this map, which represents few complex scenes.

In conclusion, it turned out that the architecture of the software project, as defined before the start of implementation, was well implementable and basically works as expected, even if improvements are needed in some places. The most important points are listed in the next subsection.

Besides the conclusion of the work within the scope of this thesis, a résumé of BSSD can also be formulated at this point. It was shown that most of the falsification criteria are derivable from the BSSD specification without problems of interpretability and formalized in such a way, that they are correctly fulfilled or not fulfilled in scenes close to reality. Good machine readability is given at these points. This was achieved despite the fact that the specification has a strong abstraction of behavioral constraints. However, ambiguities and room for improvement were identified for a few parts, which made the machine interpretability and formalization work more difficult. The suggestions for improvement regarding the specification are also listed below.

8.2 Further Work

From the previous discussion, recommendations for further work on the RL behavior planner based on BSSD are emerging at this point.

Special attention should be paid to improving the implementation of the falsification criterion for obstruction. Instead of further development of the used tree search variant, it makes more sense to focus on the method using the comparison of the trajectories, as described in subsection 6.2. Only in this way it can really be determined whether a TP is obstructed by the ego vehicle. But therefore the tolerance value for the deviation from theoretical to actual trajectory, as well as the time in the future at which the points on the trajectories are to be compared, must be determined beforehand, e.g. with an empirical perceptual study. In addition also further development of the environment module is required, so that each TP has a theoretical and an actual trajectory.

Furthermore, regarding the falsification criterion for collision, it should be considered that only foreseeable collisions should be penalized, as described in subsection 6.1. The threshold of foreseeability for this must still be found. To be able to detect collisions in intersections, further geometry data of the map must be included in the environment. Then the falsification criterion can be implemented as formalized. To model overtaking within lanes and to be able to implement the corresponding falsification criterion as formalized like described in subsection 6.6, a lateral resolution within areas also must be provided, too.

The evaluation of the behavior planner showed that in the case of the stop condition and collision, it is not profitable to pay attention to them, because more reward is lost by doing so. Therefore, the negative reward values for fulfilling the falsification criteria should be adjusted to increase the incentive to comply with the behavioral constraints. If this is still not sufficient, it should be checked whether comparable to the positive reward depending on the speed driven in relation to the maximum allowed speed, for example, collision avoidance can be learned better, if the collision speed is included in the evaluation as a measure for the severity of an accident. This could also guide the behavior planner in the intended direction when searching for the right way dealing with behavioral constraints. It is also conceivable to give positive rewards for complying with certain rules, for example stopping at a stop sign. Additionally, it should be examined whether it is more appropriate to train the behavior step by step, i.e. to start with a simple rule and then gradually add others.

Another important point for the successful further development of this project is the RL behavior planner itself. To overcome the problem with the fixed input vector, the architecture of the NN should be rebuilt to handle an input that dynamically changes in length. Such NN are for example convolutional neural networks or recurrent neural networks. This would then also allow to put parts of the map outside the segment sequence into the NN, which would allow the behavior planner to react to TP which are not on the sequence from the ego vehicle. Further investigation is needed with different depths of the NN or sizes of the layers. Moreover, Hoel et al.[90] use a bypass that sends certain information around some layers. A discount factor varying over time are also possibilities to improve the result.[91] Regarding the exploration and exploitation trade-off value, it can be tried whether a nonlinear decay or an event-based change would be better.

Efforts to avoid overfitting and instead make the behavior more robust would also be particularly important. For this, the ego vehicle should not always start at the same position and always take the same route. It is even conceivable that a road map is randomly generated to confront the behavior planner with different combinations of BSSD elements. As described in subsection 3.2, Qin et al.[92] showed the advantage of adversarial agents in a simulation environment for RL. They trick the ego vehicle into critical scenarios to provoke the fulfillment of falsification criteria and deliberately train the behavior planner for such scenarios. Such agents can be other TP or traffic lights, but theoretically can also be a map generator. A method for selecting critical driving scenarios for automated vehicles based on an objective characterization of driving behavior was elaborated by Schwab[93] in his thesis. Junietz et al.[94] also focused on the metric for evaluating the criticality of traffic situations and scenarios in a paper. Furthermore, the extension of the simulation environment to a multi-agent system, in which several vehicles are moving, controlled by the behavior planner, can be used to represent more different scenes and to make the planner more robust. In any case, to evaluate the behavior planner should not continue to use the same routes which were used for training.

Improvements can also be made for the translation of actions into actual movements. In the current implementation, if a lane change is interrupted prematurely, the ego vehicle is immediately only in the

[90] Hoel, C.-J. et al.: Planning and Deep Reinforcement Learning (2020).

[91] François-Lavet, V. et al.: How to Discount Deep Reinforcement Learning (2015).

[92] Qin, X. et al.: Falsification with Dynamically Constrained Reinforcement Learning (2020).

[93] Schwab, A.: Master Thesis, Methode zur Auswahl kritischer Fahrszenarien (2019).

[94] Junietz, P. et al.: Metrik zur Bewertung der Kritikalität (2017).

original area. Moving back to the center of the original area could be implemented in the future to make the environment more realistic for the RL. Furthermore, the functionality used could be extended from CC to ACC to ensure that the ego vehicle maintains the correct distance for follow maneuvers in the future. This is also used by Hoel et al.[95] together with the Intelligent Driver Model, and Fridman et al.[96] together with the Safety System. Hence, they only let the RL behavior planner do the higher level motion planning, so it can focus on it, as described in subsection 3.2. This relieves the training process.

In addition to the options for further development of the RL behavior planner, during the work on this thesis, parts of the `BSSD` specification[97] have emerged that could also be improved. First, this document currently offers room for interpretation regarding the foreseeability of a collision and the definition of an obstruction. Therefore, the falsification criteria could not be finally formalized. Instead, metrics were developed to assess foreseeability and an obstruction. Further guidance from the BSSD specification would be helpful to complete the formalization. With regard to obstruction, in future it should be included, following the description from PolFHa, that only significant obstruction must be avoided, as described in subsection 6.2 in more detail. Finally, the no stagnant traffic condition can still be reworded to be more consistent with its actual meaning. For entering such an area, such as in an intersection, the primary deciding criterion is whether the ego vehicle can be expected to pass through that area without facing restrictions on its trajectory. Whether there are other `TP` within the area, which form a stagnant traffic flow, is only of secondary importance.

[95] Hoel, C.-J. et al.: Planning and Deep Reinforcement Learning (2020).

[96] Fridman, L. et al.: DeepTraffic (2018).

[97] Glatzki, F.; Lippert, M.: BSSD Specification (2021).

9 Conclusion

In this thesis, six falsification criteria were derived and formalized based on the specification of BSSD. These concern collision, obstruction, longitudinal and lateral boundary regulation of an area, speed limit and overtake regulation. They serve an online behavior planner based on RL to show the boundaries of the legal behavior space.

The formalization of four of the criteria was implemented without any problems. So, a good machine readability was determined to that extent. But regarding the criteria for collision and obstruction, difficulties occurred due to the fact that parts of the specification are not clearly defined and therefore are not fully machine-interpretable. First, the rule regarding collision states that only foreseeable collisions must be avoided, without defining more precisely what foreseeability is. Second, the term for obstruction is not defined. It was not possible to formalize these two criteria in an all-encompassing way. Instead, metrics were created to evaluate foreseeability and obstruction in these cases, for which thresholds will need to be defined in the future to complete the formalization of the criteria. Regarding collision, it was also found that going beyond the BSSD specification, only those collisions that could have been avoided should result in a penalty to the behavior planner. It also turned out that BSSD's longitudinal entry condition *no stagnant traffic* does not completely achieve its purpose.

Another task of this thesis was to implement a simple RL behavior planner that makes use of the formalized falsification criteria. For this purpose, a synthetic road map was first created and converted into the BSSD format using an assistance utility implemented before. Furthermore, a training environment including simulation of another TP and a visualization was created. As RL algorithm of the online behavior planner DDQN was used with an experience replay buffer and separate target and evaluation networks to increase the performance during training.

The reward value for the RL is derived, on the one hand, from the falsification criteria, which add negative contributions for each step outside the allowed behavioral space. On the other hand, each step in which no falsification criterion is met and in which the ego-vehicle is moving forward is rewarded with a positive contribution depending on the speed, in order to create an incentive to train a goal-directed behavior within the allowed behavior space. For the purpose of reward generation, the formalized falsification criteria were implemented to match the abstraction level of the training environment. To be able to implement the criterion for collision, the test for predictability was omitted. Since the criterion for obstruction was not be fully formalized due to the scope of this thesis and the described approach was not implemented consequently, another solution for the implementation had to be found. However, it turned out that the implemented solution is insufficient for the coverage of the criterion.

The scope of this thesis only allows the implementation of a simple behavior planner. It was not possible for this to satisfy all behavior rules equally. Thus, the training of the behavior planner was performed only on a map section in which it was supposed to avoid a collision with a randomly crossing pedestrian, not leaving the lane, to stop at a stop sign and not exceeding the maximum speed. For some falsification criteria, in addition to a more powerful behavior planner, more simulations of TP are needed.

The evaluation of the behavior planner based on the course of reward, reward per step and the distance covered by the ego-vehicle during a run showed that the behavior planner is quite capable of learning a behavior so that it can increase its success while training. To evaluate the result in more detail, the

behavior model resulting from the training process was examined more closely in the visualization. A more differentiated picture emerged. While the lane was properly not left and the speed range was well utilized without over-speeding, no consideration was given to the stop sign or to the pedestrian.

One reason for these failures was found in the fact that in total it is not profitable to brake, because the reduced speed has a greater influence than the negative reward due to the stop sign violation or the collision. Therefore, the penalty values of the individual falsification criteria must be adjusted in the future. Further suggestions for improvement were also given about how the behavior planner can be developed so that it can handle more specifications at the same time. For example, an architecture should be used that can deal with a variable length of the NN input.

Bibliography

Abrahamczyk, M.: Ab wann ist ein Stau ein Stau? (2020)
Abrahamczyk, Markus: Ab wann ist ein Stau ein Stau? t-online.de, URL: https://www.t-online.de/auto/recht-und-verkehr/id_85202538/ab-wann-ist-ein-stau-ein-stau-.html, 2020, visited on 04/07/2021

Ahmed, A. et al.: Formal Asymptotic Analysis (2018)
Ahmed, Asad; Hasan, Osman; Awwad, Falah; Bastaki, Nabil; Hasan, Syed Rafay: Formal Asymptotic Analysis of Online Scheduling Algorithms for Plug-In Electric Vehicles' Charging, in: Energies, Vol. 12, 2018

Buechel, M. et al.: Ontology-based traffic scene modeling (2017)
Buechel, Martin; Hinz, Gereon; Ruehl, Frederik; Schroth, Hans; Gyoeri, Csaba; Knoll, Alois: Ontology-based traffic scene modeling, traffic regulations dependent situational awareness and decision-making for automated vehicles, in: 2017 IEEE Intelligent Vehicles Symposium (IV), pp. 1471–1476, 2017

Bundesamt für Strassen: Definitionen (2018)
Bundesamt für Strassen: Definitionen. Bundesamt für Strassen ASTRA, URL: https://www.astra.admin.ch/astra/de/home/themen/nationalstrassen/verkehrsfluss-stauaufkommen/definitionen.html, 2018, visited on 04/07/2021

Bundesministerium der Justiz und für Verbraucherschutz: Straßenverkehrs-Ordnung (2013)
Bundesministerium der Justiz und für Verbraucherschutz: Straßenverkehrs-Ordnung vom 6. März 2013 (BGBl. I S. 367), die zuletzt durch Artikel 1 der Verordnung vom 18. Dezember 2020 (BGBl. I S. 3047) geändert worden ist, 2013

Bundesministerium für Verkehr und digitale Infrastruktur: Längenstatistik Straßen überörtlicher Verkehr (2020)
Bundesministerium für Verkehr und digitale Infrastruktur: Längenstatistik der Straßen des überörtlichen Verkehrs, Stand: 1. Januar 2020, URL: https://www.bmvi.de/SharedDocs/DE/Anlage/StB/laengenstatistik-2020.pdf?__blob=publicationFile, 2020, visited on 04/07/2021

Burg, H. et al.: Handbuch Verkehrsunfallrekonstruktion (2017)
Burg, Heinz; Moser, Andreas: Handbuch Verkehrsunfallrekonstruktion: Unfallaufnahme, Fahrdynamik, Simulation, ATZ/MTZ-Fachbuch, 3. Edition, Springer Vieweg, 2017

Donzé, A.: On Signal Temporal Logic (2014)
Donzé, Alexandre: On Signal Temporal Logic, URL: https://people.eecs.berkeley.edu/~sseshia/fmee/lectures/EECS294-98_Spring2014_STL_Lecture.pdf, 2014, visited on 11/26/2020

Dupuis, M.: OpenDRIVE (2006)
Dupuis, Marius: OpenDRIVE, URL: http://web.archive.org/web/20070102195854/http://www.opendrive.org/docs/OpenDRIVEFormatSpecRev1.0A.pdf, 2006, visited on 05/01/2021

Ertel, W.: Grundkurs Künstliche Intelligenz (2016)
Ertel, Wolfgang: Grundkurs Künstliche Intelligenz: Eine praxisorientierte Einführung, 4. Edition, Springer Fachmedien, 2016

Esterle, K. et al.: Formalizing Traffic Rules for Machine Interpretability (2020)
Esterle, Klemens; Gressenbuch, Luis; Knoll, Alois: Formalizing Traffic Rules for Machine Interpretability, in: 2020 IEEE 3rd Connected and Automated Vehicles Symposium (CAVS), 2020

François-Lavet, V. et al.: How to Discount Deep Reinforcement Learning (2015)
François-Lavet, Vincent; Fonteneau, Raphael; Ernst, Damien: How to Discount Deep Reinforcement Learning: Towards New Dynamic Strategies, in: Neural Information Processing Systems (NIPS 2015) Deep Reinforcement Learning Workshop, vol. 28, 2015

Fridman, L. et al.: DeepTraffic (2018)
Fridman, Lex; Terwilliger, Jack; Jenik, Benedikt: DeepTraffic: Crowdsourced Hyperparameter Tuning of Deep Reinforcement Learning Systems for Multi-Agent Dense Traffic Navigation, in: Neural Information Processing Systems (NIPS 2018) Deep Reinforcement Learning Workshop, vol. 31, 2018

Glatzki, F. et al.: BSSD Specification (2021)
Glatzki, Felix; Lippert, Moritz: Behavior-Semantic Scenery Description v0.3 Draft, 2021, visited on 02/25/2021

Glatzki, F. et al.: Behavioral Attributes for a BSSD (2021)
Glatzki, Felix; Lippert, Moritz; Winner, Hermann: Behavioral Attributes for a Behavior-Semantic Scenery Description (BSSD) for the Development of Automated Driving Functions, in: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), pp. 667–672, 2021

Glorot, X. et al.: Deep Sparse Rectifier Neural Networks (2011)
Glorot, Xavier; Bordes, Antoine; Bengio, Yoshua: Deep Sparse Rectifier Neural Networks, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, vol. 15, pp. 315–323, 2011

Harmon, M. E. et al.: Reinforcement Learning (1997)
Harmon, Mance E.; Harmon, Stephanie S.: Reinforcement Learning: A Tutorial, URL: http://www.dtic.mil/docs/citations/ADA323194, 1997, visited on 05/01/2021

Hasselt, H. v.: Double Q-learning (2010)
Hasselt, Hado van: Double Q-learning, in: Neural Information Processing Systems (NIPS 2010) Deep Reinforcement Learning Workshop, vol. 23, pp. 2613–2621, 2010

Hasselt, H. v. et al.: Deep Reinforcement Learning with Double Q-learning (2016)
Hasselt, Hado van; Guez, Arthur; Silver, David: Deep Reinforcement Learning with Double Q-Learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 30, pp. 2094–2100, 2016

Hesse, W.: Ontologie(n) (2002)
Hesse, Wolfgang: Ontologie(n), in: Informatik-Spektrum, Vol. 25, pp. 477–480, 2002

Hessel, M. et al.: Rainbow (2018)
Hessel, Matteo; Modayil, Joseph; Hasselt, Hado van; Schaul, Tom; Ostrovski, Georg; Dabney, Will; Horgan, Dan; Piot, Bilal; Azar, Mohammad; Silver, David: Rainbow: Combining Improvements in Deep Reinforcement Learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, pp. 3215–3222, 2018

Hoel, C.-J. et al.: Planning and Deep Reinforcement Learning (2020)
Hoel, Carl-Johan; Driggs-Campbell, Katherine; Wolff, Krister; Laine, Leo; Kochenderfer, Mykel J.: Combining Planning and Deep Reinforcement Learning in Tactical Decision Making for Autonomous Driving, in: IEEE Transactions on Intelligent Vehicles, Vol. 5, pp. 294–305, 2020

Junietz, P. et al.: Metrik zur Bewertung der Kritikalität (2017)
Junietz, Philipp; Schneider, Jan; Winner, Hermann: Metrik zur Bewertung der Kritikalität von Verkehrssituationen und -szenarien, in: 2017

Karg, J.: ADAC-Chef August Markl (2019)
Karg, Josef: ADAC-Chef August Markl: Das ist der oberste Gelbe Engel. Augsburger Allgemeine, URL: https://www.augsburger-allgemeine.de/wirtschaft/ADAC-Chef-August-Markl-Das-ist-der-oberste-Gelbe-Engel-id55963926.html, 2019, visited on 04/27/2021

Karimi, A. et al.: Formalizing traffic rules for uncontrolled intersections (2020)
Karimi, Abolfazl; Duggirala, Parasara Sridhar: Formalizing traffic rules for uncontrolled intersections, in: 2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS), pp. 41–50, 2020

Kingma, D. P. et al.: Adam (2015)
Kingma, Diederik P.; Ba, Jimmy Lei: Adam: A Method for Stochastic Optimization, in: 3rd International Conference on Learning Representations (ICLR 2015), 2015

Kohlhaas, R. et al.: Semantic state space for high-level maneuver planning (2014)
Kohlhaas, Ralf; Bittner, Thomas; Schamm, Thomas; Zollner, J. Marius: Semantic state space for high-level maneuver planning in structured traffic scenes, in: 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), pp. 1060–1065, 2014

Kraftfahrt-Bundesamt: BT-KAT-OWI (2020)
Kraftfahrt-Bundesamt: Bundeseinheitlicher Tatbestandskatalog Straßenverkehrsordnungswidrigkeiten, 13. Edition, Kraftfahrt-Bundesamt, 2020

Lin, L.-J.: Diss., Reinforcement Learning for Robots (1993)
Lin, Long-Ji: Reinforcement Learning for Robots Using Neural Networks, Dissertation School of Computer Science Carnegie-Mellon University Pittsburgh, 1993

Lippert, M. et al.: BSSD of Road Networks for Automated Driving (2022)
Lippert, Moritz; Glatzki, Felix; Winner, Hermann: Behavior-Semantic Scenery Description (BSSD) of Road Networks for Automated Driving, 2022

Maurer, M. et al.: Autonomes Fahren (2015)
Maurer, Markus; Gerdes, J. Christian; Lenz, Barbara; Winner, Hermann: Autonomes Fahren: Technische, rechtliche und gesellschaftliche Aspekte, 1. Edition, Springer Vieweg, 2015

Mnih, V. et al.: Playing Atari with Deep Reinforcement Learning (2013)
Mnih, Volodymyr; Kavukcuoglu, Koray; Silver, David; Graves, Alex; Antonoglou, Ioannis; Wierstra, Daan; Riedmiller, Martin: Playing Atari with Deep Reinforcement Learning, in: Neural Information Processing Systems (NIPS 2013) Deep Reinforcement Learning Workshop, vol. 26, 2013

Mnih, V. et al.: Human-level control through deep reinforcement learning (2015)
Mnih, Volodymyr; Kavukcuoglu, Koray; Silver, David; Rusu, Andrei A.; Veness, Joel; Bellemare, Marc G.; Graves, Alex; Riedmiller, Martin; Fidjeland, Andreas K.; Ostrovski, Georg; Petersen, Stig; Beattie, Charles; Sadik, Amir; Antonoglou, Ioannis; King, Helen; Kumaran, Dharshan; Wierstra, Daan; Legg, Shane; Hassabis, Demis: Human-level control through deep reinforcement learning, in: Nature, Vol. 518, pp. 529–533, 2015

Oberlandesgericht Brandenburg: 1 Ss (OWi) 86 B/05 (2005)
Oberlandesgericht Brandenburg: 1 Ss (OWi) 86 B/05, URL: https://dejure.org/dienste/vernetzung/rechtsprechung?Gericht=OLG%20Brandenburg&Datum=23.05.2005&Aktenzeichen=1%20Ss%20OWi%2086%20B/05, 2005, visited on 04/28/2021

Oberlandesgericht Braunschweig: 2 U 34/90 (1990)
Oberlandesgericht Braunschweig: 2 U 34/90, URL: https://dejure.org/dienste/vernetzung/rechtsprechung?Text=VersR%201992,%20841, 1990, visited on 04/28/2021

Oberlandesgericht Karlsruhe: 1 Ss 159/03 (2004)
Oberlandesgericht Karlsruhe: 1 Ss 159/03, URL: https://dejure.org/dienste/vernetzung/rechtsprechung?Gericht=OLG%20Karlsruhe&Datum=14.04.2004&Aktenzeichen=1%20Ss%20159/03, 2004, visited on 04/28/2021

Oberlandesgericht Köln: Ss 782/84 (1985)
Oberlandesgericht Köln: Ss 782/84, URL: https://dejure.org/dienste/vernetzung/rechtsprechung?Gericht=OLG%20K%F6ln&Datum=22.01.1985&Aktenzeichen=Ss%20782/84, 1985, visited on 04/28/2021

OpenAI: Gym (2016)
OpenAI: Gym, URL: https://gym.openai.com, 2016, visited on 05/01/2021

Parkinson, A.: Epsilon-Greedy Algorithm for Reinforcement Learning (2019)
Parkinson, Avery: The Epsilon-Greedy Algorithm for Reinforcement Learning. Medium, URL: https://medium.com/analytics-vidhya/the-epsilon-greedy-algorithm-for-reinforcement-learning-5fe6f96dc870, 2019, visited on 05/01/2021

Poggenhans, F. et al.: Lanelet2 (2018)
Poggenhans, Fabian; Pauls, Jan-Hendrik; Janosovits, Johannes; Orf, Stefan; Naumann, Maximilian; Kuhnt, Florian; Mayr, Matthias: Lanelet2: A high-definition map framework for the future of automated driving, in: 2018 IEEE 21st International Conference on Intelligent Transportation Systems (ITSC), 2018

Qin, X. et al.: Falsification with Dynamically Constrained Reinforcement Learning (2020)
Qin, Xin; Aréchiga, Nikos; Best, Andrew; Deshmukh, Jyotirmoy: Automatic Testing and Falsification with Dynamically Constrained Reinforcement Learning, in: arXiv:1910.13645 [cs, eess], 2020

Rizaldi, A. et al.: Formalising and Monitoring Traffic Rules (2017)
Rizaldi, Albert; Keinholz, Jonas; Huber, Monika; Feldle, Jochen; Immler, Fabian; Althoff, Matthias; Hilgendorf, Eric; Nipkow, Tobias: Formalising and Monitoring Traffic Rules for Autonomous Vehicles in Isabelle/HOL, in: Polikarpova, Nadia; Schneider, Steve (Eds.): Integrated Formal Methods, Springer International Publishing, 2017

Rossum, G. v. et al.: PEP 484 (Type Hints) (2014)
Rossum, Guido van; Lehtosalo, Jukka; Langa, Lukasz: Type Hints, URL: https://www.python.org/dev/peps/pep-0484/, 2014, visited on 04/28/2021

Schreier, M.: Diss., Environment Representation, Prediction, Criticality Assessment (2015)
Schreier, Matthias: Bayesian Environment Representation, Prediction, and Criticality Assessment for Driver Assistance Systems, Dissertation Technische Universität Darmstadt, 2015

Schwab, A.: Master Thesis, Methode zur Auswahl kritischer Fahrszenarien (2019)
Schwab, Alexander: Eine Methode zur Auswahl kritischer Fahrszenarien für automatisierte Fahrzeuge anhand einer objektiven Charakterisierung des Fahrverhaltens, Master Thesis Technische Universität München, 2019

Schweizerischer Bundesrat: Verkehrsregelnverordnung (2021)
Schweizerischer Bundesrat: Verkehrsregelnverordnung, 2021

Shalev-Shwartz, S. et al.: Formal Model of Safe Self-driving Cars (2018)
Shalev-Shwartz, Shai; Shammah, Shaked; Shashua, Amnon: On a Formal Model of Safe and Scalable Self-driving Cars, in: arXiv:1708.06374 [cs, stat], 2018

Silver, D. et al.: Mastering Go without human knowledge (2017)
Silver, David; Schrittwieser, Julian; Simonyan, Karen; Antonoglou, Ioannis; Huang, Aja; Guez, Arthur; Hubert, Thomas; Baker, Lucas; Lai, Matthew; Bolton, Adrian; Chen, Yutian; Lillicrap, Timothy; Hui, Fan; Sifre, Laurent; Driessche, George van den; Graepel, Thore; Hassabis, Demis: Mastering the game of Go without human knowledge, in: Nature, Vol. 550, pp. 354–359, 2017

Sola, J. et al.: Importance of input data normalization (1997)
Sola, J.; Sevilla, J.: Importance of input data normalization for the application of neural networks to complex industrial problems, in: IEEE Transactions on Nuclear Science, Vol. 44, pp. 1464–1468, 1997

Sovani, S.: Simulation beschleunigt Entwicklung des autonomen Fahrens (2017)
Sovani, Sandeep: Simulation beschleunigt Entwicklung des autonomen Fahrens, in: ATZ - Automobiltechnische Zeitschrift, Vol. 119, pp. 26–33, 2017

Sporrer, A. et al.: Realsimulation von Spurwechselvorgängen im Straßenverkehr (1998)
Sporrer, A; Prell, G; Buck, J; Schaible, St: Realsimulation von Spurwechselvorgängen im Straßenverkehr, in: Verkehrsunfall und Fahrzeugtechnik, Vol. 36, pp. 69–76, 1998

Stryk, O. v.: Grundlagen der Robotik (2018)
Stryk, Oskar von: Grundlagen der Robotik, Fachgebiet Simulation, Systemoptimierung und Robotik der Technischen Universität Darmstadt, 2018

Tabor, P.: dueling_dqn_keras.py (2020)
Tabor, Phil: Youtube-Code-Repository/dueling_dqn_keras.py at master. github.com, URL: https://github.com/philtabor/Youtube-Code-Repository/blob/master/ReinforcementLearning/DeepQLearning/dueling_dqn_keras.py, 2020, visited on 05/01/2021

Tabor, P.: main_keras_dueling_dqn_lunar_lander.py (2020)
Tabor, Phil: Youtube-Code-Repository/main_keras_dueling_dqn_lunar_lander.py at master. github.com, URL: https://github.com/philtabor/Youtube-Code-Repository/blob/master/ReinforcementLearning/DeepQLearning/dueling_dqn_keras.py, 2020, visited on 05/01/2021

Torres, J.: Deep Q-Network (DQN)-II (2020)
Torres, Jordi: Deep Q-Network (DQN)-II. towards data science, URL: https://towardsdatascience.com/deep-q-network-dqn-ii-b6bf911b6b2c, 2020, visited on 05/01/2021

Ulbrich, S. et al.: Defining Terms for Automated Driving (2015)
Ulbrich, Simon; Menzel, Till; Reschka, Andreas; Schuldt, Fabian; Maurer, Markus: Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving, in: 2015 IEEE 18th International Conference on Intelligent Transportation Systems, pp. 982–988, 2015

Verlag Deutsche Polizeiliteratur GmbH: Erläuterungen zur Straßenverkehrsordnung (2019)
Verlag Deutsche Polizeiliteratur GmbH: Erläuterungen zur Straßenverkehrsordnung, in: Verlag Deutsche Polizeiliteratur GmbH (Eds.): Polizei-Fach-Handbuch, Verlag Deutsche Polizeiliteratur GmbH, 2019

Wang, X. et al.: Falsification-Based Robust Adversarial Reinforcement Learning (2020)
Wang, Xiao; Nair, Saasha; Althoff, Matthias: Falsification-Based Robust Adversarial Reinforcement Learning, in: 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA), 2020

Wang, Z. et al.: Dueling Deep Reinforcement Learning (2016)
Wang, Ziyu; Schaul, Tom; Hessel, Matteo; Hasselt, Hado van; Lanctot, Marc; Freitas, Nando de: Dueling Network Architectures for Deep Reinforcement Learning, in: arXiv:1511.06581 [cs], 2016

Warsaw, B. et al.: PEP 435 (Enum) (2013)
Warsaw, Barry; Bendersky, Eli; Furman, Ethan: Adding an Enum type to the Python standard library, URL: https://www.python.org/dev/peps/pep-0435/, 2013, visited on 04/28/2021

Watkins, C. J. C. H. et al.: Q-learning (1992)
Watkins, Christopher J. C. H.; Dayan, Peter: Q-learning, in: Machine Learning, Vol. 8, pp. 279–292, 1992

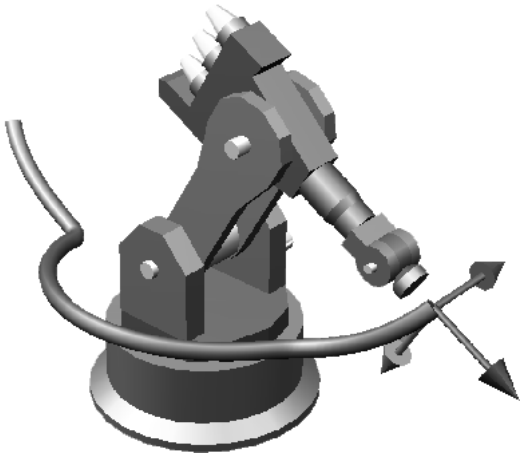
Winner, H. et al.: Handbuch Fahrerassistenzsysteme (2015)
Winner, Hermann; Hakuli, Stephan; Lotz, Felix; Singer, Christina: Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort, ATZ/MTZ-Fachbuch, 3. Edition, Springer Fachmedien, 2015

Annex

The annex comprises three pages (title page, page 24 and 56) of the used but not publicly available source: Stryk, Oskar von: Grundlagen der Robotik, Fachgebiet Simulation, Systemoptimierung und Robotik der Technischen Universität Darmstadt, 2018.

Skriptum

Grundlagen der Robotik



Univ.-Prof. Dr. Oskar von Stryk
Fachgebiet Simulation, Systemoptimierung und Robotik
Technische Universität Darmstadt

Ausgabe Wintersemester 2018/19

- Erfolgen die Rotationen um drei verschiedene Achsen des *festen* Bezugssystems, so werden diese Repräsentationen auch häufig **Kardanwinkel** genannt (X-Y-Z, X-Z-Y, Y-X-Z, Y-Z-X, Z-X-Y, Z-Y-X).

- Die Lösung des inversen Problems ${}^{a}R_b \longrightarrow \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}$ oder ${}^{a}R_b \longrightarrow \begin{pmatrix} \psi \\ \theta \\ \phi \end{pmatrix}$ ist nicht immer eindeutig.

- Es gibt *keinen* Satz von drei minimalen Drehwinkeln, der alle möglichen Orientierungen im Winkelraum *eindeutig* repräsentieren kann.

- Außer mit Rotationsmatrizen oder drei Winkelparametern kann eine Orientierungsrepräsentation auch mit *Quaternionen* erfolgen (siehe Anhang A.1).

- Eine allgemeine Rotation kann auch bezüglich eines einzelnen Drehwinkels angegeben werden, der bezüglich einer allgemeinen Drehachse gemessen wird, die nicht mit einer der Achsen des festen oder momentanen Bezugssystems übereinstimmt (siehe Anhang A.2).

Literaturhinweis: Bei [Craig, 1989], Appendix B, findet man jeweils 12 (und insgesamt 24) Darstellungen allgemeiner Rotationen durch Variationen der Konventionen für Euler- und Kardanwinkel.

2.5 Homogene Transformationen

Die klassische Transformationsbeziehung lautete (siehe Abschnitt 2.3):

$$ {}^{a}\mathbf{p} = {}^{a}\mathbf{r}_b + {}^{a}R_b \cdot {}^{b}\mathbf{p} $$

Eine einfachere Darstellung erhält man mit ***homogenen*** **Transformationsmatrizen** ${}^{a}T_b$:

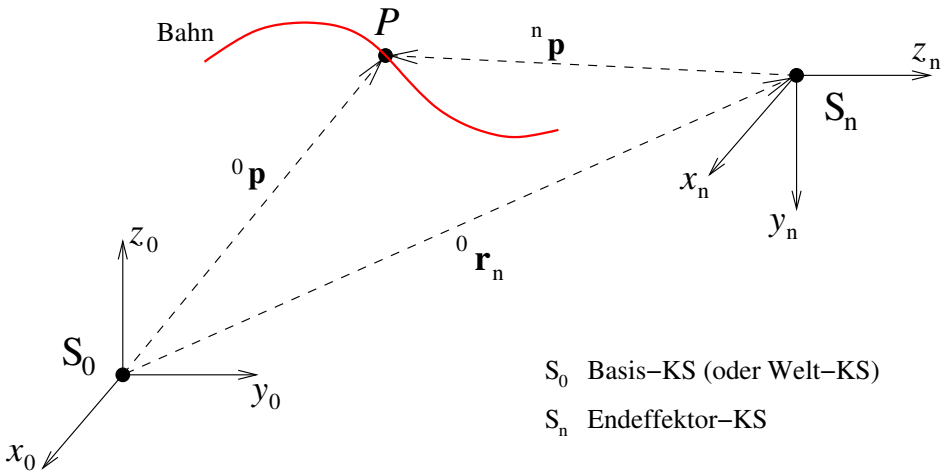
$$ \underbrace{{}^{a}\hat{\mathbf{p}}}_{\in\mathbb{R}^4} = \begin{pmatrix} {}^{a}\mathbf{p} \\ \hline 1 \end{pmatrix} = \underbrace{\left(\begin{array}{c|c} {}^{a}R_b & {}^{a}\mathbf{r}_b \\ \hline \mathbf{0}^{\mathrm{T}} & 1 \end{array}\right)}_{=:\,{}^{a}T_b} \cdot \begin{pmatrix} {}^{b}\mathbf{p} \\ \hline 1 \end{pmatrix} = \underbrace{{}^{a}T_b}_{\in\mathbb{R}^{4\times4}} \cdot \underbrace{{}^{b}\hat{\mathbf{p}}}_{\in\mathbb{R}^4} $$

Der **Vorteil** liegt in der einheitlichen Behandlung von Rotationen **und** Translationen eines Objektes durch eine einzige Matrix/Vektor-Multiplikation anstelle einer Matrix/Vektor-Multiplikation und einer Vektor-Addition.

Die **Inverse der homogenen Transformationsmatrix** ist gegeben durch:

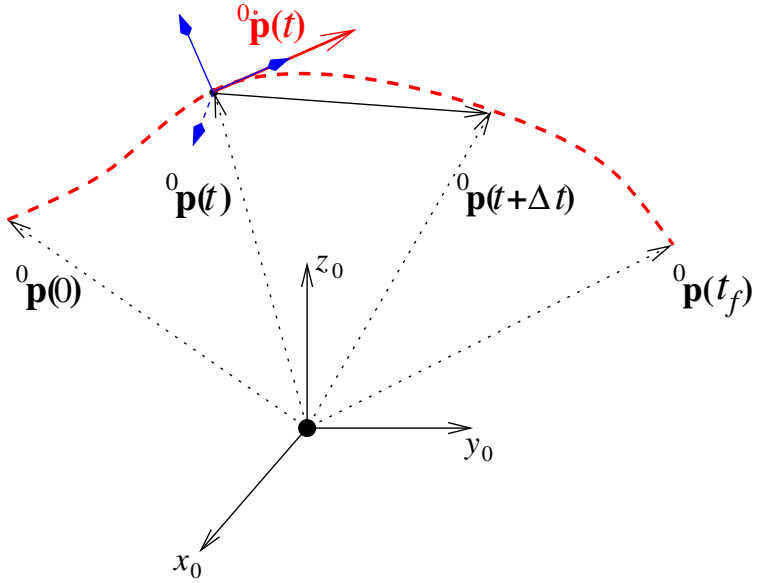
$$ ({}^{a}T_b)^{-1} = \left(\begin{array}{c|c} ({}^{a}R_b)^{\mathrm{T}} & -({}^{a}R_b)^{\mathrm{T}} \cdot {}^{a}\mathbf{r}_b \\ \hline \mathbf{0}^{\mathrm{T}} & 1 \end{array}\right) $$

Jetzt: auch Berücksichtigung des zeitlichen Verlaufs einer Bewegung:



Definition 4.1 *Eine* **Bahn** *(oder Pfad oder Weg, engl. path) ist eine kontinuierliche, räumliche Punktfolge, die z.B. vom TCP bei einer Bewegung durchlaufen wird.*

Definition 4.2 *Eine* **Trajektorie** *(engl. trajectory) ist eine Bahn mit* zeitlicher *Abhängigkeit, d.h. Position, Geschwindigkeit und Beschleunigung sind an jedem Punkt der Bahn im Raum gegeben.*



$$\lim_{\substack{\Delta t\to 0\\ \Delta t>0}} \frac{1}{\Delta t}\left({}^0\mathbf{p}(t+\Delta t)-{}^0\mathbf{p}(t)\right)=\frac{\mathrm{d}}{\mathrm{d}t}{}^0\mathbf{p}(t)={}^0\dot{\mathbf{p}}(t)=\begin{pmatrix}{}^0\dot{p}_x(t)\\ {}^0\dot{p}_y(t)\\ {}^0\dot{p}_z(t)\end{pmatrix}$$

Die Größe ${}^0\dot{\mathbf{p}} \in \mathbb{R}^3$ ist der ***Geschwindigkeitsvektor*** eines Punktes P mit Koordinatenvektor $\left({}^0\mathbf{p}(t)\right)$ zum Zeitpunkt t, d.h. der Tangentenvektor der Trajektorie $\left\{{}^0\mathbf{p}(\tilde{t}) \in \mathbb{R}^3 : 0 \le \tilde{t} \le t_f\right\}$ in $P(t)$ (t: time, t_f: final time). Dabei hängt ${}^0\dot{\mathbf{p}}$ vom Referenzkoordinatensystem (hier S_0) ab. Die Länge $\|{}^0\dot{\mathbf{p}}\|$ des Geschwindigkeitsvektors ist die absolute Geschwindigkeit und ändert sich beim Wechsel des Koordinatensystems (z.B. von S_0 zu S_n) nicht (solange S_n bzgl. S_0 fest ist).

$\Rightarrow$ Der Geschwindigkeitsvektor ${}^0\dot{\mathbf{p}}$ eines Punktes auf einer Trajektorie ist immer **relativ** zum jeweiligen Bezugskoordinatensystem (hier S_0)!

