Verhaltensentscheidung für automatisierte Fahrzeuge mittels Arbitrationsgraphen

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

von der KIT-Fakultät für Maschinenbau
des Karlsruher Instituts für Technologie (KIT)

angenommene

Dissertation

von

M.Sc. Piotr Franciszek Orzechowski

Tag der mündlichen Prüfung:	17.10.2022
Hauptreferent:	Prof. Dr.-Ing. Christoph Stiller
Korreferent:	Prof. Dr.-Ing. Tamim Asfour

Abstract

Automated driving promises to improve safety, comfort, and efficiency of road traffic and has the potential to catalyze a fundamental transformation of mobility. In addition to perceiving and interpreting its environment, an automated vehicle must reliably make safe and temporally consistent decisions — for example, whether, when and how to change lanes.

Since specialized behavior planning solutions already exist for many scenarios, the decision-making should be able to combine them appropriately. At the same time, it must be robust against faulty outputs or even failures of individual behavior options. Finally, the decision-making process should be transparent and traceable to enable an effective development process.

Therefore, this work first proposes an application-independent system architecture for secure and robust behavioral decision-making. It assembles basic behavior blocks in a hierarchical arbitration graph and ensures safety through verification and diverse levels of fallback. The respective behavior blocks are responsible for situation interpretation and behavior planning, while generic arbitrators carry out the decision process.

This architecture is then applied to the context of automated driving and evaluated in simulation. Thereby, the behavior options plan individual driving maneuvers and output them as trajectories. Multiple verifiers check these for feasibility, drivability and road safety. If a driving maneuver turns out to be unsafe, for example, the arbitration uses its alternative options and three fallback levels to continue generating safe behavior.

The evaluation shows that the presented method produces safe and stable driving behavior even at high failure rates. Meanwhile, the decoupling of situation interpretation and decision-making contributes to a transparent and

comprehensible decision-making process. This rigorous modularity allows to combine a wide range of behavior planning methods in an efficient and scalable manner. In addition, the bottom-up design leads to fast prototyping and iterative enhancement of the overall system.

Kurzfassung

Automatisiertes Fahren verspricht die Sicherheit, den Komfort und die Effizienz des Straßenverkehrs zu verbessern und hat das Potenzial eine grundlegende Transformation der Mobilität anzustoßen. Neben der Wahrnehmung und Interpretation seines Umfelds muss ein automatisiertes Fahrzeug zuverlässig sichere und zeitlich konsistente Entscheidungen treffen — bspw. ob, wann und wie ein Fahrstreifenwechsel durchgeführt wird.

Da es für viele Szenarien bereits spezialisierte Lösungsansätze zur Verhaltensplanung gibt, sollte die Verhaltensentscheidung in der Lage sein, diese sinnvoll miteinander zu kombinieren. Gleichzeitig muss sie robust gegen fehlerhafte Ausgaben oder gar Ausfälle einzelner Verhaltensoptionen sein. Schließlich sollte der Entscheidungsprozess transparent und nachvollziehbar sein, um einen effektiven Entwicklungsprozess zu ermöglichen.

Daher wird in dieser Arbeit zunächst eine anwendungsunabhängige Systemarchitektur zur sicheren und robusten Verhaltensentscheidung vorgeschlagen. Diese setzt grundlegende Verhaltensoptionen in einem hierarchischen Arbitrationsgraphen zusammen und sichert dabei die Stellgrößen mittels Verifikation und diversen Rückfallebenen ab. Die jeweiligen Verhaltensbausteine übernehmen dabei die Situationsinterpretation und Verhaltensplanung, während generische Arbitratoren den Entscheidungsprozess realisieren.

Anschließend wird diese Architektur auf den Kontext des automatisierten Fahrens angewandt und in Simulation evaluiert. Dabei planen die Verhaltensoptionen einzelne Fahrmanöver und geben diese als Trajektorien aus. Drei Verifikatoren prüfen diese auf Gültigkeit, Realisierbarkeit und Verkehrssicherheit. Stellt sich ein Fahrmanöver bspw. als unsicher heraus, greift die Arbitration auf Alternativoptionen und drei Rückfallebenen zurück, um weiterhin ein sicheres Verhalten zu erzeugen.

Die Evaluation zeigt, dass die vorgestellte Methode auch bei hohen Ausfallraten ein sicheres und stabiles Fahrverhalten erzeugt. Die Entkopplung von Situationsinterpretation und Verhaltensentscheidung trägt außerdem zu einer transparenten und nachvollziehbaren Entscheidungsfindung bei. Dank konsequenter Modularität können vielfältige Methoden der Verhaltensplanung effizient und skalierbar miteinander kombiniert werden. Zudem ermöglicht der Bottom-Up Entwurf schnelles Prototyping und eine iterative Weiterentwicklung des Gesamtsystems.

Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Mess- und Regelungstechnik (MRT) des Karlsruher Instituts für Technologie (KIT) sowie am FZI Forschungszentrum Informatik. Sie wäre nicht möglich gewesen ohne die vielfältige Unterstützung meiner KollegInnen, Familie und Freunde.

Zunächst möchte ich mich bei Herrn Prof. Dr.-Ing. Christoph Stiller bedanken, für die Möglichkeit der Promotion und die tollen Rahmenbedingungen, die er am MRT geschaffen hat. Besonders habe ich zu schätzen gelernt, dass Ergebnisse direkt auf den Versuchsträgern integriert und anschließend auf Demos sowie im öffentlichen Straßenverkehr in Aktion gebracht werden konnten. Ebenso danke ich Prof. Dr.-Ing. Tamim Asfour für die Übernahme des Korreferats und das inhaltliche Feedback aus Sicht der humanoiden Robotik. Für die inhaltliche Inspiration zu den Arbitratoren und hervorragende wissenschaftliche Betreuung möchte ich Herrn Dr. rer. nat. Martin Lauer danken.

Meine Promotion startete ich insbesondere aus der Motivation heraus, ein wissenschaftliches Thema anwendungsnah zu erforschen und im Team umzusetzen. Am MRT habe ich hierfür ein tolles Doktorandenkollegium gefunden, geprägt von Teamgeist, Ansporn und Spaß an der Sache. Ich verzichte darauf, alle KollegInnen aufzuzählen, hat doch jeder seinen Teil beigetragen. Daher bedanke ich mich bei allen für die anregenden sowie unterhaltsamen Fach- und Triviadiskussionen in unseren Sommerseminaren, Gruppenmeetings, Konferenzbesuchen, Social-Tuesdays und endlosen Kaffeepausen. Ganz konkret einen Dank für das Korrekturlesen an Frank, Hendrik, Jan, Lingguang, Martin, Nick und Sahin. Schließlich ein besonderer Dank an Christoph für die enge inhaltliche Zusammenarbeit an den Arbitrationsgraphen. Christoph hat einen wesentlichen Beitrag zum Gelingen dieser Arbeit beigetragen.

Auch dem Sekretariat möchte ich danken, das durch seinen Einsatz einen reibungslosen Forschungs- und Lehrbetrieb ermöglicht. Ebenso den Kollegen aus den Werkstätten mein Dank, sie erwecken große sowie kleine Versuchsträger zum Leben und halten sie für die Doktoranden und Studis am Laufen.

Einen Gruß möchte ich auch an die Hochschulgruppe KITcar aussprechen, die mich in meinem Masterstudium sehr geprägt und letztlich zu der Promotion im Automatisierten Fahren gebracht hat.

Nicht zuletzt möchte ich meinen Eltern von Herzen danken. Sie haben mich von klein auf geliebt, gefördert und ermutigt – und somit im weiteren Sinne die Grundlage für diese Arbeit geschaffen. Mein größter Dank gilt meiner Frau Anjuli, die mich tagtäglich bereichert, inspiriert, stärkt und unterstützt. Ohne ihren Rückhalt hätte ich diese Arbeit vermutlich nicht zu Ende gebracht.

Inhaltsverzeichnis

Abstract . . . . . **i**

Kurzfassung . . . . . **iii**

Danksagung . . . . . **v**

1 Einleitung **1**
1.1 Motivation und Umfeld der Arbeit 1
1.2 Stand der Technik 4
1.2.1 Verhaltensplanung 4
1.2.2 Verhaltensentscheidung 5
1.2.3 Verhaltensverifikation 8
1.3 Beiträge der Arbeit 9
1.4 Überblick 10

2 Grundlagen **11**
2.1 Architekturen für automatisierte Fahrzeuge 11
2.2 Verhaltensentscheidung 13
2.2.1 Endliche Zustandsautomaten 14
2.2.2 Entscheidungsbäume 17
2.2.3 Verhaltensbäume 19
2.2.4 Arbitrationsgraphen 21
2.3 Trajektorienplanung 27
2.3.1 Pfadplanung 29
2.3.2 Intelligent Driver Model 30
2.3.3 Nichtlineare Optimierung 30
2.4 Fehlertolerante Systeme 31

2.5 Verkehrssicherheit . . . 34
2.5.1 Sicherheitsanalyse . . . 35
2.5.2 Verhaltensverifikation . . . 41

3 Verhaltensarbitration für automatisierte Fahrzeuge . . . 49
3.1 Umweltmodell . . . 49
3.2 Verhaltensbausteine . . . 52
3.2.1 Abstraktionsgrad . . . 52
3.2.2 Manöverrepräsentation . . . 53
3.2.3 Start- und Fortsetzungs-Bedingungen im Kontext des automatisierten Fahrens . . . 55
3.2.4 Verhaltensgenerierung . . . 56
3.3 Arbitratoren . . . 60
3.3.1 Kosten-Arbitrator . . . 61
3.3.2 Übersicht . . . 62

4 Sichere und fehlertolerante Verhaltensarbitration . . . 63
4.1 Sicherheitsziele . . . 64
4.2 Sicherheitskonzept . . . 66
4.2.1 Fehlervermeidung beim Systementwurf . . . 67
4.2.2 Fehlervermeidung dank Echtzeitfähigkeit . . . 69
4.2.3 Ausfallsicherheit durch Fehlerkompensation . . . 73
4.2.4 Verkehrssicherheit durch Verifikation . . . 79
4.3 Systemübersicht . . . 82

5 Verhaltensbausteine in der Anwendung . . . 85
5.1 Grundlegende Fahrmanöver . . . 85
5.1.1 Ausparken . . . 86
5.1.2 Einparken . . . 89
5.1.3 Folgefahrt . . . 91
5.1.4 Kreuzung passieren . . . 95
5.1.5 Gefahrenstelle langsam queren . . . 99
5.1.6 Fahrstreifenwechsel . . . 102
5.2 Fahrmanöver der Rückfallebenen . . . 105
5.2.1 Letztes Manöver fortsetzen . . . 105

5.2.2 Fail-Safe-Trajektorie . . . 106
5.2.3 Nothalt . . . 107

6 Evaluation und Validierung . . . 109
6.1 Validierung im Realversuch . . . 109
6.1.1 Teststrecke . . . 111
6.1.2 Arbitrationsgraph . . . 111
6.2 Simulative Evaluation . . . 112
6.2.1 Testumgebung . . . 113
6.2.2 Implementierung . . . 114
6.2.3 Valide Verhaltensentscheidung . . . 116
6.2.4 Sichere Verhaltensentscheidung . . . 127

7 Zusammenfassung und Ausblick . . . 135
7.1 Zusammenfassung . . . 135
7.2 Diskussion . . . 137
7.3 Ausblick . . . 140

Literatur . . . 143

Abkürzungsverzeichnis . . . 163

Anhang

A Verhaltenskompetenzen . . . 167

1 Einleitung

Dieses Kapitel stellt zunächst die Motivation für die vorliegende Dissertation, sowie das zugrunde liegende Arbeitsumfeld vor. Anschließend wird in Abschnitt 1.2 der Stand der Technik zu verwandten Forschungsthemen umrissen, bevor Abschnitt 1.3 die Beiträge dieser Arbeit zusammenfasst und Abschnitt 1.4 einen Überblick über die weiteren Kapitel gibt.

1.1 Motivation und Umfeld der Arbeit

Fahrerassistenzsysteme und automatisiertes Fahren haben in den letzten vier Jahrzehnten enorme Fortschritte erreicht [Ben14, Bad21]. Trotz gestiegener Neuzulassungen hat die Fahrerassistenz auf Stabilisierungsebene, wie ABS und ESP, die Zahl tödlicher Unfälle drastisch reduzieren können. Dies leitete u. a. die Entwicklung von Unfallwarn- und Komfortsystemen ein, die sich mittlerweile fest auf dem Automobilmarkt etabliert haben. Das automatisierte Fahren stellt den nächsten Schritt dar, bei dem das Fahrzeug je nach Automatisierungsgrad die dynamische Fahraufgabe teilweise oder sogar ganz vom Menschen übernimmt [SAE21].

Durch diese weitere Automatisierung soll die Verkehrssicherheit sowie der Fahrkomfort noch weiter erhöht sowie der Verkehrsfluss optimiert und somit Emissionen reduziert werden. Zusätzlich werden vollautomatisierte Fahrzeuge neue Formen der Mobilität ermöglichen, die sogar gesellschaftliche Themen wie die soziale Teilhabe mobilitätseingeschränkter Menschen berühren. Erhöht sich durch die Automatisierung zudem die Attraktivität von Carsharing und anderen alternativen Mobilitätsformen, besteht die Chance heutige Parkflächen wieder zu lebenswerten öffentlichen Räumen umzugestalten.

Schließlich werden die hohen Investitionen zweifellos auch durch wirtschaftliche Überlegungen, wie bspw. Kosteneinsparungen in der Logistik und der Personenbeförderung, motiviert. Einerseits sind in diesen Branchen folglich Arbeitsplatzverluste zu erwarten [Moh22], andererseits verschlechtern sich die Arbeitsbedingungen von Berufskraftfahrern in der Logistik immer weiter [Dri14, Göt22]. Verkehrsverbünde im Öffentlichen Personennahverkehr haben zudem seit Jahren Schwierigkeiten ausreichend Berufskraftfahrer zu finden [Völ18].

Teil- oder vollautomatisiertes Fahren hat also das Potenzial, die Art und Weise wie wir uns fortbewegen grundlegend hin zu einer sicheren, effizienten, nachhaltigen und inklusiven Mobilität zu verändern. Begleitet wird diese Transformation von einem Wandel in der Arbeitswelt des Mobilitätssektors und weiteren, bisweilen kaum voraussehbaren, Sekundäreffekten.

Angetrieben durch die positiven Gestaltungsmöglichkeiten sowie technologische Sprünge im Bereich des Maschinellen Lernens, haben die Forschungsanstrengungen im Bereich der Fahrerassistenzsysteme und vollautomatisierter Fahrzeuge zuletzt sowohl im akademischen als auch im kommerziellen Bereich drastisch zugenommen. Dabei muss ein automatisiertes Fahrzeug seine Umgebung zum einen zuverlässig wahrnehmen, zum anderen auf dieser Grundlage sichere Fahrmanöver generieren. Die Verhaltensentscheidung bestimmt hierfür Zeitpunkt, Art und Ausprägung des geplanten Fahrmanövers – also ob, wie, wo und wann bspw. ein Fahrstreifenwechsel stattfinden oder wann und in welche Parklücke eingeparkt werden soll. Gleichzeitig stellt sie sicher, dass das Verhalten zeitlich konsistent, komfortabel, zielführend und sicher ist.

Während die Wahrnehmung auf großen Datensätzen evaluiert werden kann, wird die Regelung in Simulation, mittels Hardware-in-the-Loop oder auf geschlossenem Testgelände auf dem Fahrzeug erprobt. Zwar können auch Teile der Verhaltensgenerierung in Simulation validiert werden, allerdings hängt ihre Leistungsfähigkeit stark von der Interaktion mit anderen Verkehrsteilnehmern im realen Straßenverkehr ab, da menschliches Verhalten in Simulation nur unzureichend modelliert werden kann. Daher ist eine sichere und

robuste Verhaltensentscheidung von wesentlicher Bedeutung, um die Erprobung und den späteren Einsatz von automatisierten Fahrzeugen im Straßenverkehr zu ermöglichen.

Gleichzeitig besteht der Bedarf für eine skalierende und generalisierende Softwarearchitektur, die es ermöglicht, die vielfältigen erprobten Verfahren zur Verhaltensplanung in bestimmten Szenarien [Hoe17, Mir18, Hub19, Deb21] – einschließlich klassischer und probabilistischer Methoden sowie Ansätzen des Maschinellen Lernens – in einer übergeordneten Verhaltensentscheidung miteinander zu kombinieren. Unternehmen wie Waymo und Uber setzen die Generierung von taktischem und strategischem Verhalten jedoch ausschließlich mittels Maschinellem Lernen, in Teilen sogar in sog. Ende-zu-Ende Architekturen, um [Ban19, Cas21]. Deutsche Fahrzeughersteller und -zulieferer setzen hingegen auf klassische Ansätze, wie Endliche Zustandsautomaten oder Entscheidungsbäume [Aeb15], die eine solche Kombination von Methoden ermöglicht. Aber auch kleinere Forschungseinrichtungen, bspw. aus dem universitären Kontext, setzen auf regelbasierte Verfahren zur Entscheidungsfindung, weil diese einfach umzusetzen und schnell einsatzbereit sind. Bei steigender Anzahl an Verhaltensoptionen leiden solche Ansätze in der Regel jedoch unter schlechter Erklärbarkeit, Wartbarkeit und Skalierbarkeit. Daher hat die Robotik viele Architekturen hervorgebracht, um diese Probleme zu adressieren. Darunter sind die verhaltensbasierten Systeme und ihre Derivate besonders interessant.

Vor diesem Hintergrund wird in der vorliegenden Arbeit ein vielversprechendes, verhaltensbasiertes Verfahren zur Entscheidungsfindung aus der Robotik aufgegriffen und auf den Anwendungsbereich des automatisierten Fahrens übertragen. Anschließend wird die Methode mit Maßnahmen aus der Forschung zu zuverlässigen und fehlertoleranten Systemen sowie der Verkehrssicherheit erweitert, um schließlich eine sichere und robuste Verhaltensentscheidung für automatisierte Fahrzeuge zu realisieren. Die vorgeschlagene Methode nutzt modulare Verhaltensbausteine in einem hierarchischen Arbitrationsgraphen, ist skalierbar in der Anzahl der Verhaltensoptionen und ermöglicht die Kombination vielfältiger szenariospezifischer Verfahren. Außerdem gewährleistet sie eine robuste Ausführung und sicheres Verhalten

durch Verifikation und diverse Rückfallebenen. Der modulare und hierarchische Aufbau ermöglicht zudem einen iterativen Entwurfsprozess, erhöht die Wartbarkeit und führt zu einer transparenten sowie nachvollziehbaren Entscheidungsfindung.

1.2 Stand der Technik

Die Literatur ist reich an diversen Verfahren zur Verhaltens- bzw. Trajektorienplanung in spezifischen Situationen sowie zur Verhaltensentscheidung, die einer Strategie folgend zwischen diesen Manöveroptionen wählt. Dabei treffen erstere teilweise auch implizite kombinatorische Manöverentscheidungen, bspw. wann und in welche Lücke sich das Fahrzeug in einem Kreisverkehr einordnet [Deb21]. Mittels Verhaltensverifikation soll zudem gewährleistet werden, dass nur sichere Fahrmanöver ausgeführt werden. Im Folgenden wird der Stand der Technik in diesen drei Forschungsfeldern beleuchtet.

1.2.1 Verhaltensplanung

Zu den klassischen Methoden der Trajektorienoptimierung zählen u. a. Graphensuchverfahren aus der Dynamischen Programmierung, die Modellprädiktive Regelung zur Lösung des Optimalsteuerungsproblems und direkte oder indirekte Methoden aus der statischen Optimierung [Wer17]. 2013 wurde die sog. *Bertha Benz Memorial Route* nahezu vollständig automatisiert befahren. Die Trajektorienplanung wurde hierbei als quadratisches Optimierungsproblem formuliert und mittels Sequentieller Quadratischer Programmierung gelöst [Zie14a]. Gutjahr u. a. [Gut17] schlagen hingegen eine lineare Modellprädiktive Regelung vor, um ein linear-quadratisches Optimalsteuerungsproblem zu lösen. Durch eine geschickte Modellierung konvergiert das Verfahren garantiert und außergewöhnlich schnell, sofern eine Lösung existiert. Schließlich bestimmen Banzhaf u. a. [Ban18] Trajektorien zum Manövrieren in engen Umgebungen mittels Rapidly-exploring

Random Trees (RRT*) und kinematischen Bewegungsmodellen. Bemerkenswert ist, dass hierbei auch zu erwartende Unsicherheiten in der Regelung und Lokalisierung berücksichtigt werden.

Neben Ansätzen aus der Regelungstechnik und Optimierungstheorie gewinnen probabilistische Methoden sowie Verfahren des Maschinellen Lernens immer mehr Aufmerksamkeit [Kir21]. Partiell beobachtbare Markow-Entscheidungsprobleme (POMDPs) werden eingesetzt, um Unsicherheiten explizit in das Entscheidungsproblem zu integrieren. Somit werden Aktionen gefördert, die dazu beitragen Unsicherheiten bspw. in der Wahrnehmung und Prädiktion zu reduzieren. Hubmann u. a. [Hub18a] planen Trajektorien für komplexe Kreuzungssituationen entlang eines vorab bestimmten Pfades, wobei die nur indirekt beobachtbare Routenintention anderer Verkehrsteilnehmer mittels POMDP optimal in die Planung einbezogen wird.

Ansätze des Maschinellen Lernens, auch die des Bestärkenden Lernens, bauen häufig auf erfolgreiche Künstliche Neuronale Netze aus der Bildverarbeitung auf. Beispielsweise projizieren Chen u. a. [Che19] die Eingangsdaten in Vogelperspektive, reduzieren dieses generierte Bild mithilfe eines Autoencoders in eine niedrig-dimensionale Repräsentation (sog. Latente Zustandscodierung) und speisen diese bspw. in eine Actor-Critic-Architektur ein. Hiermit gelingt es der Methode die Quer- und Längsplanung für einen komplexen, stark befahrenen Kreisverkehr zu realisieren.

1.2.2 Verhaltensentscheidung

Im Bereich der Verhaltensentscheidung finden sich sowohl monothematische Methoden als auch generische Architekturen, die es ermöglichen diverse Verfahren der Verhaltensplanung zu kombinieren. Insbesondere Ende-zu-Ende Architekturen des Maschinellen Lernens, setzen auf einen einheitlichen Ansatz für alle möglichen Manöver- und Trajektorienvarianten. In [Cas21] wird die gesamte Verarbeitungskette – von der Wahrnehmung, über die Onlinekartierung bis hin zur Routenplanung – gelernt. Die Trajektorienkandidaten werden aus einer Datenbank real beobachteter Experten-Trajektorien entnommen und mit einer ebenfalls gelernten Kostenfunktion bewertet.

Solche Ansätze des Maschinellen Lernens haben, neben einer einheitlichen Formulierung, den Vorteil, dass sie Unsicherheiten in der Situationsinterpretation implizit mit berücksichtigen. Allerdings sind hierzu meist enorme Datenmengen und eine leistungsfähige Rechnerinfrastruktur notwendig. Zudem bieten sie beim Entwurf und der Weiterentwicklung nur sehr indirekte Möglichkeiten das Systemverhalten zu beeinflussen. Wird also in spezifischen Situationen ein Fehlverhalten beobachtet, ist es schwierig dies gezielt zu verbessern, zumal bei Systemänderungen im Zweifelsfall auch das Verhalten in anderen Situationen verändert wird.

Klassische Architekturen der Verhaltensentscheidung können solche Kopplungen reduzieren oder gar vollständig aufheben. Außerdem erlauben sie den Einsatz unterschiedlicher Verfahren zur Verhaltensplanung, sodass je nach angestrebter Funktionalität oder adressierter Situation die hierfür bestmögliche Methode eingesetzt werden kann. Zunächst wurden vielfach zustandsbasierte Architekturen, wie Endliche Zustandsautomaten verwendet, um je nach Situation einen geeigneten Fahrmodus zu wählen [Mon08, Zie14b]. In [Mon08] bspw. bilden die Zustände verschiedene Manöveroptionen – z. B. Freie Fahrt, Kreuzung passieren, Einparken und Wenden – ab. [Ard11] stellt ein hybrides Konzept vor, das Endliche Zustandsautomaten und Entscheidungsbäume für das hochautomatisierte Fahren auf Autobahnen kombiniert. Ein übergeordnetes Netzwerk aus hybriden deterministischen Zustandsautomaten bestimmt dabei den Systemzustand. Daraufhin bestimmen zwei Entscheidungsbäume den Fahrwunsch, prüfen die Durchführbarkeit von Manöveroptionen und legen schließlich jeweils den Modus für die Quer- und Längsführung fest.

Zustandsbasierte Architekturen sind i. d. R. einfach umzusetzen und können verschiedene Verfahren zur Verhaltensplanung miteinander kombinieren. Durch ihren Top-Down Ansatz muss allerdings im Entwurfsprozess, wie auch der Weiterentwicklung, stets das Gesamtsystem und die Wechselwirkungen einzelner Zustände miteinander berücksichtigt werden. Bei einer großen Zahl an Zuständen wird somit die Komplexität von bspw. Endlichen Zustandsautomaten nicht mehr beherrschbar. Folglich skalieren solche Systeme nur schlecht mit der Anzahl an Verhaltensoptionen.

Als Gegenentwurf zu zustandsbasierten Architekturen haben sich in der Robotik, ausgehend von Brooks' Subsumptionskonzept [Bro86], zahlreiche verhaltensbasierte Methoden entwickelt. Diese setzen das Gesamtverhalten im Bottom-Up Design aus einfachen Teilverhalten zusammen. Die Computerspielbranche brachte hieraus später die Verhaltensbäume hervor, die zunächst in [Ögr12] konzeptionell für unbemannte Luftfahrzeuge eingesetzt und anschließend auf weitere Anwendungsgebiete der Robotik übertragen wurden [Col18]. Dabei wird die Verhaltensentscheidung über eine Baumstruktur mittels sog. *Kontrollfluss-Knoten* realisiert. Die Blätter des Baumes übernehmen hingegen die Verhaltensplanung oder gar Regelung, wobei sie direkten Zugriff auf Sensorik und Aktorik haben können.

Unabhängig davon wurden im Roboterfußball Arbitrationsgraphen entworfen, die u. a. das Subsumptionskonzept mit Objektorientierter Programmierung kombinieren [Lau10]. Dabei adressieren modulare Verhaltensbausteine grundlegende Verhaltenskompetenzen und übernehmen außerdem die zugehörige Situationsinterpretation. Jeder Verhaltensbaustein gibt also selbstständig an, ob sein Verhalten in der vorliegenden Situation sinnvoll anwendbar ist. Auf dieser Grundlage entscheiden anwendungsunabhängige Arbitratoren, die in einer hierarchischen Graphstruktur angeordnet sind, welche Verhaltensoption voraussichtlich am geeignetsten ist. Einheitliche Eingabe- und Ausgabeschnittstellen der Verhaltensbausteine und Arbitratoren stellen ihre Wiederverwendbarkeit sicher und erhöhen zugleich die Verständlichkeit des Gesamtsystems. In [Orz20] wurden Arbitrationsgraphen erstmalig in den Kontext des automatisierten Fahrens übertragen und ihre Vorzüge in Simulation validiert. Die Verhaltensbausteine realisieren dabei einzelne Fahrmanöver, wie Folgefahrt, Fahrstreifenwechsel und Einparken. Für die Verhaltensentscheidung wurden Arbitratoren eingesetzt, die zwischen diesen Optionen nach festgelegter Priorität und zu erwartenden Routingkosten wählen.

Verhaltensbasierte Architekturen zeichnen sich u. a. durch eine hohe Reaktivität, konsequente Modularität und entsprechend gute Skalierbarkeit aus. Sie sind vergleichbar einfach umzusetzen wie zustandsbasierte Architekturen, und können ebenfalls verschiedene Verfahren der Verhaltensplanung in

einem Gesamtsystem kombinieren. Zusätzlich bieten sie durch ihren hierarchisch modularen Aufbau eine optimale Struktur, um einen Verifikationsmechanismus mit Rückfallebenen direkt in die Verhaltensentscheidung zu integrieren.

1.2.3 Verhaltensverifikation

Die Verifikation geplanter Fahrmanöver bzw. die Absicherung der Verhaltensgenerierung ist ein noch junges und offenes Forschungsfeld. Bisher werden diesbezüglich insbesondere folgende zwei prominente Methodenklassen verfolgt: Das Konzept der Verantwortungsbewussten Sicherheit [Sha17] ist darauf ausgelegt zu bewerten, ob ein Verhalten für die jeweilige Situation angemessen und verantwortungsvoll ist. Hierzu formalisiert es Verkehrsregeln, legt sinnvolle Annahmen fest und definiert Begriffe wie *sicherer Abstand*, *gefährliche Situation*, *angemessene Reaktion* und *Verantwortung*. Die Autoren zeigen zudem, dass es zu keinem Unfall käme, wenn sich alle Verkehrsteilnehmer an die vorgestellten Regeln hielten. Während die einzelnen formulierten Regeln sich in einfachen mathematischen Formeln niederschlagen, bildet das gesamte Konzept einen umfangreichen Katalog an situationsspezifischen Regeln und Anforderungen an die zeitlichen sowie räumlichen Abstände in Quer- und Längsrichtung, die Fahrzeuggeschwindigkeit und die Beschleunigung. Die überwältigende Anzahl an festzulegenden Parametern erschwert allerdings den Einsatz der Methode.

Ein einheitlicherer Ansatz basiert auf der Erreichbare-Mengen-Analyse und zielt darauf ab, eine beweisbar sichere sog. Fail-Safe-Trajektorie vorzuhalten, auf die gewechselt werden kann, sofern eine Kollision droht [Alt16]. Dabei wird die Fail-Safe-Trajektorie nur dann als sicher eingestuft, wenn sie keine Überlappung mit den Worst-Case-Belegungen anderer Verkehrsteilnehmer hat, also garantiert kollisionsfrei ist. Die Soll-Trajektorie muss hingegen nur sicherstellen, dass im nächsten Planungsintervall noch auf die Fail-Safe-Trajektorie gewechselt werden kann. Allerdings treffen bisherige Publikationen teilweise weitreichende Annahmen, die für die reale Anwendung sicherlich noch relaxiert werden müssen. Beispielsweise gehen sie davon aus, dass

Fahrzeuge ihren Fahrstreifen – abgesehen von Fahrstreifenwechseln – nicht verlassen, worauf man gerade in beengten Kreuzungsbereichen nicht vertrauen kann, insbesondere wenn keine Fahrbahnmarkierungen vorhanden sind. Dennoch stellt die Erreichbare-Mengen-Analyse einen hilfreichen Ansatz dar, u. a. weil er konsistent, anschaulich und zugleich effizient ist. Zudem lässt sich das Konzept der Fail-Safe-Trajektorien sehr gut in eine Verhaltensentscheidung mit Rückfallebenen integrieren.

1.3 Beiträge der Arbeit

Diese Arbeit schlägt eine Methode zur Verhaltensgenerierung für automatisierte Fahrzeuge vor, welche bereits innerhalb der Verhaltensentscheidung sicherstellt, dass geplante Fahrmanöver realisierbar und sicher sind. Somit kann sie bei riskanten Manövern oder bei Ausfällen einzelner Verhaltensoptionen rechtzeitig eingreifen und auf Alternativoptionen oder Rückfallebenen wechseln.

Hierzu wird ein Arbitrationsgraph entworfen, dessen Verhaltensbausteine grundlegende Verhaltenskompetenzen bzw. Fahrmanöver adressieren und in Form von Trajektorien ausgeben. Zudem wird das Arbitrationsverfahren um einen Verifikationsschritt und eine darauf ausgelegte Fehlerbehandlung erweitert. So wird garantiert, dass die Arbitratoren nur solche Trajektorien weiterreichen, die die Verifikation bestehen. Für die Rückfallebene werden Verhaltensbausteine implementiert, die das vorige Manöver fortsetzen, eine vorgehaltene Plan-B-Trajektorie zurückgeben oder eine Notbremsung durchführen.

Schließlich tragen diese Maßnahmen zu einer robusten und sicheren Verhaltensgenerierung bei, die auch auf andere Robotikanwendungen übertragbar ist. Die wesentlichen Beiträge dieser Arbeit sind zusammenfassend:

- Definition einer Szenario-unabhängigen *Manöverrepräsentation* für das automatisierte Fahren.

- Erweiterung des Arbitrationsverfahrens um eine *Verifikationslogik*, die sicherstellt, dass nur Verhaltensoptionen ausgeführt werden, welche einer Verifikation standhalten.
- Definition dreier *Verifikatoren* für das automatisierte Fahren, die potenzielle Manöver auf Gültigkeit, Realisierbarkeit und Verkehrssicherheit überprüfen.
- Entwurf eines *Sicherheitskonzepts*, um – mittels Verifikation, Echtzeitfähigkeit, Redundanz und Diversität – eine robuste und sichere Verhaltensarbitration zu gewährleisten.
- *Evaluation* des vorgeschlagenen Sicherheitskonzepts in einer anwendungsnahen Simulationsumgebung.

1.4 Überblick

Der restliche Teil der Arbeit ist wie folgt aufgebaut.

In Kapitel 2 wird zunächst eine Einführung in die für diese Arbeit wesentlichen Grundlagen gegeben. Im Anschluss beschreiben Kapitel 3 und 4 den Kern dieser Arbeit: Die Methode der Verhaltensarbitration wird zunächst auf das automatisierte Fahren übertragen und anschließend zu einem fehlertoleranten und sicheren System erweitert. Daraufhin werden in Kapitel 5 grundlegende Fahrmanöver zur Befahrung der Karlsruher Teststrecke entworfen sowie Fahrmanöver der Rückfallebenen definiert, um das Sicherheitskonzept aus Kapitel 4 zu komplettieren. Anschließend präsentiert Kapitel 6 den Realversuch und Ergebnisse der Evaluation. Zuletzt schließt Kapitel 7 die Arbeit mit einer Diskussion und Zusammenfassung ab.

2 Grundlagen

In diesem Kapitel sind die für diese Arbeit relevanten theoretischen Grundlagen gebündelt. Zunächst wird die Fahraufgabe in Abschnitt 2.1 modelliert und gängige Softwarearchitekturen für automatisierte Fahrzeuge vorgestellt. Anschließend fasst Abschnitt 2.2 verwandte Verfahren zur Verhaltensentscheidung zusammen, insbesondere die in dieser Arbeit erweiterten Arbitrationsgraphen. Abschnitt 2.3 geht in Kürze auf die Trajektorienplanung ein, bevor sich Abschnitt 2.4 zuverlässigen und fehlertoleranten Systemen widmet. Schließlich wird in Abschnitt 2.5 die Verkehrssicherheit und darin im Speziellen die Erreichbare-Mengen-Analyse beschrieben.

2.1 Architekturen für automatisierte Fahrzeuge

Die Fahraufgabe lässt sich laut [Mic85] wie in Abb. 2.1 als kaskadierter Regelkreis in eine strategische, taktische und operative Ebene gliedern. Die strategische Ebene definiert allgemeine langfristige Präferenzen und Ziele, u. a. der geplanten Route. Abhängig von der aktuellen Situation und den zuvor definierten Zielen werden in der taktischen Ebene Fahrmanöver umgesetzt, beispielsweise ein Fahrstreifenwechsel oder ein Haltemanöver. Auf der operativen Ebene wird das beabsichtigte Manöver letztlich mit einer Update-Frequenz in der Größenordnung von Millisekunden eingeregelt.

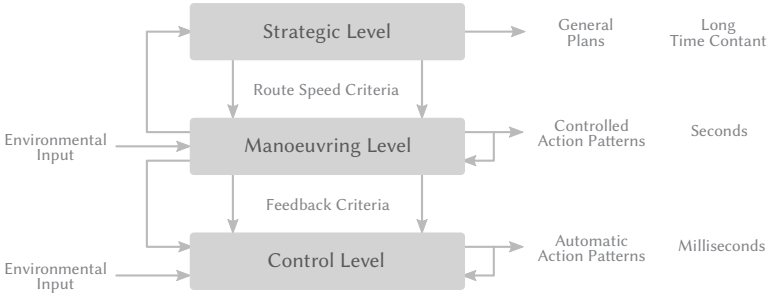


Abbildung 2.1: Das 3-Ebenen-Modell nach [Mic85] unterteilt die Fahraufgabe in eine strategische, taktische und operative Ebene.

Dieses ursprünglich menschliche Fahrer beschreibende Modell wird häufig auch auf die Automatisierung der Fahrfunktion angewandt. Die strategische Ebene übernimmt ein Navigationsmodul. Es bestimmt ausgehend von einem gegebenen Fahrtziel, meist voreingestellten Gütekriterien, dem Wegenetz und ggf. der aktuellen Verkehrslage die Route. Diese wird in Form von Zwischenzielen, Straßen oder gar Straßenabschnitten dargestellt. Ein Planungsmodul bestimmt Fahrmanöver auf taktischer Ebene mit einem Zeithorizont von etwa 5 bis 20 Sekunden. Es bildet die aktuelle Situation einschließlich wahrgenommener Verkehrsteilnehmer unter Berücksichtigung der vorgesehenen Route und vorhandenen Verkehrsregeln auf eine geeignete Soll-Trajektorie ab. Hierfür bestimmt es ggf. auch den zu befahrenden Fahrstreifen. Die operative Ebene wird schließlich von der Regelung übernommen. Sie setzt die Soll-Trajektorie in einer hochfrequenten Regelschleife in Aktor-Stellgrößen wie Lenkwinkel und Beschleunigung um.

Die somit gängigsten Architekturen für automatisierte Fahrzeuge legt Abb. 2.2 dar. Im einfachsten Fall werden die Wahrnehmung, Prädiktion, Verhaltensentscheidung, Trajektorienplanung und Regelung in einer entkoppelten *Verarbeitungskette* durchgeführt. Für die Planung von interaktivem Verhalten muss die Prädiktion hingegen in die Verhaltens- und Trajektorienplanung *integriert* werden. Wird die gesamte Verarbeitungskette in einen gemeinsamen Verarbeitungsschritt verwoben, wie es beispielsweise in Ansätzen des Maschinellen Lernens häufig zu beobachten ist, spricht man von *Ende-zu-Ende Planung*.

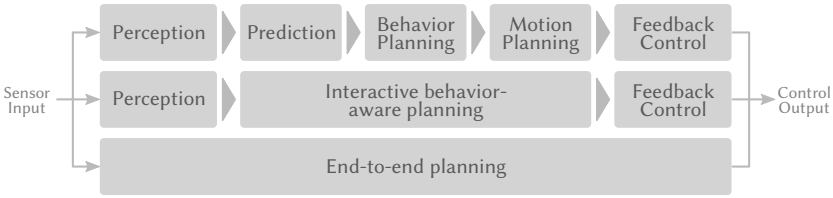


Abbildung 2.2: Verschiedene Architekturen für automatisierte Fahrzeuge, angelehnt an [Sch18].

2.2 Verhaltensentscheidung

Die Verhaltensentscheidung ist Teil der taktischen Ebene (vgl. Abb. 2.1) bzw. des Planungsmoduls. Ausgehend von der aktuellen Situation trifft sie die Entscheidung, welches Manöver gefahren wird. Dieses gibt sie entweder in Form von Nebenbedingungen an die nachgelagerte Trajektorienplanung oder, wenn sie die Trajektorienplanung selbst übernimmt, als Soll-Trajektorie an die Regelung weiter.

In der Literatur haben sich vielfältige Verfahren etabliert, deren Einsatzbereiche von reiner Verhaltensentscheidung hin zur situationsspezifischen Trajektorienplanung übergehen. Regelbasierte Verfahren, wie Endliche Zustandsautomaten, Entscheidungsbäume oder Arbitrationsgraphen, adressieren die reine Verhaltensentscheidung durch diskrete Zustands-/Moduswechsel. Graph- bzw. suchbasierte Verfahren wie A*, Probabilistische Straßenkarten (PRM*) und Rapidly-exploring Random Trees (RRT*), sind insbesondere in der Pfadplanung mobiler Roboter beliebt, können aber auch zur Entscheidungsfindung eingesetzt werden. In der Trajektorienplanung kommen immer häufiger probabilistische Methoden, wie POMDPs, sowie Verfahren des Maschinellen Lernens, u. a. das Bestärkende Lernen, zum Einsatz. Dabei übernehmen sie teilweise auch implizite Entscheidungen, bspw. wann und in welche Lücke ein Fahrstreifenwechsel stattfindet.

In diesem Abschnitt liegt der Fokus auf regelbasierten Verfahren, zu denen sowohl klassische zustandsbasierte Verfahren wie Endliche Zustandsautomaten (Abschnitt 2.2.1) und Entscheidungsbäume (Abschnitt 2.2.2), sowie Verhaltensbasierte Verfahren wie Verhaltensbäume (Abschnitt 2.2.3) und Arbitrationsgraphen (Abschnitt 2.2.4) zählen. Für eine umfangreiche Methodenübersicht sei auf [Sch18, Yur20, Voß21, Gam21] verwiesen. Eine umfangreichere Fassung dieses Abschnitts erscheint außerdem in [Orz23].

Regelbasierte Verfahren sind, wie bereits in Abschnitt 1.3 ausgeführt, insbesondere im Kontext der Fahrerassistenzsysteme sowie für den Einsatz von Versuchsfahrzeugen kleinerer Forschungsgruppen weit verbreitet [Aeb15, Bac08, Mon08, Zie14a]. Dies lässt sich u. a. auf ihre einfache Anwendung, zahlreiche frei verfügbare Software-Frameworks [Sch14, Bur18, Gre21b, The20b], einschließlich direkter Einbindung in MATLAB [The20a], oder auch deren Standardisierung in der Unified Modeling Language [Obj17] zurückführen.

2.2.1 Endliche Zustandsautomaten

Endliche Zustandsautomaten, auch als deterministische endliche Automaten bezeichnet, haben ihren Ursprung im Hardware Design und der Theoretische Informatik [Wag06, Hop07, Vos16]. Mittlerweile werden sie auch in der Robotik [Sic16] und im Bereich der Fahrerassistenzsysteme eingesetzt [Zie14b, Aeb15]. Endliche Zustandsautomaten sind leicht zu verstehen und zugleich sehr einfach zu implementieren.

Dieser Abschnitt fasst die wichtigsten Inhalte aus [Vos16] und [Hop07] zusammen, um eine praxisorientierte Einführung in die Theorie Endlicher Zustandsautomaten zu geben.

Ein Endlicher Zustandsautomat[1] besteht aus

- einer endlichen Menge an Zuständen S,
- einer endlichen Menge an Eingangssymbolen Σ (oder Ereignissen),
- einer endlichen Menge an Ausgangssymbolen Δ (oder Aktionen),
- einer Zustandsübergangsfunktion $\delta : S \times \Sigma \to S$,
- einer Ausgabefunktion $\lambda : S \to \Delta$,
- einem Anfangszustand $s_0 \in S$ und
- einer endlichen Menge an Endzuständen $F \subseteq S$.

Ein Endlicher Zustandsautomat beginnt zunächst in dem Anfangszustand s_0. Bei jedem, in der Regel von außen eingehenden, Ereignis $e_i \in \Sigma$ wechselt er in einen neuen Zustand $s_{i+1} = \delta(s_i, e_i)$. Dabei kann der neue Zustand auch als derselbe Zustand $s_{i+1} = s_i$ festgelegt sein. Nach jedem Zustandswechsel in einen Zustand s_i gibt der Zustandsautomat die Ausgabe $a_i = \lambda(s_i)$ aus. Einzelne Zustände können in Erweiterungen wie den Hierarchischen Zustandsautomaten wiederum selbst Zustandsautomaten sein oder auch nebenläufige Zustandsautomaten enthalten.

In der Robotik allgemein decken Zustände i. d. R. einzelne Verhaltensmodi ab, sodass ihre Ausgaben, die geplante Aktion, als Stell- oder Zielgrößen an die ausführende Schicht weitergegeben werden. Die Ereignisse des Zustandsautomaten werden meist von einer Situationsinterpretation erzeugt, um ggf. einen Wechsel des Verhaltensmodus einzuleiten.

Bspw. setzte das Team Junior der Universität Stanford Endliche Zustandsautomaten erfolgreich in der DARPA Urban Challenge ein [Mon08]. Wie in Abbildung 2.3 veranschaulicht, stellen hierbei die Zustände taktische Fahrmanöver wie Weiterfahrt, Passieren einer Kreuzung oder Einparken ab. Im Sinne einer besseren Übersichtlichkeit wurden allerdings zwei Zustände (*Escape* und *Traffic Jam*) ausgelassen, weil sie von fast allen übrigen Zuständen aus erreichbar sind.

[1] In dieser Arbeit wird die Moore-Notation verwendet. Alternativ lässt sich die Notation auch in das sog. Mealy-Modell überführen.

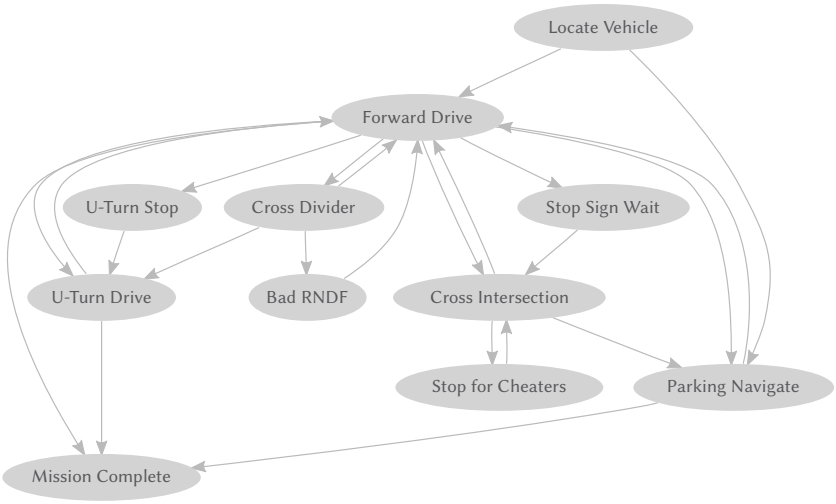


Abbildung 2.3: Zustandsautomat das Team Junior zur Teilnahme an der DARPA Urban Challenge, angelehnt an [Mon08].

Für überschaubare Verhaltensprobleme sind Endliche Zustandsautomaten aufgrund ihrer einfachen Umsetzung und intuitiven Darstellung eine gute Wahl. Da die Anzahl möglicher Zustandsübergänge allerdings im schlimmsten Fall quadratisch mit der Anzahl der Zustände wächst, skalieren sie bei komplexeren Systemen schlecht. Dieses Wachstum können Hierarchische Zustandsautomaten zumindest auf manuell definierte hierarchische Ebenen begrenzen.

Die schlechte Modifizierbarkeit ist ein weiterer Nachteil von Endlichen Zustandsautomaten: Beim Hinzufügen oder Entfernen von Zuständen müssen ggf. viele der bereits vorhandenen Zustände und Zustandsübergänge in Betracht gezogen und mit angepasst werden.

Die Zustandsübergänge in Endlichen Zustandsautomaten gleichen außerdem Sprunganweisungen, die es erschweren Quellcode zu verstehen, zu analysieren oder zu verifizieren. Um nachzuvollziehen, warum ein bestimmter Zustand aktuell aktiv ist, muss folglich eine Ereignishistorie erstellt und aufwendig Schritt für Schritt nachvollzogen werden. Daher werden Goto Befehle in

der Softwareentwicklung spätestens seit Einführung Strukturierter Programmiersprachen weitgehend vermieden [Dij68, Dah72].

Bei der Visualisierungen von Zustandsautomaten komplexerer Systeme leidet schließlich die Übersichtlichkeit an der großen Zahl an Zustandsübergängen. In Abb. 2.3 sah sich Team Junior daher sogar gezwungen, zwei stark vernetzte Zustände auszulassen.

2.2.2 Entscheidungsbäume

Entscheidungsbäume wurden ursprünglich als rekursive Strukturen zur Beschreibung von Klassifikationsregeln entworfen [Mor82, Qui90]. Formal werden sie als gerichtete geordnete Bäume formuliert. Die Knoten bilden dabei Bedingungen in Form diskreter oder gar boolescher Entscheidungsvariablen x_i ab, während die Blätter die daraus resultierenden Entscheidungen $f(x_1, \ldots, x_n)$ repräsentieren. Die Auswertung beginnt am Wurzelknoten und wird, je nach Wert seiner Entscheidungsvariable $x_i = k$ am k-ten Kindknoten, rekursiv fortgesetzt bis ein Blatt erreicht wird und somit die Entscheidung feststeht.

In der Praxis werden Entscheidungsbäume häufig als hierarchische Verkettung von `if/else` Verzweigungen realisiert. Daher erfreuen sie sich in überschaubaren Entscheidungsproblemen großer Beliebtheit.

Zum Einsatz in Fahrerassistenzsystemen wird i. d. R. eine geeignete Zustandsraumdarstellung gewählt, die anschließend mit den Entscheidungsvariablen geeignet unterteilt wird [Ard10]. Abbildung 2.4 stellt bspw. zwei Entscheidungsbäume dar, die von BMW – in Kombination mit Endlichen Zustandsautomaten – für hochautomatisiertes Fahren auf Autobahnen und einen Nothalteassistenten eingesetzt wurde [Ard11]. Hierbei wurden zwei separate Entscheidungsbäume, jeweils für die Quer- und Längsführung, entworfen. Die Entscheidungsvariablen wurden beispielsweise als Spurwechselwunsch aus der Situationsinterpretation abgeleitet. Schließlich gaben die Blätter Sollgrößen und Nebenbedingungen für die nachgelagerte Trajektorienplanung vor.

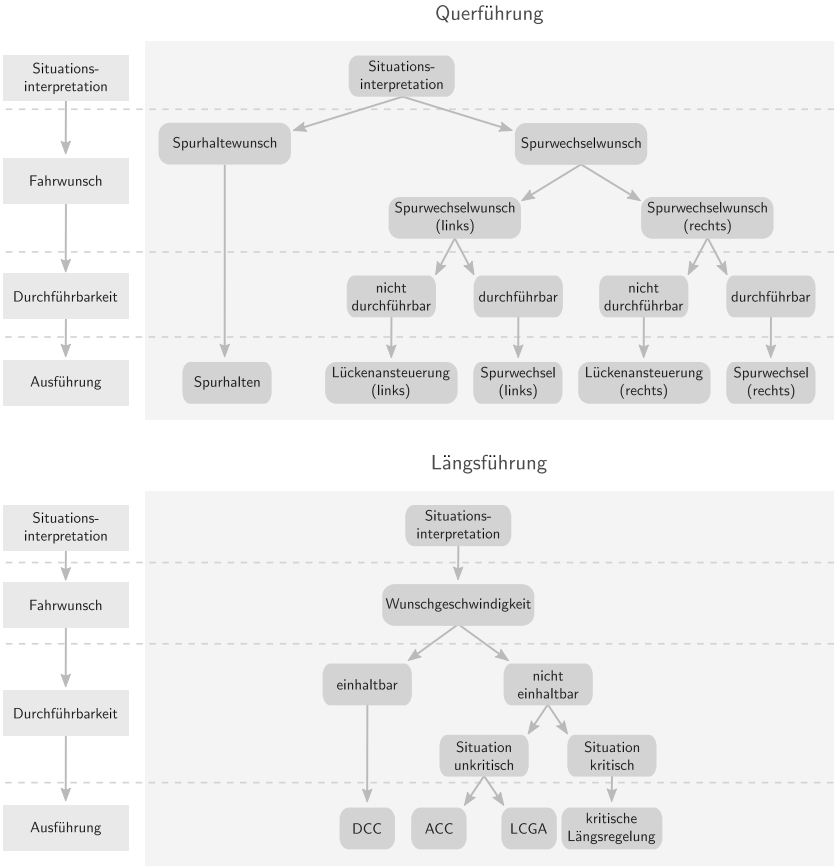


Abbildung 2.4: Entscheidungsbäume von BMW für hochautomatisiertes Fahren auf Autobahnen und einen Nothalteassistenten [Ard11].

Vergleichbar zu Endlichen Zustandsautomaten sind Entscheidungsbäume einfach zu implementieren und intuitiv begreifbar. Ein weiterer Vorteil ist die Modularität und Erweiterbarkeit: Teilbäume können unabhängig vom Rest des Baumes entworfen, entwickelt und in den Entscheidungsbaum hinzugefügt werden können. Die Blätter geben allerdings keinerlei (Erfolgs-)Status oder anderweitige Leistungskriterien zurück, sodass das Resultat eines Verhaltens die weitere Verhaltensauswahl nicht direkt beeinflussen kann.

2.2.3 Verhaltensbäume

Verhaltensbäume wurden zunächst in der Computerspiel-Entwicklung entworfen [Iov22] und sind seit 2012 immer häufiger auch in Robotikanwendungen im Einsatz [Bag12, Ögr12]. Im Bereich der Fahrerassistenzsysteme oder automatisierten Fahrzeuge wurde hingegen bisher nur eine Arbeit publiziert [Ols16]. Sie untersucht die Überlegenheit von Verhaltensbäumen gegenüber Endlichen Zustandsautomaten in Simulation, indem sie die Skalierbarkeit beider Methoden mit Softwaremetriken vergleicht. Eine ausführliche Übersicht und Einführung in Verhaltensbäume liefern [Iov22, Col18]. Dieser Abschnitt soll jedoch eine kurze praxisorientierte Einführung in die Methodik geben.

Verhaltensbäume zeichnen sich zunächst durch eine strikte funktionale Trennung zwischen Verhaltensentscheidung und -ausführung aus. Formal betrachtet sind sie zusammenhängende kreisfreie ungerichtete Graphen, dessen innere Knoten (sog. *Kontrollfluss-Knoten*) den Selektionsmechanismus festlegen, während die Blätter mögliche Verhalten (sog. *Aktions-Knoten*) sowie Bedingungen (sog. *Bedingungs-Knoten*) beschreiben.

Ausgehend vom Wurzelknoten wird der Baum mit einer festgelegten Frequenz, ähnlich einer Tiefensuche, ausgewertet. Dabei gibt ein ausgewerteter Knoten über seinen Rückgabewert an, ob er noch ausgeführt wird, erfolgreich abgeschlossen wurde oder fehlgeschlagen ist. Je nach Rückgabewert wertet der übergeordnete Kontrollfluss-Knoten weitere Kindknoten aus oder gibt seinen eigenen Status zurück. Dabei stehen verschiedene Arten von Kontrollfluss-Knoten zur Verfügung, u. a. zur Realisierung von Sequenzen, Fallback-Strukturen und Nebenläufigkeit.

Wird ein Bedingungs-Knoten ausgewertet, gibt er über seinen Rückgabewert zurück, ob die zugrundeliegende Bedingung erfüllt ist, ohne selbst Einfluss auf die Umwelt zu nehmen. Aktions-Knoten führen bei einem Aufruf hingegen das entsprechende Verhalten aus und geben über ihren Rückgabewert den Status dieses Verhaltens zurück. Die Aktions-Knoten beschreiben folglich die einzelnen Verhaltensoptionen eines Systems, während ihre Vorbedingungen über Bedingungs-Knoten modelliert werden. Durch die Unterscheidung zwischen Bedingungs- und Aktions-Knoten, sind die Vorbedingungen eines

Verhaltens allerdings von der eigentlichen Ausführung des Verhaltens entkoppelt.

Als Beispiel veranschaulicht Abb. 2.5 einen Verhaltensbaum eines interaktiven humanoiden Unterhaltungs-Roboters (vereinfacht aus [Col18]). Der Roboter soll sich nach Aufforderung des Bedieners setzen, aufstehen, Ball spielen oder sich verabschieden. Jede Aktion wird hierbei über einen Sequenz-Knoten mit ihrem korrespondierenden Bedingungs-Knoten, welcher zurückgibt, ob diese Aktion ausgeführt werden soll, verknüpft. Bspw. hängt die Aktion *Stand Up* von der Bedingung *Activity Stand Up* ab.

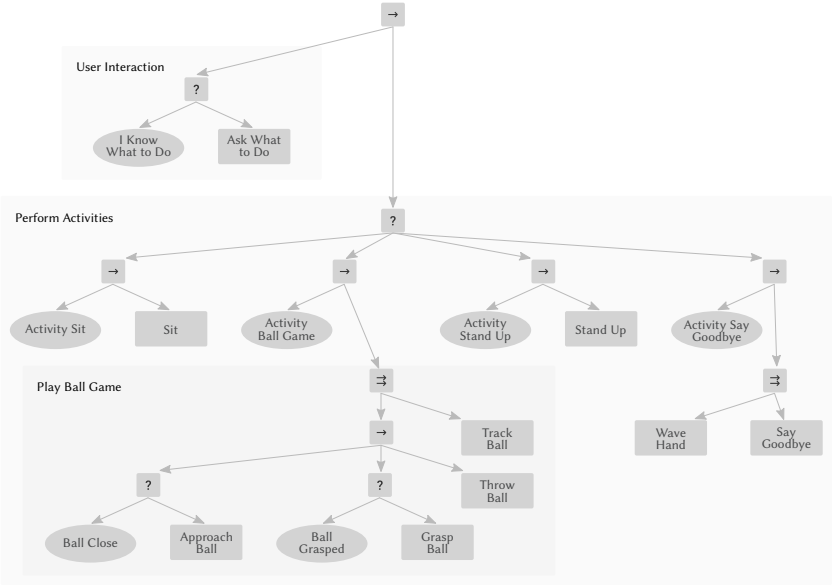


Abbildung 2.5: Verhaltensbaum für einen interaktiven humanoiden Roboter (angepasst aus [Col18]). Runde Blätter stellen die Bedingungs-Knoten und eckige Blätter die Aktions-Knoten dar. Sequenz-Knoten sind als Pfeil (→), Fallback-Knoten mit einem Fragezeichen (?) und Nebenläufigkeits-Knoten mit einem Doppelpfeil (⇉) gekennzeichnet.

Im Kontext der Fahrerassistenzsysteme oder des automatisierten Fahrens könnten Aktions-Knoten Fahrmanöver wie Folgefahrt, Fahrstreifenwechsel

oder Einparken realisieren. Um ein sicheres System zu entwerfen, müssten sie allerdings mit zuverlässigen Bedingungs-Knoten verknüpft werden.

Zusammenfassend generalisieren Verhaltensbäume viele andere Architekturen wie Hierarchische Endliche Zustandsautomaten und Entscheidungsbäume [Col17], bieten diesen gegenüber allerdings viele Vorteile: Sie überzeugen insbesondere durch Modularität, hierarchische Anordnung, Wiederverwendbarkeit der Komponenten, Reaktionsschnelligkeit und Interpretierbarkeit [Col18]. Im Vergleich zu Endlichen Zustandsautomaten ist vor allem die größere Flexibilität vorteilhaft. Einzelne Verhalten können innerhalb eines Verhaltensbaums (wieder-)verwendet werden ohne spezifizieren zu müssen, welchen Bezug sie zu den anderen Verhaltensoptionen haben [Bag12]. In der grafischen Darstellung ist der Selektionsmechanismus eines Verhaltensbaums zudem intuitiv sehr gut erfassbar und auch im Online-Betrieb leicht nachvollziehbar. Andererseits kann die Darstellung in der Praxis doch sehr umfangreich werden, da häufig jede Vorbedingung als separater Blattknoten modelliert wird. Die Sicherheit des Systems hängt außerdem, durch die erwähnte Entkopplung von Vorbedingungen und Ausführung eines Verhaltens, maßgeblich von der Anordnung der Knoten im Baum ab. Diese Nachteile werden von den Arbitrationsgraphen im folgenden Abschnitt adressiert.

2.2.4 Arbitrationsgraphen

Das Konzept der Verhaltensarbitration ist im Kontext des Roboterfußballs entstanden und kombiniert Ideen aus Brooks' verhaltensbasierter Subsumption [Bro86], wissensbasierten Architekturen wie Belief-Desire-Intention (BDI) [Rao92] und Programmierparadigmen wie der Objektorientierten Programmierung [Ste85]. Es wurde ausführlich in [Lau10] beschrieben und wird in diesem Abschnitt am selben Beispiel, dem Roboterfußball, zusammengefasst.

Fußball ist durch eine sich hochdynamisch verändernde Umwelt und zugleich etablierte Spieltaktiken und -strategien für Angreifer, Verteidiger und Torhüter gekennzeichnet. Daher wurde für die Anwendung im Roboterfußball eine Architektur gesucht, die es ermöglicht geringe Reaktionszeiten zu erreichen, bekannte Spieltaktiken aus einfachen Teilbausteinen zusammenzusetzen und

dabei deliberative Komponenten mit Ansätzen des Maschinellen Lernens zu kombinieren. Es sollte somit ein modulares Software-Framework entwickelt werden, das inkrementell erweiterbar ist, sich durch klare Schnittstellen auszeichnet und schließlich zu einem transparenten Entscheidungsprozess führt.

Das Konzept setzt auf atomare Verhaltensbausteine, die einfache Fähigkeiten und Verhaltensweisen abbilden. Diese werden nach Subsumptions-Prinzip mittels Arbitratoren zu komplexerem Systemverhalten, also Taktiken bis hin zu Strategien, kombiniert. Statt ein Problem also im wissensbasierten Top-Down Ansatz zu Teilproblemen zu zerlegen, wird komplexes Verhalten im Bottom-Up Entwurf iterativ aus einfachen Verhaltenskompetenzen zusammengesetzt.

Im Roboterfußball gibt es bspw. Verhaltensbausteine zum Dribbeln, Ziel ansteuern oder Schießen. In Kombination realisieren sie komplexeres Verhalten — von simpleren Taktiken wie einem Flankenangriff oder Doppelpass bis hin zu ganzen Angriffs- oder Verteidigungs-Strategien in höheren Abstraktionsebenen.

Als Eingang dient den Verhaltensbausteinen die aktuelle Situation $\boldsymbol{s}$ des Umfelds in Form von sensornahen Messdaten oder einem abstrahierten bis hin zu interpretierten Umweltmodell. Diese werden mit der *Stellfunktion* `Aktion` auf aktornahe Stellgrößen $\boldsymbol{u}$ abgebildet:

$$\boldsymbol{u} = \texttt{Aktion}(\boldsymbol{s}) \tag{2.1}$$

Im Roboterfußball besteht $\boldsymbol{u}$ bspw. aus der gewünschten Längs- und Winkelgeschwindigkeit eines Spielers sowie aus Anweisungen für seine Schussvorrichtung. Um dabei nicht nur reaktives, sondern auch deliberatives Verhalten zu ermöglichen, können die Verhaltensbausteine Situations- und Befehlshistorien anlegen oder randomisierte Verfahren nutzen.

Jeder *Verhaltensbaustein* bestimmt, neben seiner eigentlichen Aktion, über die sogenannte *Start-Bedingung* `Start` auch, ob seine Vorbedingungen erfüllt sind und das Verhalten somit aktuell überhaupt anwendbar ist. Ein Fußballspieler kann den Ball bspw. nur dann in Richtung Tor dribbeln, wenn er auch tatsächlich in Ballbesitz ist. Ist ein Verhaltensbaustein aktiv, gibt er über die

Algorithmus 1 : Generisches Arbitrationsverfahren

```
1  function BesteAnwendbareAktion(Situation s)
2  |   Filtere anwendbare Optionen A ⊂ O
3  |   Wähle Intention θ ∈ A aus                          // Arbitration
4  |   Bestimme u = Aktion_θ(s)
5  |   return u
6  end
7
8  while true do
9  |   Bestimme die aktuelle Situation s
10 |   Bestimme u = BesteAnwendbareAktion(s)
11 |   Führe u aus
12 end
```

Fortsetzungs-Bedingung `Fortsetzung` außerdem zurück, ob alle Bedingungen erfüllt sind, um das Verhalten weiterhin fortzuführen. Denn der Spieler kann nur so lange weiter dribbeln, bis er den Ball verliert oder das Dribbelziel erreicht hat. Über die Start- und Fortsetzungs-Bedingung bestimmt jeder Verhaltensbaustein also selbst, ob er in der gegebenen Situation anwendbar ist. Daher benötigt die aufrufende Instanz selbst kein Wissen über die Voraussetzungen zum Ausführen eines Verhaltensbausteins.

Generische *Arbitratoren* stellen eine Menge an Verhaltensbausteinen, an dieser Stelle auch als Optionen $\mathcal{O}$ bezeichnet, zu einer Taktik zusammen. Algorithmus 1 beschreibt das Arbitrationsverfahren im Allgemeinen. Zunächst wird also die aktuelle Situation $\boldsymbol{s}$ aus den aktuellsten Eingangsdaten bestimmt (Zeile 9) und den Verhaltensbausteinen zur Verfügung gestellt. Als Nächstes bestimmt der Arbitrator seine beste anwendbare Aktion. Hierzu filtert er aus seinen Optionen $\mathcal{O}$ jene Optionen $\mathcal{A}$ heraus, die aktuell anwendbar sind bzw. in deren Domäne die aktuelle Situation fällt. Dies signalisieren die Verhaltensbausteine über ihre Start- und Fortsetzungs-Bedingung:

$$\boldsymbol{s} \in \text{Domäne}(o) \Leftrightarrow \texttt{Start}_o(\boldsymbol{s}) \vee \texttt{Fortsetzung}_o(\boldsymbol{s}), \quad o \in \mathcal{O} \tag{2.2}$$

$$\mathcal{A} = \{o \in \mathcal{O} | \boldsymbol{s} \in \text{Domäne}(o)\} \tag{2.3}$$

Aus den anwendbaren Optionen $\mathcal{A}$ wählt der Arbitrator schließlich die bestmögliche Option aus, definiert sie als seine Intention θ und führt diese aus.

Für die eigentliche *Arbitration*, also die Bestimmung der bestmöglichen Option, kommen unterschiedliche Schemata infrage. Ein Prioritäts-Arbitrator hält seine Optionen in einer nach Priorität sortierten Liste und wählt daraus in jeder Iteration die erstbeste anwendbare Option. Der Abschluss-Arbitrator nutzt ebenfalls eine Prioritätenliste, stellt allerdings sicher, dass eine aktive Option nicht durch eine höher priorisierte Option unterbrochen wird, solange ihre Fortsetzungs-Bedingung wahr ist. Bei dem Zufalls-Arbitrator wird die Auswahl, mit ggf. gewichteten Wahrscheinlichkeiten, zufällig gefällt. Der Sequenz-Arbitrator dient schließlich dazu, Verhaltensoptionen sequenziell abzuarbeiten. Hierzu führt er seine aktuelle Intention so lange aus, bis ihre Fortsetzungs-Bedingung nicht mehr zutrifft. Ist gleichzeitig die Start-Bedingung der nächsten Option wahr, wird diese zur neuen Intention und ausgeführt.

Bei allen Arbitrationsschemata hängt die Start- und Fortsetzungs-Bedingung eines Arbitrators ausschließlich von den Start- und Fortsetzungs-Bedingungen seiner Optionen ab. Die Start-Bedingung des Prioritäts-, Abschluss- und Zufalls-Arbitrators ist zum Zeitpunkt k bspw. dann wahr, solange eine seiner Optionen eine wahre Start-Bedingung hat:

$$\texttt{Start}_{\text{prio}}(\boldsymbol{s}_k) = \bigvee_{o \in \mathcal{O}} \texttt{Start}_o(\boldsymbol{s}_k) \tag{2.4}$$

Der Sequenz-Arbitrator hat hingegen genau dann eine wahre Start-Bedingung, wenn seine erste Option o_1 eine wahre Start-Bedingung hat:

$$\texttt{Start}_{\text{seq}}(\boldsymbol{s}_k) = \texttt{Start}_{o_1}(\boldsymbol{s}_k) \tag{2.5}$$

Die Fortsetzungs-Bedingung eines Arbitrators zeigt wie bei den Verhaltensbausteinen an, ob er in der aktuellen Situation fortgesetzt werden kann. Bei den meisten Arbitratoren trifft dies bereits dann zu, wenn eine ihrer Optionen $\mathcal{O}$ eine wahre Start-Bedingung hat oder die letzte Intention θ_{k-1} fortgesetzt werden kann. Dies trifft für den Prioritäts, Abschluss und Zufalls-Arbitrator zu:

$$\texttt{Fortsetzung}_{\text{prio}}(\boldsymbol{s}_k) = \bigvee_{o \in \mathcal{O}} \texttt{Start}_o(\boldsymbol{s}_k) \vee \texttt{Fortsetzung}_{\theta_{k-1}}(\boldsymbol{s}_k) \tag{2.6}$$

Beim Sequenz-Arbitrator hängt die Fortsetzungs-Bedingung wiederum von seiner letzten Intention θ_{k-1} und der in $\mathcal{O}$ darauf folgenden Option $\theta_{k-1}+1$ ab:

$$\texttt{Fortsetzung}_{\text{seq}}(\boldsymbol{s}_k) = \texttt{Fortsetzung}_{\theta_{k-1}}(\boldsymbol{s}_k) \vee \texttt{Start}_{\theta_{k-1}+1}(\boldsymbol{s}_k) \quad (2.7)$$

Es ist an dieser Stelle hervorzuheben, dass die Arbitratoren ihre Wahl also nicht situationsspezifisch, sondern rein auf Basis der abstrakten Start- und Fortsetzungs-Bedingungen der Verhaltensoptionen treffen. Gegeben einer Menge anwendbarer Optionen $\mathcal{A}$, wählt ein Arbitrator seine Intention θ also ohne Berücksichtigung der Situation $\boldsymbol{s}$ aus. Dadurch wird eine starke Entkopplung der Situationsinterpretation und der Auswahllogik erreicht. Erstere wird über die Start- und Fortsetzungs-Bedingungen ausschließlich in den Verhaltensbausteinen durchgeführt und letztere vollständig von den Arbitratoren übernommen. Damit bleiben die Arbitratoren anwendungsunabhängig während die Verhaltensbausteine szenariospezifische Lösungen realisieren können.

Um nun aus einfachen Verhaltensoptionen und überschaubaren Taktiken komplexe Strategien zu entwerfen, können Arbitratoren und Verhaltensbausteine zu einem hierarchischen Graphen zusammengestellt werden. Hierzu gleichen Arbitratoren hinsichtlich ihrer Schnittstelle den Verhaltensbausteinen und können somit auch als solche interpretiert werden: Sie geben über ihre Start- und Fortsetzungs-Bedingungen an, ob sie in der aktuellen Situation $\boldsymbol{s}$ anwendbar sind, und bilden diese über die Stellfunktion `Aktion` auf Stellgrößen $\boldsymbol{u}$ ab. Auf den Algorithmus 1 übertragen, entspricht die `Aktion`-Funktion eines Arbitrators der `BesteAnwendbareAktion`($\boldsymbol{s}$) Funktion in Zeile 1. Dies ermöglicht es, einen Arbitrator wiederum als Verhaltensoption eines anderen Arbitrators einzusetzen.

Abbildung 2.6: Beispiel für einen hierarchischen Arbitrationsgraphen eines Angreifers im Roboterfußball. Aktuell nicht ausführbare Verhaltensoptionen sind grau hinterlegt, während die gewählte Option grün hervorgehoben ist.

Abbildung 2.6 zeigt beispielhaft einen zweistufigen Arbitrationsgraphen für einen Angreifer im Roboterfußball. Im Fließtext werden Verhaltensbausteine mit *Kursiver Schrift* und Arbitratoren mit Kapital Schrift gekennzeichnet. Die Taktik Angreifen besteht aus den sequenziell geschalteten Verhaltensoptionen *Zum Tor Dribbeln* und *Schießen*. Diese wird in der Gesamtstrategie des Angreifers (Angriff Strategie) mit höchster Priorität ausgeführt. Ist ein Angriff jedoch nicht möglich, soll sich der Angreifer *Dem Ball Nähern* oder, falls auch dies nicht möglich oder sinnvoll ist, an seiner Spielposition *Patrouillieren*.

Viele Vorteile von Arbitrationsgraphen werden insbesondere im Entwurfsprozess, der Wartung sowie der Weiterentwicklung einer darauf aufbauenden Verhaltensentscheidung deutlich. Zum einen können die Verhaltensbausteine dank der bereits erwähnten Entkopplung von Situationsinterpretation und Auswahllogik, voneinander unabhängig entworfen, entwickelt und getestet werden. Hierzu müssen lediglich die Start- und Fortsetzungs-Bedingungen spezifiziert und die Stellfunktion definiert werden. Erst in einem nachgelagerten Schritt werden die vorhandenen Verhaltensbausteine unter Berücksichtigung von anwendungsspezifischem Wissen mittels geeigneter Arbitratoren hierarchisch in einem Graphen angeordnet. Stellt sich zu einem späteren Zeitpunkt zudem heraus, dass noch weitere Verhaltensbausteine notwendig sind, können auch diese unabhängig entwickelt und in den Graphen eingefügt werden, ohne die bereits vorhandenen Bausteine anpassen zu müssen. Auch bei einer umfassenden Rekonfiguration des Graphen bleiben die Verhaltensbausteine i. d. R. unberührt.

Der modulare Aufbau der Arbitrationsgraphen bildet zugleich eine optimale Grundlage, um Maßnahmen zur Robustifizierung und Absicherung des Auswahlprozesses umzusetzen. So wird die Arbitration in Abschnitt 4.2 um eine Verifikation der Stellgrößen und Rückfallebenen erweitert. Zudem könnten Metriken zum Systemzustand einbezogen werden, um auch bspw. im Fall von Sensorstörungen in defensiveres Verhalten zu wechseln [Taş17].

2.3 Trajektorienplanung

Zweck der Trajektorienplanung ist es, ausgehend vom aktuellen Zustand, bspw. in Form eines geschätzten Umweltmodells, eine Trajektorie zu generieren, die ein gewünschtes Verhalten bzw. Manöver realisiert. Dabei können Trajektorien in kontinuierlicher oder diskreter Form modelliert werden (Definitionen 2.1 und 2.2).

Definition 2.1: Kontinuierliche Trajektorie

Eine Trajektorie beschreibt den zeitlichen Verlauf eines Zustands $\boldsymbol{x} \in \mathbb{R}^n$ über einen Zeithorizont $t \in [t_0, t_h]$:

$$\boldsymbol{x}(t) = (x_1(t), x_2(t), ...)^\mathsf{T} \tag{2.8}$$

Definition 2.2: Diskrete Trajektorie

In der diskreten Darstellung, die u. a. in numerischen Verfahren zur Trajektorienplanung Anwendung findet, wird eine Trajektorie durch ihre $N \in \mathbb{N}_0$ diskreten Stützpunkte beschrieben:

$$\boldsymbol{x}_l = (x_{l,1}, x_{l,2}, ...)^\mathsf{T}, \qquad l \in [0, N-1],\ l \in \mathbb{N}_0 \tag{2.9}$$

Beispiel 2.1: Trajektorie eines Fahrzeugs in 2D

Eine Fahrzeugtrajektorie lässt sich in Kartesischen Koordinaten mittels Position (x, y) und Orientierung ϕ modellieren [Wer17]:

$$\boldsymbol{x}(t) = (x(t), y(t), \phi(t))^\mathsf{T}, \qquad t \in [0, T] \tag{2.10a}$$

$$\boldsymbol{x}_l = (x_l, y_l, \phi_l)^\mathsf{T}, \qquad l \in [0, N-1],\ l \in \mathbb{N}_0 \tag{2.10b}$$

Im automatisierten Fahren beschreiben Trajektorien i. d. R. die Kartesischen Koordinaten x, y eines Fahrzeuges sowie seine Orientierung ϕ (Beispiel 2.1).

Da echtzeitfähige Methoden zur Trajektorienplanung zumeist nur lokale Lösungen liefern, wird häufig die Manöverentscheidung von der Trajektorienplanung getrennt. Die Manöver- oder auch Verhaltensentscheidung wählt zunächst die erfolgversprechendste Homotopieklasse aus (siehe dazu auch [Ben15]) und bestimmt ggf. eine grobe zugehörige Referenztrajektorie. Beim automatisierten Fahren beschreibt eine solche Homotopieklasse bspw. ob und in welche Lücke ein Fahrstreifenwechsel vorgenommen werden soll.

Zur Trajektorienplanung selbst bieten sich schließlich vielfältige Methoden an, darunter fallen u. a. Graphensuchverfahren, probabilistische Methoden, direkte Optimierung, Modellprädiktive Regelung sowie Ansätze des Maschinellen Lernens. Die Methoden arbeiten entweder direkt in *Kartesischen Koordinaten* oder den sog. *Frenet-Koordinaten* (Definition 2.3). Einige Ansätze planen zudem zunächst die Querbewegung in Form eines Sollpfades und danach die Längsbewegung entlang dieses Pfades, während andere die Quer- und Längsplanung gemeinsam lösen (auch als *2D Trajektorienplanung* bezeichnet).

Definition 2.3: Trajektorie in Frenet Koordinaten

Trajektorien können – z. B. zur entkoppelten Quer- und Längsplanung – in den sog. Frenet Koordinaten [Car16], also abhängig von der Bogenlänge s entlang einer Referenzkurve mit Normalenvektor $\boldsymbol{n}_r$, angegeben werden [Wer10]:

$$\boldsymbol{x}(s(t),d(t)) = \boldsymbol{r}(s(t)) + d(t)\boldsymbol{n}_r(s(t)), \tag{2.11}$$

wobei $\boldsymbol{r}$ den Lotfußpunkt entlang der Referenzkurve und d den orthogonalen Versatz dazu beschreibt.

Im Folgenden werden nur die für diese Arbeit relevanten Methoden beschrieben. Für eine umfassende Übersicht zur Manöverentscheidung und Trajektorienplanung sei u. a. auf [Pad16, Gon16, Wer17, Sch18, Kir21] verwiesen.

2.3.1 Pfadplanung

Wird die Trajektorienplanung in eine Quer- und Längskomponente entkoppelt, bestimmt die Pfadplanung dabei die Querführung des Fahrzeugs. Auch hierfür kommen vielfältige Verfahren infrage: Im Freiraum, wie bspw. Parkflächen, werden häufig Graphensuchverfahren mit kinematischen Bewegungsmodellen verwendet [Ban18], während in Szenarios mit Fahrstreifen u. a. Dynamische Programmierung [Wer10] oder auch Optimierung [Gut17] zum Einsatz kommt. Solange keine Hindernisse in den Fahrkorridor hineinragen, kann auch schlicht die Mittellinie eines Fahrstreifens – ggf. mittels Splines interpoliert – als Referenzpfad genutzt werden.

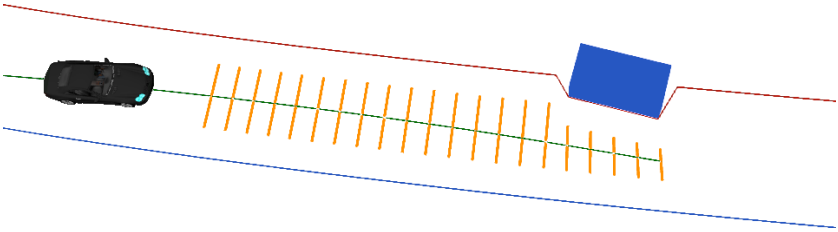


Abbildung 2.7: Die Pfadplanung bestimmt die Querführung eines Fahrzeugs unter Berücksichtigung der Fahrbahnbegrenzung und hineinragenden Hindernissen. Die Optimierungsachsen sind in Orange dargestellt, der geplante Pfad in Grün.

Abbildung 2.7 veranschaulicht beispielhaft die Pfadplanung mittels Optimierung. Die gelben Querbalken bilden die Optimierungsvariablen ab, während in Blau der resultierende Pfad dargestellt ist. Hierbei wurde folgende, aus [Zie14a] inspirierte, Kostenfunktion verwendet (vgl. Gleichung (2.15)):

$$J(\boldsymbol{x}_l) = \sum_{l=0}^{N-1} j_{\text{bor}}(\boldsymbol{x}_l) + j_{\text{pos}}(\boldsymbol{x}_l) + j_{\text{len}}(\boldsymbol{x}_l) + j_{\text{cur}}(\boldsymbol{x}_l) + j_{\text{dcur}}(\boldsymbol{x}_l), \tag{2.12}$$

wobei j_{bor} das Überschreiten der Fahrstreifenbegrenzungen, j_{pos} die Abweichung zur Fahrstreifenmitte und j_{len} die Länge des Pfades bestrafen. Durch die Einpreisung von j_{cur} und j_{dcur} werden zudem Krümmung und Krümmungsänderung minimiert.

2.3.2 Intelligent Driver Model

Das *Intelligent Driver Model (IDM)* [Tre00] ist ein mikroskopisches Verkehrsflussmodell, das das Längsverhalten von Fahrern bei der Folgefahrt beschreibt. Es kann, in Kombination mit einer vorgeschalteten Pfadplanung, als Folgefahrt-Regler oder gar zur Planung von Folgefahrt-Trajektorien verwendet werden:

$$\dot{s}_{\text{ego}} = \frac{\mathrm{d}s_{\text{ego}}}{\mathrm{d}t} = v_{\text{ego}} \tag{2.13a}$$

$$\dot{v}_{\text{ego}} = \frac{\mathrm{d}v_{\text{ego}}}{\mathrm{d}t} = a_{\max}\left(1 - \left(\frac{v_{\text{ego}}}{v_{\text{des}}}\right)^{\delta} - \left(\frac{d^*(v_{\text{ego}},\Delta v)}{d}\right)^2\right) \tag{2.13b}$$

$$d = s_{\text{obj}} - s_{\text{ego}} - l_{\text{obj}} \tag{2.13c}$$

$$\Delta v = v_{\text{ego}} - v_{\text{obj}} \tag{2.13d}$$

$$d^*(v_{\text{ego}},\Delta v) = d_{\min} + v_{\text{ego}}\, T + \frac{v_{\text{ego}}\,\Delta v}{2\sqrt{a_{\max}\, a_{\text{cmf}}}} \tag{2.13e}$$

wobei v_{ego} und v_{obj} die Geschwindigkeit des Egofahrzeugs und Referenzobjekts, d den Abstand, Δv die Geschwindigkeitsdifferenz, v_{des}, $a_{\max}$ und a_{cmf} die Wunschgeschwindigkeit, Maximalbeschleunigung und komfortable Bremsbeschleunigung, T den zeitlichen Wunschabstand (engl. „time headway“) und $d_{\min}$ den gewünschten Minimalabstand bezeichnet. Der Beschleunigungsexponent wird typischerweise auf $\delta = 4$ gesetzt.

2.3.3 Nichtlineare Optimierung

Bei der 2D-Trajektorienplanung mittels Nichtlinearer Optimierung wird ein, meist diskretes, nichtlineares Optimierungsproblem formuliert und mittels numerischer Methoden gelöst [Wer17]. Hierbei wird die Kostenfunktion J,

unter Berücksichtigung von m Ungleichungs- und q Gleichungsnebenbedingungen $\boldsymbol{g}$ und $\boldsymbol{h}$, minimiert:

$$J(\boldsymbol{x}_l), \qquad l \in [0, N-1] \tag{2.14a}$$

$$\boldsymbol{g}(\boldsymbol{x}_l) = 0, \qquad \boldsymbol{g} \in \mathbb{R}^m \tag{2.14b}$$

$$\boldsymbol{h}(\boldsymbol{x}_l) < 0, \qquad \boldsymbol{h} \in \mathbb{R}^q. \tag{2.14c}$$

Typische Kostenfunktionen umfassen die Abweichung zum Referenzpfad (bspw. die Fahrstreifenmitte) sowie der Sollgeschwindigkeit und bestrafen Beschleunigung, Ruck und Drehrate [Zie14a]:

$$J(\boldsymbol{x}_l) = \sum_{l=0}^{N-1} j_{\text{offs}}(\boldsymbol{x}_l) + j_{\text{vel}}(\boldsymbol{x}_l) + j_{\text{acc}}(\boldsymbol{x}_l) + j_{\text{jerk}}(\boldsymbol{x}_l) + j_{\text{jawr}}(\boldsymbol{x}_l). \tag{2.15}$$

Um eine erfolgreiche Optimierung zu ermöglichen, müssen die Kostenfunktionale mindestens einmal stetig differenzierbar sein. Die Kostenfunktionale zur Sollgeschwindigkeit, Beschleunigung, Ruck und Drehrate sind in diesem Fall sogar mehrfach stetig differenzierbar, was die Konvergenz weiter begünstigt. Damit auch der Abstand zu einem Polygon-basierten Referenzpfad mehrfach differenzierbar ist, muss entweder der Pfad oder die Abstandsfunktion geeignet interpoliert werden. Erstere können bspw. mittels Splines und letztere mittels sog. Pseudodistanzfunktionen [Zie14a] interpoliert werden.

2.4 Fehlertolerante Systeme

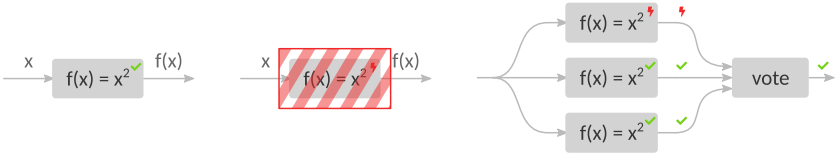
Im automatisierten Fahren können neben Hardwarefehlern eine Vielzahl von Softwareproblemen die Leistungsfähigkeit der Automatisierung und somit auch die Fahrsicherheit gefährden. Zu den Ursachen zählen u. a. Programmierfehler und Laufzeitfehler, wie Konvergenzprobleme einer Trajektorienoptimierung. Auch unsichere Trajektorien können als Fehler begriffen werden. Daher ist es unerlässlich beim Entwurf der Verhaltensentscheidung Maßnahmen zur Fehlerdiagnose und -behandlung mit einzubeziehen.

Die Forschung im Bereich der zuverlässigen und fehlertoleranten Systeme [Ech90, Lap95, Dub13, Mon99, Kor20] entwickelt Maßnahmen zum Entwurf von Hard- oder Software-Systemen, die trotz potenzieller Störungs- und Fehlerquellen eine möglichst hohe Zuverlässigkeit erreichen. Dabei können nach [Ech90] die Fehlerwahrscheinlichkeit, Überlebenswahrscheinlichkeit, mittlere Lebensdauer, Ausfallrate und Verfügbarkeit als Kenngrößen der Zuverlässigkeit dienen.

Die Begriffe *Störung*, *Fehler* und *Ausfall* werden umgangssprachlich häufig synonym verwendet und finden sogar in der Literatur unterschiedliche teils widersprüchliche Verwendung. [Dub13, Lap95, Mon99] unterscheiden zwischen einer Störung (engl. „fault“), einem Fehler (engl. „error“) und einem Ausfall (engl. „failure“). Eine Störung führt erst bei Aktivierung der gestörten Komponente (z. B. durch einen Funktionsaufruf) zu einem Fehler (bspw. einem falschen Ergebnis). Wird ein Fehler erkannt, kann dieser entweder behandelt und behoben werden oder im schlimmsten Falle zu einem Funktions- oder Systemausfall führen. Das deutschsprachige Standardwerk von Echtle [Ech90] unterscheidet hingegen nicht zwischen Störungen und Fehlern, sondern spricht stattdessen von *Fehlzuständen*, die sich auf eine „Verletzung der inneren Spezifikation“ beziehen, während der Begriff *Funktionsausfall* die „Verletzung der äußeren Spezifikation“ beschreibt. Auch [Kor20] definiert den Funktionsausfall als Manifestation eines Fehlzustands, verwendet für den Fehlzustand allerdings die engl. Begriffe *fault* und *failure* synonym und bezeichnet den Funktionsausfall als *error*. Auch wenn die Unterscheidung zwischen einem Fehler und seiner Ursache (der Störung) sinnvoll erscheint und eine Diagnose zu präzisieren hilft, ist eine konsistente Übersetzung aller engl. domänenspezifischen Fachbegriffe wie in [Ech90] vorrangig. Daher werden im weiteren Verlauf dieser Arbeit die Definitionen nach [Ech90] verwendet.

Die Maßnahmen zur Steigerung der Zuverlässigkeit lassen sich in Fehlervermeidung, -beseitigung, -toleranz und -vorhersage kategorisieren [Dub13]. Methoden zur Fehlervermeidung und -beseitigung setzen in der Entwurfs-

und Entwicklungsphase an und setzen u. a. auf Qualitätskontrolle und Verifikation (vgl. Abschnitt 2.5.1). Der Bereich der Fehlertoleranz beschreibt Ansätze, mit denen Fehlzustände im Betrieb festgestellt und Funktionsausfälle verhindert werden können. Die Fehlervorhersage beschäftigt sich schließlich damit, im laufenden Betrieb die Anzahl fehlerhafter Komponenten zu schätzen und somit bevorstehende Ausfälle vorherzusagen.



(a) Die Fehlerbehebung stellt einen fehlerfreien Zustand her.

(b) Die Fehlerausgrenzung beseitigt fehlerhafte Komponenten.

(c) Die Fehlerkompensierung stellt ein fehlerfreies Ergebnis her.

Abbildung 2.8: Maßnahmen zur Fehlerbehandlung.

Neben der Fehlervermeidung ist die Fehlertoleranz eine der wichtigsten Säulen zuverlässiger Systeme und steht daher in der Literatur besonders im Fokus. Zur Fehlertoleranz gehört die Fehlerdiagnose – welche es ermöglicht einen vorhandenen Fehlzustand zu erfassen – und die Fehlerbehandlung – die den Fehlzustand behebt, kompensiert oder ausgrenzt. Abbildung 2.8 stellt die Methoden zur Fehlerbehandlung dar: Für die Fehlerbehebung wird die fehlerhafte Komponente wieder in einen fehlerfreien Zustand (zurück-)gesetzt (z. B. über eine Zustandshistorie). Als Fehlerkompensierung bezeichnet man hingegen das Berechnen eines fehlerfreien Ergebnisses, trotz fehlerhafter Komponente (bspw. über Redundanz und Diversität). Die Fehlerausgrenzung entfernt schließlich die fehlerhafte Komponente als letzte drastische Maßnahme dauerhaft aus dem System (engl. auch „reconfiguration“). Abbildung 2.9 fasst die Maßnahmen, die in zuverlässigen Systemen zum Einsatz kommen, strukturiert zusammen.

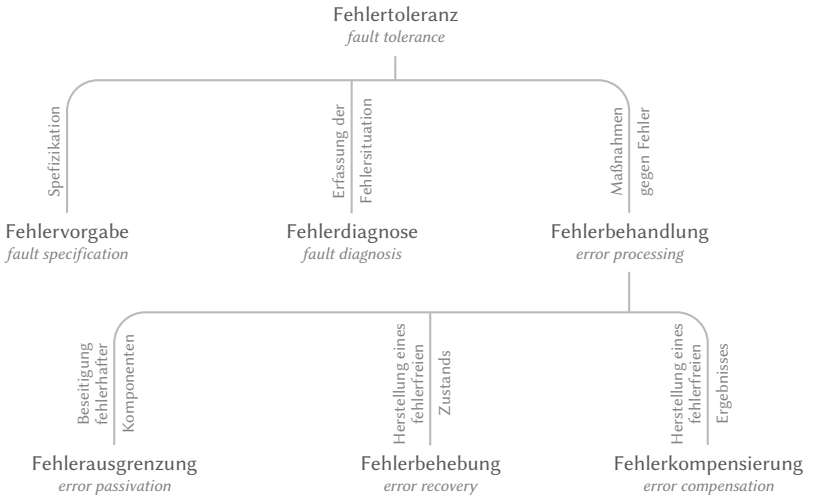


Abbildung 2.9: Maßnahmenkategorien zur Entwicklung fehlertoleranter Systeme, in Anlehnung an [Ech90].

2.5 Verkehrssicherheit

Der zunächst weitgefasste Begriff Verkehrssicherheit beschreibt „die Abwesenheit von unvertretbaren Risiken bei der Ortsveränderung von Objekten (z. B. Güter, Personen, Nachrichten) in einem definierten (Verkehrs-)System“ [Dre09]. Für den Kontext des automatisierten Fahrens ist dabei die spezifischere Verkehrsmittelsicherheit, also die Abwesenheit von unvertretbaren Risiken und Gefahren bezogen auf die Verkehrsmittel, von Bedeutung. Diese umfasst sowohl aktive unfallvermeidende Maßnahmen, sowie passive Unfallfolgen mindernde Maßnahmen.

Dieser Abschnitt 2.5 greift zwei für diese Arbeit relevanten Themenfelder der aktiven unfallvermeidenden Maßnahmen auf: Die Sicherheitsanalyse in Abschnitt 2.5.1 wird zur Fehlervermeidung beim Systementwurf eingesetzt, während die Verhaltensverifikation aus Abschnitt 2.5.2 der Fehlererkennung und -behandlung zur Laufzeit dient.

2.5.1 Sicherheitsanalyse

Das Feld der Sicherheitsanalyse insgesamt, sowie die darunter fallende Zuverlässigkeitsanalyse im Besonderen, haben viele in der Automobilbranche etablierte Methoden und Standards hervorgebracht. Die ISO26262 [ISO18] erläutert bspw. wie mittels einer Gefährdungsanalyse und Risikoabschätzung die funktionale Sicherheit sicherheitsrelevanter elektrischer/elektronischer Systeme im Kraftfahrzeug (KFZ) gewährleistet werden kann (siehe auch [Ros16]). Die Methoden der sog. „Safety of the Intended Functionality“ (SOTIF) erweitern die ISO26262 um die Bereiche Umfeldwahrnehmung, Mensch-Maschine-Schnittstelle (HMI) und erwartbare Fehlnutzung durch den Anwender. Deduktive Methoden, wie die Fehlzustandsbaumanalyse (FTA) [DIN07], identifizieren kausale Ketten zwischen Gefahren und deren Ursachen, um die Wahrscheinlichkeit festgelegter (i. d. R. kritischer) Ereignisse zu bestimmen. Induktive Methoden, wie die Fehlzustandsart- und -auswirkungsanalyse (FMEA) [DIN06], werden genutzt, um mögliche Produktfehler bereits während des Systementwurfs zu erkennen und ihr Risiko zu bewerten.

Insgesamt sind diese Methoden und Standards allerdings nicht ausreichend, um die Sicherheit von hoch- oder voll-automatisierten Fahrzeugen zu gewährleisten [Way20b]. Daher ist die Sicherheitsanalyse automatisierter Fahrzeuge noch ein aktives Forschungsfeld [Mer18, Apt19, Sch20, Way20a, Way20b] aus dem zurzeit viele neue Leitlinien und Normen entstehen [NHT17, SAE21, SAE20, BSI20a, BSI20b, ISO21]. Eine gute und umfangreiche Zusammenfassung über diese Publikationen verschafft [Way20b]. Im Folgenden sollen die für diese Arbeit wichtigsten Begriffe und Konzepte, in Anlehnung an [Way20b] und [NHT17], zusammengefasst werden.

Operational Design Domain

Um u. a. einen Szenarienkatalog zur Szenarien-basierten Validierung (siehe späteren Abschnitt zu Systemsicherheit und Validierung) und weitere Sicherheitsanforderungen an ein automatisiertes Fahrzeug herzuleiten, wurde das

Konzept der Operational Design Domain (ODD), das in Definition 2.4 erläutert wird, eingeführt.

Definition 2.4: Operational Design Domain [SAE21]

Die ODD eines gegebenen Fahrerassistenzsystems oder einer seiner Teilfunktionen legt die Betriebsbedingungen fest, für welche diese Funktion ausgelegt ist.
Diese schließen unter anderem Anforderungen an die Szenerie (z. B. bestimmte erforderliche Verkehrs- oder Fahrbahnmerkmale) sowie an die dynamischen Elemente der Szene (wie Verkehrsdichte oder Anwesenheit von vulnerable Verkehrsteilnehmer (VRUs)) und Umweltbedingungen (u. a. Wetter oder GPS-Empfang) ein.

Eine ausführliche Auflistung und Spezifikation möglicher ODD Elemente liefern unter anderem die British Standards Institution [BSI20b] und das Automated Vehicle Safety Consortium™[SAE20]. Im Folgenden soll daraus eine mögliche Kategorisierung zusammengefasst werden.

Die unterstützten Szenerien werden unter anderem beschrieben durch

- geografisch oder rechtlich festgelegte Zonen wie Stadtbezirke,
- befahrbare Flächen und Straßentypen wie Parkflächen oder Autobahnen,
- Kreuzungsarten und -infrastruktur wie dreiarmige Knotenpunkte mit Lichtsignalanlage,
- besondere Infrastruktur wie Mautschranken oder Tunnel,
- Fahrbahnelemente wie Leitpfosten und
- vorübergehende Straßenbauten wie Baustellen.

Zur Beschreibung dynamischer Elemente einer ODD zählt

- die Verkehrsdichte,
- Art und Zustand anderer Verkehrsteilnehmer und
- der Zustand des Ego-Fahrzeugs (z. B. seine Geschwindigkeit).

Die Umweltbedingungen, für die die Assistenzfunktion ausgelegt ist, werden im Wesentlichen definiert durch

- Wetterbedingungen wie Schneefall oder Regen,

- Betriebszeiten wie zur Tageszeit oder nachts,
- die Signalqualität von Satellitennavigationssystemen wie GPS und
- die Verfügbarkeit von Datennetzen wie zum Beispiel V2X[1]-Mobilfunk.

Verhaltenskompetenzen

Von der National Highway Traffic Safety Administration (NHTSA) wurden 28 grundlegende Verhaltenskompetenzen, wie sie für die Szenarien-basierte Validierung verwendet werden können, vorgeschlagen und von Waymo auf insgesamt 47 Stück erweitert [Way20a]. Darunter fallen sowohl ganze Fahrmanöver wie die Folgefahrt oder Fahrstreifenwechsel als auch Teilaufgaben, wie das Einhalten der zugelassenen Maximalgeschwindigkeit oder Fällen korrekter Vorfahrtsentscheidungen. Häufig sind diese Verhaltenskompetenzen an Wahrnehmungsaufgaben gekoppelt und als „Erkenne und reagiere angemessen"-Anweisung formuliert, bspw. „Erkenne und reagiere auf ein einscherendes Objekt". Die detaillierte Auflistung der Verhaltenskompetenzen aus [Way20a] kann Anhang A entnommen werden.

Fehlererkennung und Reaktion

Ein automatisiertes System muss sich stets überwachen und in der Lage sein, rechtzeitig Fehlfunktionen und degradierte Zustände in der Hardware oder den automatisierten Fahrfunktionen [Taş17], sowie das Verlassen der ODDs (z. B. durch sich ändernde Wetterbedingungen) zu erkennen. Handelt es sich um eine unkritische Störung, kann das Fahrzeug ggf. in einem degradierten Modus, also bspw. bei reduzierter Geschwindigkeit, die Fahrt fortsetzen. Bei schwerwiegenderen Fehlern muss das System in einen risikominimalen Zustand (MRC), wie in Definition 2.5 aus dem „Gesetz zum autonomen Fahren" [Bun21] festgelegt, überführt werden. Je nach Schwere des Fehlers wird hierfür entweder in der Hardware- oder Verhaltens-Ebene eine angemessene

[1] V2X umfasst Fahrzeug-Fahrzeug-Kommunikation (V2V), Fahrzeug-Infrastruktur-Kommunikation (V2I) und Kommunikation zur Flottenverwaltung (V2F).

Reaktion bewirkt. Bei weniger schwerwiegenden Funktionsstörungen kann die Verhaltens-Ebene ein sog. risikominimierendes Manöver durchführen, das das Fahrzeug bspw. auf einem Standstreifen zum Stillstand bringt. Bei einem schwerwiegenden Systemausfall, der auch die Verhaltens-Ebene betrifft, betätigt die Hardware-Ebene eine Notbremsung.

Definition 2.5: Risikominimaler Zustand [Bun21]

Ein risikominimaler Zustand (MRC) ist ein Zustand, in den sich das Kraftfahrzeug mit autonomer Fahrfunktion auf eigene Veranlassung oder auf Veranlassung der technischen Aufsicht selbständig versetzt, um unter angemessener Beachtung der Verkehrssituation die größtmögliche Verkehrssicherheit für andere Verkehrsteilnehmende und Dritte zu gewährleisten.

Definition 2.6: Risikominimierendes Manöver [Apt19]

Ein risikominimierendes Manöver (MRM) ist ein Manöver, das das Fahrzeug in einen MRC überführt.

In [Apt19], einer gemeinsamen Publikation zahlreicher Automobilhersteller und -zulieferer, darunter Audi, BMW, Daimler, VW, Intel und Here, werden mögliche MRCs und MRMs vorgestellt:

Beispiel 2.2: Risikominimale Zustände aus [Apt19]

Je nach Fehlerschwere und ob ein Fahrzeugführer anwesend ist, kommen verschiedene MRCs infrage.

Übernahme durch Fahrzeugführer Der Fahrzeugführer hat die Fahraufgabe vollständig übernommen.

Eingeschränkter Betrieb Fahrzeug ist noch innerhalb der eingeschränkten Fahrfunktionen betriebsbereit. Je nach Funktionsdefinition und verbleibenden Fähigkeiten kann es mehrere eingeschränkte Betriebszustände geben.

Betrieb einstellen Diese Bedingung beschreibt einen Fahrzeugzustand, der eine sichere Abschaltung der Funktion ermöglicht.

Beispiel 2.3: Risikominimierende Manöver aus [Apt19]

Je nach angestrebten MRC kommen verschiedene MRMs infrage, die sich insbesondere bzgl. Komfort- und Funktionseinschränkung unterscheiden.

Aufforderung zur Übernahme Übernahme durch den Fahrzeugführer anfordern.

Funktionsumfang einschränken Überleiten in den eingeschränkten Betrieb. Je nach MRC und dem aktuellen Zustand sind mehrere MRM-Varianten möglich.

Komfortabler Halt Komfortabler Übergang zum Ende des Betriebs

Sicherer Halt Bei schwerwiegenden Ausfällen ist ein schneller, aber sicherer Übergang zum Betriebsende erforderlich.

Nothalt Bei hinreichend seltenen schweren Systemausfällen wird ein Notstopp eingeleitet, um das Risiko zu minimieren und den Betrieb einstellen zu können.

Systemsicherheit und Validierung

Um die Sicherheit des Gesamtsystems, also eines im öffentlichen Straßenverkehr agierenden hoch- oder voll-automatisierten Fahrzeuges, zu gewährleisten, müssen adäquate Methoden der Sicherheitsanalyse in den Entwicklungsprozess integriert werden, währenddessen laufend die Risiken analysiert und auch im Betrieb überwacht werden. Waymo unterscheidet dabei drei Ebenen [Way20b] (Abb. 2.10): Die Hardware-Ebene umfasst das physische Fahrzeug, die Sensorik, die Aktorik, die Rechenplattform sowie die Cybersicherheit. Diese können wie bisher mit den klassischen Methoden, u. a. FMEA, FTA, usw. verifiziert und validiert werden.

Die Verhaltens-Ebene schließt die Trajektorien- und Verhaltensplanung ein. Aufgrund der Interaktion mit anderen Verkehrsteilnehmern und ihrem teils nicht-deterministischen Verhalten sind die klassischen Methoden der Sicherheitsanalyse in dieser Ebene nicht ausreichend, sodass ein dreiteiliger Ansatz gewählt wurde:

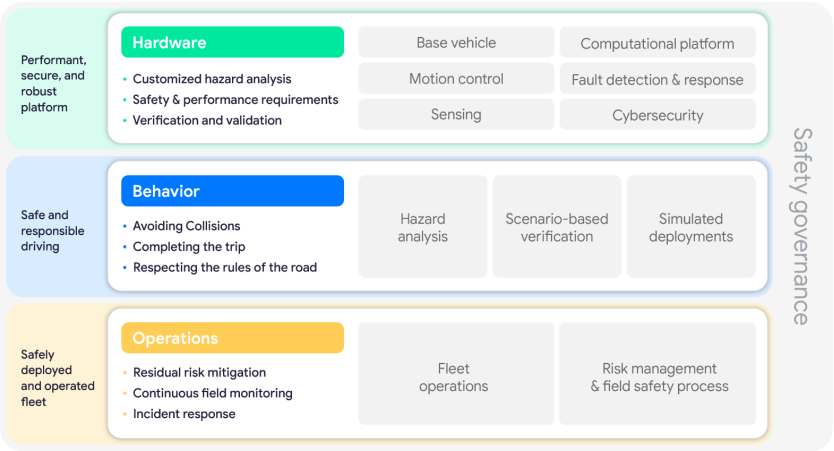


Abbildung 2.10: Die Sicherheitsstrategie von Waymo gliedert sich in drei Ebenen [Way20b]. Copyright © 2020 Waymo LLC

Zusätzlich zur Gefahrenanalyse mittels FTA, FMEA und einer systemtheoretischen Prozessanalyse wird eine Szenario-basierte Verifikation mithilfe eines ständig zu erweiternden Szenarienkatalogs durchgeführt. Dieser Katalog soll sowohl grundlegende Verhaltenskompetenzen, aufbauend auf den Empfehlungen der US-amerikanischen NHTSA, sowie erwartbare kritische Szenarien abdecken. Zudem soll der Katalog das gesamte Spektrum der ODDs abdecken.

Schließlich werden neue Softwareversionen in einer sog. simulierten Bereitstellung (engl. „Simulated Deployments“) getestet. Diese bestehen einerseits aus Simulationen, die die Software auf umfangreich aufgezeichneten Daten neu auswerten. Bei immer mehr Realfahrten aus der semi-automatisierten Flotte und damit immer größeren Realdatensätzen ist dieser Ansatz sehr gut skalierbar. Andererseits wird ein solcher Softwarestand auf Fahrzeugen mit Sicherheitsfahrern auch im Realversuch getestet. Sollten die Fahrer eingreifen, werden diese manuell gesteuerten Abschnitte nochmals in Simulation überprüft. Hiermit sollen die in der Daten-basierten Evaluation attestierten Verbesserungen an Glaubwürdigkeit gewinnen.

Sowohl in der Szenario-basierten Verifikation als auch der simulierten Bereitstellung wird die Sicherheit des Systems anhand folgender Metriken bemessen:

Unfallvermeidung Die in Simulation geschätzte Unfallrate des automatisierten Fahrzeugs wird u. a. mit Unfallraten des Menschen verglichen.

Abschluss automatisierter Fahrten Die Anzahl erfolgreich abgeschlossener Fahrten im automatisierten Modus gibt Aufschluss über die Leistungsfähigkeit des Gesamtsystems.

Einhalten von Verkehrsregeln Schließlich sollte das Fahrzeug auch die gesetzlich vorgeschriebenen oder gesellschaftlich etablierten Verkehrsregeln einhalten, um Gefahrensituationen zu vermeiden.

Die dritte Ebene adressiert schließlich den Betrieb einer ganzen Flotte automatisierter Fahrzeuge und den damit verbundenen Sicherheitsfragen. Mittels Methoden des Risikomanagements soll das Gesamtrisiko des Flottenbetriebs eingedämmt werden. Außerdem wird klar geregelt, welche Daten bei einem Unfall aufgezeichnet und gesichert werden müssen und wie der Betreiber nach dem Unfall handeln wird, z. B. wie er mit den zuständlichen örtlichen Behörden in Kontakt treten wird.

2.5.2 Verhaltensverifikation

Das Problem der Risikoanalyse automatisierter Fahrzeuge, insbesondere im Hinblick auf die Sicherheit eines geplanten Verhaltens, wird umfangreich erforscht. [Lef14] bietet einen umfassenden Überblick über das Forschungsfeld.

Die meisten der vorgestellten Methoden versuchen allerdings, das Risiko über Verhaltensmodelle abzuschätzen. Hierzu entwickeln sie Verhaltensmodelle für die verschiedenen Verkehrsteilnehmer, prädizieren die Szene unter Annahme dieser Modelle und prüfen schließlich die geplante Trajektorie auf Kollisionen oder versuchen Abweichungen von den Modellen zu erkennen. In jedem Fall gehen solche Risikobewertungsansätze davon aus, dass alle möglichen Manöver in einem bestimmten Szenario modelliert werden können oder dass unerwartete Situationen zuverlässig erkannt werden können. Darüber

hinaus wird in vielen Fällen die Unvollständigkeit eines Umgebungsmodells überhaupt nicht berücksichtigt.

Risikoanalyse und Manöverplanung unter Unsicherheiten

In letzter Zeit hat das Thema verdeckungsbewusste Risikobewertung und Verhaltensgenerierung an Aufmerksamkeit gewonnen [Hoe17, Lee17, Chu09, Bre14, Isa08, Sad04, Bou14, Zha16, Taş18a]. Die präsentierten Ansätze reichen von einfachen Sichtbarkeitsmodellierungen [Isa08], die das Tracking von zuvor erkannten Hindernissen verbessert, bis hin zu komplexen mehrschichtigen Umgebungsmodellen [Hoe17].

Einige von ihnen berücksichtigen explizit Unsicherheiten [Zha16, Hoe17, Bre14, Taş18a], während andere die Sichtbarkeitsanalyse nutzen, um Geschwindigkeitsbeschränkungen für die Trajektorienplanung zu definieren [Chu09, Lee17, Taş18a]. Nur drei dieser Veröffentlichungen weisen wenigstens passive Bewegungssicherheit (Definition 2.7) nach [Bou14, Taş18a] oder beweisen Kollisionsfreiheit für diskrete Zeitschritte ihrer Trajektorien [Zha16].

Definition 2.7: Passive Bewegungssicherheit [Mac09]

Ein Zustand $\boldsymbol{x}$ des mobilen Robotersystems r ist genau dann sicher im Sinne der passiven Bewegungssicherheit, wenn mindestens ein Bremsmanöver existiert, das bei $\boldsymbol{x}$ beginnt und bis T_b kollisionsfrei ist, wobei T_b genau die Zeit ist, in der r im Stillstand ist. Folglich kollidiert niemals der Roboter selbst in ein anderes Objekt, sondern höchstens umgekehrt.

Ein interessanter Ansatz wird in [Hoe17] vorgestellt. Die Autoren modellieren und prädizieren die Umgebung in einer dreischichtigen Grid-Map: Objektbasierte, objektfreie und unbeobachtbare Umgebung. Während der Planung betrachten sie eine Zelle als belegt, sobald eine der drei Schichten als belegt vorhergesagt wird.

Dennoch minimieren diese Methoden allenfalls das Risiko von Kollisionen [Bre14, Chu09, Hoe17, Lee17] und bieten keine oder zu geringe Sicherheitsgarantien [Bou14, Zha16]. Einige der ersten Ansätze weisen sogar

grundsätzliche Probleme auf, wie z. B. dass noch nicht beobachtete Hindernisse ignoriert werden [Isa08], dass verdeckte Bereiche ohne Prädiktion berücksichtigt werden [Sad04] oder dass ausschließlich Verdeckungen durch statische Hindernisse untersucht werden [Chu09]. Fahrzeuge mit solchen Verfahren der Risikoanalyse werden folglich der geplanten Trajektorie folgen, solange sie kein nennenswertes Risiko erkennen. Dies kann – aufgrund der unzureichenden Berücksichtigung von Unsicherheiten – jedoch zu Situationen führen, in denen eine Kollision unvermeidlich ist. Daher müssen auch solche Methoden, bspw. durch Einbindung in Arbitrationsgraphen mit Verifikationslogik (siehe folgenden Teilabschnitt und Abschnitt 4.2), abgesichert werden.

Verhaltensverifikation mittels Erreichbare-Mengen-Analyse

Um diese Herausforderungen zu lösen, schlagen Althoff u. a. [Alt16] ein Verfahren zur Verhaltensverifikation mittels Erreichbare-Mengen-Analyse vor, das die Sicherheit einer geplanten Trajektorie unter klar spezifizierten Annahmen beweisen kann. Orzechowski u. a. [Orz18] erweitern die Methode, um auch Verdeckungen und begrenzte Sensorreichweite bzw. -abdeckung zu berücksichtigen. Im Kern basiert die Methode zum einen darauf, zusätzlich zur Soll-Trajektorie eine Fail-Safe-Trajektorie zu planen, die das Fahrzeug in einen sicheren Zustand überführt und der Verifikation standhalten soll. Zum anderen werden alle möglichen Verhalten anderer Verkehrsteilnehmer mit einer Worst-Case-Belegung überapproximiert. Überlappt die Belegung der geplanten Fail-Safe-Trajektorie schließlich mit keiner der prädizierten Worst-Case-Belegungen, führt die Fail-Safe-Trajektorie das Fahrzeug in jedem Fall kollisionsfrei in einen sicheren Zustand und ist somit – unter den Annahmen des Verfahrens – beweisbar sicher. Abbildung 2.11 veranschaulicht den Abgleich der Belegungen einer geplanten Ego-Trajektorie mit den Worst-Case-Belegungen eines entgegenkommenden Fahrzeugs. Der Prädiktionshorizont wurde dabei in drei Zeitintervalle unterteilt.

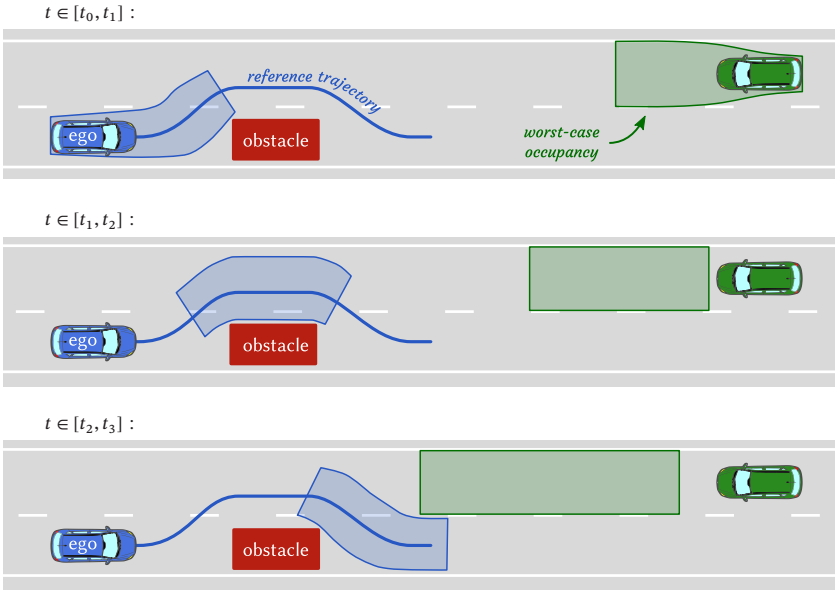


Abbildung 2.11: Belegungen einer geplanten Ego-Trajektorie sowie prädizierte Worst-Case-Belegungen eines entgegenkommenden Fahrzeugs, allerdings ohne Berücksichtigung von Verdeckungen oder begrenzter Sensorreichweite, nach [Alt16].

Solange also für eine geplante Soll-Trajektorie eine beweisbar sichere Fail-Safe-Trajektorie gefunden werden kann, auf die im Zweifel im folgenden Planungsintervall ausgewichen werden kann, kann auch die Soll-Trajektorie selbst im aktuellen Planungsintervall sicher ausgeführt werden. Andernfalls muss auf die im vorigen Intervall verifizierte Fail-Safe-Trajektorie zurückgegriffen werden. Abbildung 2.12 fasst den daraus resultierenden Prozess der Verhaltensgenerierung im Zusammenspiel mit einer Verifikation mittels Erreichbare-Mengen-Analyse zusammen.

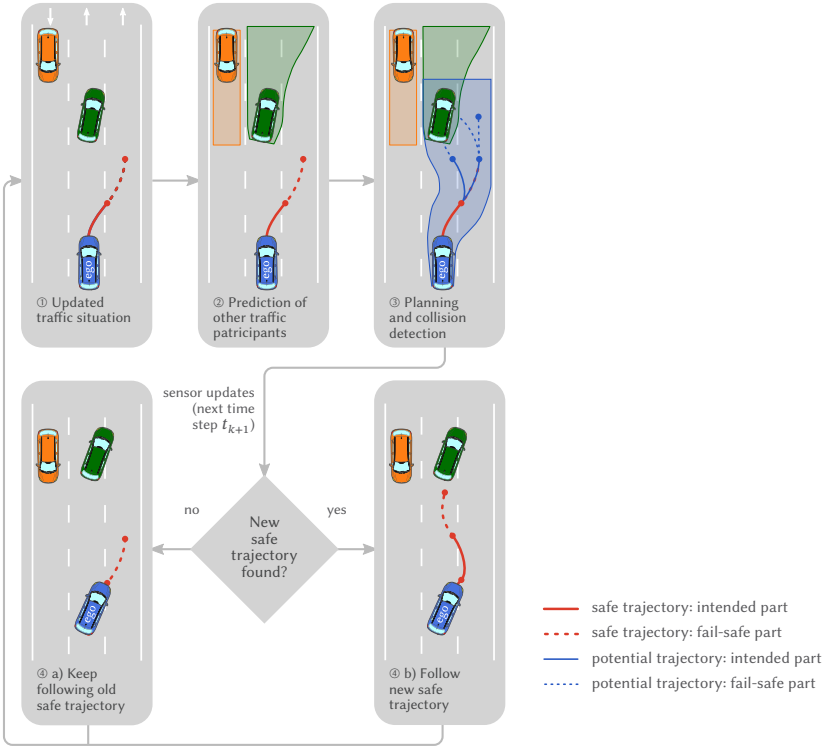


Abbildung 2.12: Ablauf einer Verhaltensverifikation mit der Erreichbare-Mengen-Analyse, nach [Alt16].

Mittelpunkt der Methode ist folglich die Prädiktion der Worst-Case-Belegungen anderer Verkehrsteilnehmer. Gegeben sei hierzu das dynamische System $\dot{\boldsymbol{x}} = f(\boldsymbol{x}(t), \boldsymbol{u}(t))$ mit Zustand $\boldsymbol{x}$, Eingangsgröße $\boldsymbol{u}$ und Zeit t. Dann beschreibt das Tupel $M = (f, \mathcal{X}_0, \mathcal{U})$, wobei $\boldsymbol{x}(0) \in \mathcal{X}_0$ und $\boldsymbol{u} \in \mathcal{U}$, ein Modell dieses Systems. Als Erreichbare-Menge bezeichnet man schließlich alle gemäß M möglichen zukünftigen Zustände zum Zeitpunkt $t = r$:

$$\mathcal{R}(M, r) = \left\{ \int_0^r f(\boldsymbol{x}(t), \boldsymbol{u}(t))\, dt \;\middle|\; \boldsymbol{x}(0) \in \mathcal{X}_0, \forall t : \boldsymbol{u}(t) \in \mathcal{U} \right\}. \tag{2.16}$$

Eine exakte Prädiktion der Erreichbaren-Menge eines Verkehrsteilnehmers ist u. a. aufgrund der nicht-linearen hybriden[1] Dynamik nicht effizient möglich. Andererseits ist zur Kollisionsvermeidung ohnehin hauptsächlich relevant welchen Raum die Verkehrsteilnehmer belegen könnten. Daher werden einerseits nur Belegungen prädiziert und diese andererseits effizient überapproximiert.

Formal betrachtet bildet die Belegung $\mathcal{B}(t)$ eine Projektion der Erreichbaren-Menge auf einen Unterraum $\text{proj}(\boldsymbol{x}) = [x, y, \phi]^\mathsf{T}$, mit der Position x, y und Orientierung ϕ, ab:

$$\mathcal{B}(t) = \text{proj}(\mathcal{R}(M, t)) = \{\text{proj}(x) \mid x \in R(M, t)\}. \tag{2.17}$$

Diese, auch Worst-Case-Belegungen genannten, Projektionen werden nicht nur zu diskreten Zeitpunkten, sondern für konsekutive Zeitintervalle $\tau_l = [t_l, t_{l+1}]$ mit Zeithorizont t_h bestimmt. Zudem werden sie nicht exakt berechnet, sondern näherungsweise mit Modellen M_i überapproximiert: $\forall t > 0 : \mathcal{R}(M, t) \subseteq \mathcal{R}(M_i, t)$. Werden schließlich m unterschiedliche Modelle miteinander kombiniert, nähert sich die Schnittmenge ihrer Überapproximationen der realen Belegung des Modells M_0 an:

$$\forall t > 0 : \text{proj}(\mathcal{R}(M_0, t)) \subseteq \bigcap_{i=1}^{m} \text{proj}(\mathcal{R}(M_i, t)). \tag{2.18}$$

Somit kann die mögliche Belegung mit mehreren effizienten Überapproximationen hinreichend gut angenähert werden. Aus dieser Motivation heraus wurden in [Alt16] zwei Modelle vorgestellt: die sog. Beschleunigungs-basierte Belegung M_1 und die sog. Fahrstreifen-basierte Belegung M_2.

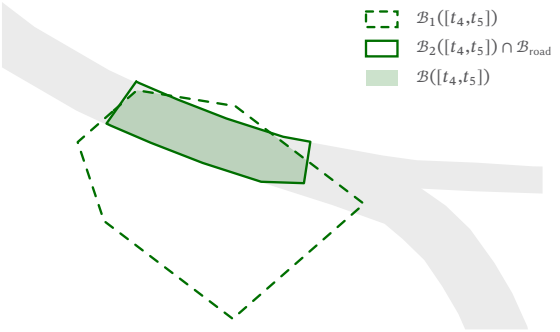
[1] resultierend aus der kontinuierlichen Fahrdynamik sowie diskreten Verkehrsregeln und Homotopieklassen

Beiden Modellen liegenden folgende Annahmen zugrunde:

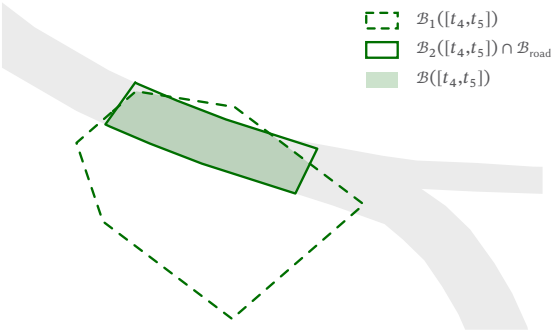
1 Die positive Längsbeschleunigung wird beendet, sobald eine festgelegte Geschwindigkeit $v_{\max}$ erreicht ist.
2 Ab einer Geschwindigkeit v_S ist die positive Längsbeschleunigung umgekehrt proportional zur Geschwindigkeit.[1]
3 In einem Fahrstreifen ist Rückwärtsfahren nicht zugelassen.
4 Die absolute Beschleunigung ist begrenzt durch $a_{\max}$.
5 Es ist untersagt, die Fahrstreifen, Geh-, Rad- und Überwege zu verlassen. Das Kreuzen von Fahrstreifen ist erlaubt, solange es nicht durch Fahrbahnmarkierungen oder Verkehrsregeln verboten ist.

M_1 prädiziert die Belegungen unter Verwendung eines Punktmassemodells und des Kamm'schen Kreises, womit die Annahmen 3 und 4 berücksichtigt werden. M_2 wiederum addressiert explizit das Längsverhalten innerhalb der Fahrstreifengeometrien, womit die Annahmen 1, 2 und 5 die Längskomponente von Annahme 4 einbezogen werden. Abbildung 2.13 veranschaulicht die resultierende Worst-Case-Belegung in einer Beispielsituation.

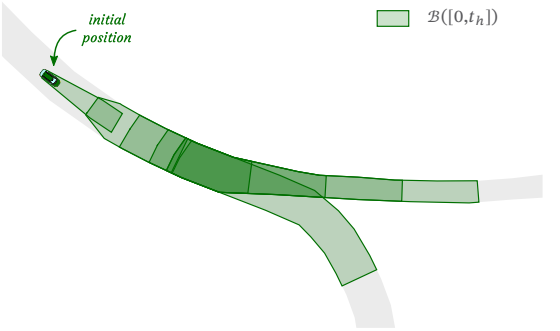
[1] Dies soll eine begrenzte Motorleistung modellieren.



(a) Belegung für $t \in [t_4, t_5]$ im Falle des Linksabbiegens.



(b) Belegung für $t \in [t_4, t_5]$ im Falle des Rechtsabbiegens.



(c) Resultierende Gesamtbelegung für den ganzen Prädiktionshorizont.

Abbildung 2.13: Worst-Case-Prädiktionen möglicher Belegungen des beobachteten Fahrzeugs.

3 Verhaltensarbitration für automatisierte Fahrzeuge

Dieses sowie das nächste Kapitel beschreiben den Kern der vorliegenden Arbeit: ein Verfahren zur sicheren und fehlertoleranten Verhaltensentscheidung für automatisierte Fahrzeuge. Hierzu wird die Methode der Verhaltensarbitration (Abschnitt 2.2.4) in diesem Kapitel auf das automatisierte Fahren übertragen. In Kapitel 4 wird es um ein Sicherheitskonzept auf Basis der Erreichbaren-Mengen-Analyse (Abschnitt 2.5.2) sowie Maßnahmen zum Entwurf fehlertoleranter Systeme (Abschnitt 2.4) erweitert. Somit garantiert das Verfahren nicht nur, dass ausschließlich aktuell sinnvolle und physikalisch realisierbare Aktionen gewählt werden können, sondern auch, dass das System dabei stets – unter den gegebenen Annahmen – in einem sicheren Zustand bleibt.

Zu Beginn wird in Abschnitt 3.1 das Umweltmodell beschrieben, auf dessen Grundlage die Verhaltensbausteine in Abschnitt 3.2 zum einen ihre Anwendbarkeit überprüfen und zum anderen die eigentliche Verhaltensplanung realisieren. Abschnitt 3.3 erläutert wie die Arbitratoren aus Abschnitt 2.2.4 nun zusätzlich zu den Start- und Fortsetzungs-Bedingungen die Realisierbarkeit und Sicherheit der Verhaltensoptionen prüfen.

3.1 Umweltmodell

Ein Umweltmodell beschreibt im Allgemeinen die aktuelle Situation, wobei diese aus einer interpretierten „Momentaufnahme des [statischen und dynamischen] Umfelds“ sowie den „relevanten Zielen und Werten“ besteht

[Ulb15]. Somit vereint das Umweltmodell die für die Verhaltensplanung relevanten Eingangsdaten und Parameter.

Im Folgenden wird das in dieser Arbeit verwendete Umweltmodell für automatisiertes Fahren vorgestellt. Dabei ist hervorzuheben, dass ein weitgehend allgemeines und übertragbares Modell beschrieben wird, allerdings auch andere, ggf. Sensor-nahe Repräsentationen, denkbar wären, um bspw. noch vielfältigere Verhalten zu ermöglichen.

Den statischen Teil der Umfeldrepräsentation bildet insbesondere eine hochgenaue Planungskarte [Pog18] ab, die die Fahrstreifengeometrien samt Topologie, Fuß- und Radwege, Fußgängerüberwege, Verkehrszeichen, Lichtsignalanlagen, befahrbare Flächen, Parkflächen und Verkehrsregeln umfasst. Dabei werden Fahrstreifen in atomare Segmente, sog. *Lanelets*, unterteilt, innerhalb derer einheitliche Verkehrsregeln gelten. Darauf aufbauend gibt schließlich der Fahrstreifengraph Auskunft darüber, in welcher Relation die Lanelets zueinander stehen und welche Verkehrsregeln dabei gelten. Ein externes Routenplanungsmodul bestimmt außerdem die vorgesehene Route innerhalb dieser Lanelet Karte und stellt diese in Form von Zwischenzielen bereit. Somit ist die Route hinreichend festgelegt, ohne die Wahl der Fahrstreifen vorwegzunehmen.

Zu den dynamischen Elementen des Umweltmodells zählt zunächst eine Eigenbewegungsschätzung und Lokalisierung. Sie bestimmt die Pose (Position und Orientierung) und Geschwindigkeit des sogenannten Ego-Fahrzeugs beispielsweise durch Kalman-Filterung einer bildbasierten Lokalisierung [Son17], der geschätzten GPS-Position und der Rad-Odometrie. Um ein sicheres Verhalten im Straßenverkehr zu ermöglichen ist zudem eine genaue und robuste Erkennung und -verfolgung anderer Verkehrsteilnehmer und Hindernisse unumgänglich. Diese beruht in der Regel auf Kamera- [Fra13, Sal20], Radar- [Dic15] und/oder LIDAR-Sensorik [Ste20, Ric19] und liefert Pose, Dimension, Geschwindigkeit sowie Semantik statischer und dynamischer Objekte. Die semantischen Klassen umfassen beispielsweise KFZs, wie Personenkraftwagen (PKWs) und Lastkraftwagen (LKWs), Fahrzeuge des öffentlichen Personennahverkehrs, wie Busse und Straßenbahnen, VRUs, wie Motorradfahrer, Fahrradfahrer, Elektrokleinstfahrzeuge (EKFs)

und Fußgänger, aber auch Einsatzfahrzeuge der Polizei und Rettungskräfte. Das Prädiktionsmodul liefert darauf aufbauend eine Vorhersage über den wahrscheinlichsten Verlauf der Objektbewegungen [Pet13] sowie die mittels Erreichbarer-Mengen-Analyse bestimmten Worst-Case-Belegungen (Abschnitt 2.5.2). Außerdem werden die Wahrnehmungsgrenzen durch zwei-dimensionale Polygone angegeben, um in der Verhaltensplanung auch Verdeckungen und begrenzte Sensorreichweite angemessen berücksichtigen zu können.

Schließlich wird die zuletzt geplante Soll-Trajektorie sowie die Fail-Safe-Trajektorie im Umweltmodell hinterlegt. Erstere stellt dabei sicher, dass die nächste Soll-Trajektorie ruckfrei und somit komfortabel geplant werden kann (vgl. [Zie14a, Kapitel III-F]). Die Fail-Safe-Trajektorie wird hingegen im Rahmen des in Abschnitt 4.2 eingeführten Sicherheitskonzepts für die Rückfallebene vorgehalten. Abbildung 3.1 veranschaulicht einige Elemente des Umweltmodells an einer Beispielsituation.

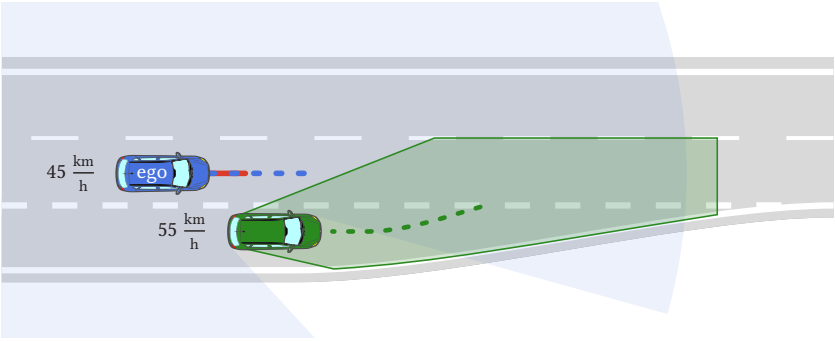


Abbildung 3.1: Beispielhafte Elemente eines Umweltmodells: Das Ego-Fahrzeug bewegt sich mit $45\ \frac{\text{km}}{\text{h}}$, die zuletzt geplante Soll- und Fail-Safe-Trajektorie sind blau gestrichelt bzw. rot gekennzeichnet und die Sensorreichweite blau hinterlegt. Ein weiteres Fahrzeug (grün) wurde mit $55\ \frac{\text{km}}{\text{h}}$ beobachtet, die prädizierte Trajektorie ist als gestrichelte Linie angedeutet und die prädizierte Worst-Case-Belegung grün unterlegt.

3.2 Verhaltensbausteine

In diesem Abschnitt sollen die in Abschnitt 2.2.4 vorgestellten Verhaltensbausteine in den Kontext des automatisierten Fahrens übertragen werden. Dies betrifft insbesondere welchen Abstraktionsgrad ein Verhaltensbaustein adressiert, die daraus resultierende Repräsentation der Stellgrößen, eine geeignete Spezifikation der Start- und Fortsetzungs-Bedingungen und schließlich wie in einem Verhaltensbaustein dessen Stellgrößen bestimmt werden können.

Bevor diese Details ausgeführt werden, seien zur besseren Übersicht nochmals die Schnittstellen eines Verhaltensbausteins zusammengefasst:

Start-Bedingung Die Start-Bedingung `Start` gibt an, ob ein Verhaltensbaustein in der gegebenen Situation aufgerufen werden kann.

Fortsetzungs-Bedingung Die Fortsetzungs-Bedingung `Fortsetzung` gibt an, ob ein bereits aktiver Verhaltensbaustein fortgesetzt werden kann.

Stellfunktion Die Stellfunktion `Aktion` führt die Verhaltensplanung des Verhaltensbausteins aus und gibt entsprechende Stellgrößen zurück.

3.2.1 Abstraktionsgrad

Wie in Abschnitt 2.1 diskutiert, lässt sich die Fahraufgabe in drei hierarchische Ebenen aufteilen: eine strategische, taktische und operative Ebene. Die in dieser Arbeit entworfene Verhaltensentscheidung ist dafür ausgelegt mindestens die taktische Ebene, also die Verhaltensentscheidung zu realisieren. Sie kann zwar auch für strategische Entscheidungen eingesetzt werden, allerdings werden diese in der Regel bereits offline im Voraus oder im Betrieb von den Nutzern oder externen Diensten festgelegt.

Ein Verhaltensbaustein bildet in dieser Arbeit also taktische Fahrmanöver ab, die sich über einen Zeitraum von etwa 5 bis 20 Sekunden erstrecken. Dies schließt beispielsweise Fahrstreifenwechsel, Einpark- oder auch Haltemanöver ein.

3.2.2 Manöverrepräsentation

Für die Repräsentation des von einem Verhaltensbaustein intendierten Verhaltens, ist entscheidend, ob die Verhaltensarbitration ausschließlich die Manöver- oder auch eine Bewegungsplanung übernehmen soll. Einerseits ist eine abstrakte Manöverrepräsentation, die im Wesentlichen die relevanten Fahrstreifenbegrenzungen, prädizierte Objekte samt gewählter Homotopieklasse sowie Verkehrsregeln umfasst, sehr gut für die Nebenbedingungen einer nachgelagerten Trajektorienplanung geeignet [Orz20]. Andererseits können nicht alle Situationen mit einer einheitlichen abstrakten Repräsentation abgedeckt werden. Beispielsweise haben Parkflächen nicht notwendigerweise wohldefinierte Fahrstreifen, deren Begrenzungen den Fahrkorridor festlegen könnten.

Zudem werden die Arbitratoren im Rahmen des Sicherheitskonzepts in Abschnitt 4.2 bereits konkrete Trajektorien für die Erreichbare-Mengen-Analyse benötigen, um entscheiden zu können, ob ein Verhalten nicht nur anwendbar, sondern auch sicher ist. Aus diesen Gründen wird in dieser Arbeit das geplante Manöver in Form einer *Soll-Trajektorie* $\boldsymbol{x}(t)$ ausgegeben. Sie bildet den Hauptbestandteil der Stellgrößen, die von der Funktion `Aktion` bestimmt werden.

Zusätzlich definiert die *Fail-Safe-Trajektorie* $\check{\boldsymbol{x}}(t)$, wie in Abschnitt 2.5.2 und 4.2 diskutiert, eine ausfallsichere Alternativ-Trajektorie für den Fall, dass im nächsten Planungsschritt keine neue beweisbar sichere Trajektorie gefunden werden kann. Die Fail-Safe-Trajektorie garantiert also ein kollisionsfreies Verhalten bzgl. der Worst-Case-Prädiktionen der Erreichbaren-Mengen-Analyse.

Weiterhin werden Verhaltensbausteine mit der Umwelt oder auch den Passagieren kommunizieren wollen. Dazu zählen Ausgaben und Informationen für die nach innen wie nach außen gerichteten HMIs sowie zur V2X-Kommunikation.

Eine *HMI* kann Insassen beispielsweise über den aktuellen Zustand der Automatisierung oder dynamische Größen wie die aktuelle Geschwindigkeit aufklären. Sie kann aber auch zur Interaktion mit den Insassen herangezogen werden, um bei Bedarf die Autonomiegrenzen des Systems zu überwinden.

Beispielsweise könnte darüber eine manuelle Freigabe von schwierigen Vorfahrtsituationen angefragt werden. HMIs werden außerdem auch eingesetzt, um die Interaktion mit anderen Verkehrsteilnehmern zu verbessern — sei es mit konventionellen Fahrtrichtungsanzeigern oder innovativen Ansätzen zur Kommunikation mit kreuzenden Fußgängern [Ras20].

V2V und *V2I* können verwendet werden, um mit anderen Verkehrsteilnehmern auf Informations- oder gar Manöverebene zu kooperieren [Bur17]. Bei der informationsbasierten Kooperation werden bspw. Sensor- oder Planungsdaten weitergegeben, um den anderen Verkehrsteilnehmern eine bessere Wahrnehmung und Prädiktion zu ermöglichen. Bei manöverbasierter Kooperation werden zudem Manöveroptionen samt assoziierten Kosten oder Nutzen kommuniziert, um in gemeinsamer Absprache interaktive Fahrmanöver zu koordinieren. In Anwendungen, die hohe Automatisierungsgrade anstreben oder große Flotten einsetzen, wird eine sog. Flottenverwaltung mittels *V2F* bspw. über Fehlerfälle informiert, um entscheiden zu können, ob ein Fahrzeug zur Wartung geordert werden soll [Mer18].

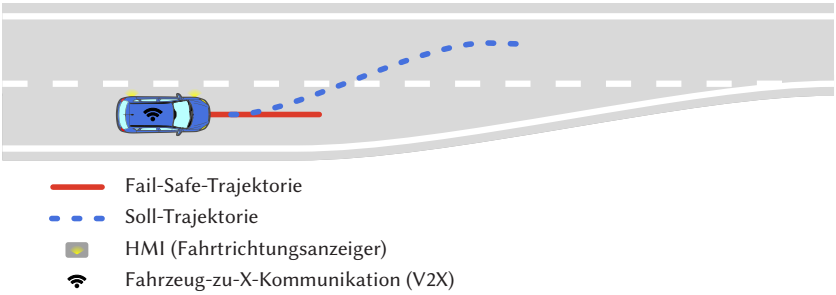


Abbildung 3.2: In dieser Arbeit gewählte Manöverrepräsentation: Soll- und Fail-Safe-Trajektorie, Fahrtrichtungsanzeiger als Beispiel einer HMI und V2X Kommunikation.

Die Manöverrepräsentation setzt sich schließlich aus

- der Soll-Trajektorie $\boldsymbol{x}(t)$,
- einer Fail-Safe-Trajektorie $\check{\boldsymbol{x}}(t)$,
- HMI-Ausgaben m_{HMI} und

- V2X-Nachrichten m_{V2X}

zusammen. Abbildung 3.2 stellt diese in einem beispielhaften Szenario dar.

3.2.3 Start- und Fortsetzungs-Bedingungen im Kontext des automatisierten Fahrens

Die Start- und Fortsetzungs-Bedingungen der Verhaltensbausteine leiten sich im Kontext des automatisierten Fahrens aus den in Abschnitt 2.5.1 eingeführten ODDs der adressierten Teilverhalten ab. Dabei beziehen sich die ODDs auf das Verhalten der gesamten Verarbeitungskette eines Fahrerassistenzsystems (oder einer seiner Teilfunktionen, nicht aber Teilmodule), sodass alle Verarbeitungsschritte – von der Wahrnehmung bis zur Regelung – innerhalb der spezifizierten ODDs voll funktionsfähig sein müssen. Es kann daher angebracht sein, in den Start- und Fortsetzungs-Bedingungen eines Verhaltensbausteins neben den ODDs auch die Qualität der Eingangsdaten, beispielsweise in Form von Unsicherheiten, Sensorabdeckung oder -reichweite, zu prüfen. Dies ist insbesondere bei Forschungs- und Erprobungsfahrzeugen deren Wahrnehmung noch nicht die ODDs erfüllt, oder als zusätzliche Vorsichtsmaßnahme in Serienfahrzeugen, ratsam. Beispiel 3.1 zeigt exemplarisch die Start-Bedingungen eines Verhaltensbausteins zum Einordnen im Reißverschlussverfahren.

Beispiel 3.1: Einordnen im Reißverschlussverfahren

Ein Verhaltensbaustein eines Erprobungsfahrzeugs zum Einordnen im Reißverschlussverfahren kann beispielsweise eingeschränkt werden auf

- Engstellen auf dem Testfeld Autonomes Fahren Baden-Württemberg [Fuc18],
  - an welchen zwei Fahrstreifen gleicher Fahrtrichtung
  - mit zulässigen Höchstgeschwindigkeiten von bis zu 60 $\frac{\text{km}}{\text{h}}$
  - ohne Lichtsignalanlage zusammengeführt werden,
- zu jeder Tageszeit,
- bei sonnig trockenem Wetter bis einschließlich gemäßigtem Regen,
- solange der zweite Fahrstreifen mind. 65 m nach hinten sensorisch erfasst ist
- und sich unter den betroffenen Verkehrsteilnehmern keine VRUs befinden.

Schließlich muss beim Systementwurf darauf geachtet werden, dass sich die ODDs der Verhaltensbausteine überlappen, damit eine sichere Übergabe zwischen jeweils zwei Verhaltensoptionen gewährleistet werden kann.

3.2.4 Verhaltensgenerierung

Sind die Start- und Fortsetzungs-Bedingungen eines Verhaltensbausteins erfüllt und wird dieser daraufhin von seinem übergeordneten Arbitrator aufgerufen, erfolgt die eigentliche Verhaltensgenerierung (oder auch Bewegungsplanung) in der `Aktion`-Funktion. Diese schließt die Planung der Soll- und Fail-Safe-Trajektorien sowie ggf. die Bestimmung von HMI-Ausgaben oder V2X-Nachrichten ein.

Soll-Trajektorie

Die Soll-Trajektorie, im Terminus von [Alt16] auch *Langzeit-Referenz-Trajektorie* genannt, gibt an wie sich das Fahrzeug für die nächsten 5 bis 20 Sekunden bewegen soll. Sie wird im Rahmen der Verhaltensgenerierung bspw. mittels Graphensuche [Ban18], strukturierter Trajektorienscharen [Wer10], Nichtlinearer Optimierung [Zie14a], POMDP [Hub18b] oder lernender Verfahren wie Bestärkendem Lernen [Zen20] geplant. Dabei gibt es zum einen harte Anforderungen, die sie in jedem Falle erfüllen muss. Zum anderen werden auch weiche Anforderungen formuliert, die zwar im Regelfall erfüllt sein sollten, aber nicht zwingend notwendig sind.

Zu den harten Anforderungen an eine Soll-Trajektorie zählt insbesondere ihre Realisierbarkeit. Es muss also sichergestellt sein, dass die resultierenden Aktor-Stellgrößen wie Soll-Beschleunigung und Soll-Lenkwinkel beschränkt sind und die Trajektorie auch fahrdynamisch umsetzbar ist. Die Grenzen der Realisierbarkeit werden durch technische Leistungsgrenzen, z. B. den maximal möglichen Lenkeinschlag, sowie physikalische Grenzen, wie die maximal übertragbaren Seitenkräfte, festgelegt. Um diese Anforderung zuverlässig zu erfüllen, kann ein dynamisches Fahrzeugmodell in die Trajektorienplanung eingebunden werden.

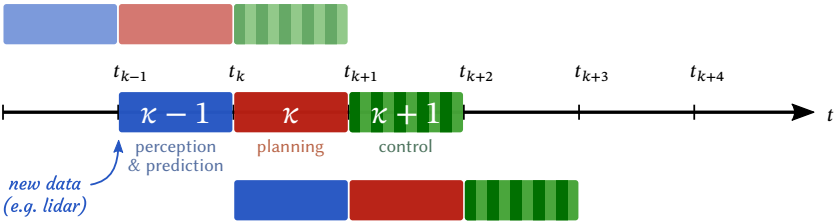


Abbildung 3.3: Intervalle der Verarbeitungskette: Zum Planungsintervall $\kappa = [t_k, t_{k+1})$ wird die im vorigen Intervall bestimmte Trajektorie $\boldsymbol{x}^{\kappa-1}(t)$ eingeregelt. Die Soll-Trajektorie $\boldsymbol{x}^{\kappa}(t)$ wird folglich frühestens zum Zeitpunkt t_{k+1} eingeregelt. Die Fail-Safe-Trajektorie $\check{\boldsymbol{x}}^{\kappa}(t)$, auf die im Planungsintervall $\kappa + 1$ gewechselt werden könnte, würde hingegen frühestens zum Zeitpunkt t_{k+2} eingeregelt.

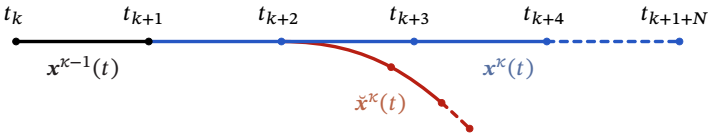


Abbildung 3.4: Die im Zeitintervall $\kappa = [t_k, t_{k+1})$ geplante Soll-Trajektorie $\boldsymbol{x}^{\kappa}(t)$ muss in diesem Intervall mit der vorherigen Trajektorie $\boldsymbol{x}^{\kappa-1}(t)$ übereinstimmen, um einen ruckarmen Übergang zu ermöglichen. Da die Fail-Safe-Trajektorie $\check{\boldsymbol{x}}^{\kappa}(t)$ frühestens zum Zeitpunkt t_{k+2} eingeregelt wird, muss sie für das Zeitintervall $[t_k, t_{k+2}]$ mit der Soll-Trajektorie $\boldsymbol{x}^{\kappa}(t)$ übereinstimmen.

Eine weitere harte Anforderung resultiert aus dem Laufzeitverhalten der Verarbeitungskette. Abbildung 3.3 veranschaulicht vereinfachend die Laufzeitintervalle der drei konsekutiven Verarbeitungsschritte *Wahrnehmung und Prädiktion*, *Verhaltensgenerierung* und *Regelung*. Bei Eintreffen neuer Messdaten zum Zeitpunkt t_{k-1} bestimmt die Wahrnehmung und Prädiktion das in Abschnitt 3.1 beschriebene Umweltmodell. Liegt dieses zum Zeitpunkt t_k vor, wird auf dessen Basis die Verhaltensgenerierung durchgeführt. Da diese allerdings voraussichtlich erst zum Zeitpunkt t_{k+1} abgeschlossen sein wird, muss davon ausgegangen werden, dass bis dahin die im vorigen Planungsschritt $\kappa-1$ geplante Trajektorie eingeregelt wird. Um folglich den Übergang von der zuvor zu der aktuell geplanten Trajektorie zu ermöglichen, muss die neue Trajektorie $\boldsymbol{x}^{\kappa}(t)$ im Intervall $[t_k, t_{k+1}]$ mit der vorherigen Trajektorie $\boldsymbol{x}^{\kappa-1}(t)$,

wie in Abb. 3.4 dargestellt, übereinstimmen[1]:

$$\boldsymbol{x}^{\kappa}(t) = \boldsymbol{x}^{\kappa-1}(t), \qquad \forall t \in [t_k, t_{k+1}] \qquad \text{bzw.} \tag{3.1a}$$

$$\boldsymbol{x}_l^{\kappa} = \boldsymbol{x}_{l+1}^{\kappa-1}, \qquad \forall l \in [0,1]. \tag{3.1b}$$

Wäre diese Anforderung nicht erfüllt, käme es am Übergang der Trajektorien zu einem Sprung und somit voraussichtlich auch zu einem ruckbehafteten Sprung in den Stellgrößen des Reglers. In einem solchen Falle wäre daher die Realisierbarkeit der Soll-Trajektorie nicht mehr gewährleistet.

Weiterhin sollte die Soll-Trajektorie kollisionsfrei bzgl. der prädizierten Objektbewegungen sein. Da sich andere Verkehrsteilnehmer allerdings aufgrund von Unsicherheiten in der Wahrnehmung und Prädiktion sowie der nichtdeterministischen menschlichen Natur durchaus anders verhalten können als vorhergesehen, muss die Soll-Trajektorie diese Zielsetzung nicht notwendigerweise erfüllen, solange die Fail-Safe-Trajektorie beweisbar kollisionsfrei ist.

Als zweite weiche Anforderung gilt die Einhaltung der Verkehrsregeln. Da allerdings Sondersituationen die Überschreitung einzelner Verkehrsregeln erfordern können, bspw. um den Verkehrsfluss aufrechtzuerhalten oder Rettungsfahrzeugen Platz zu schaffen, darf die Soll-Trajektorie die Verkehrsregeln gelegentlich brechen.

Fail-Safe-Trajektorie

Die Fail-Safe-Trajektorie stellt eine ausfallsichere Alternativ-Trajektorie dar, auf die die Verhaltensarbitration im Planungsintervall $\kappa + 1$ zurückgreifen kann, sollte dann keine der anwendbaren Verhaltensoptionen ein beweisbar sicheres Manöver zurückgeben. Prinzipiell können Fail-Safe-Trajektorien mit

[1] Hierbei wird vereinfachend, allerdings o. B. d. A. angenommen, dass die zeitliche Auflösung der Soll-Trajektorie mit den Laufzeitintervallen der Trajektorienplanung übereinstimmt. Eine bspw. höhere Auflösung der Soll-Trajektorie kann mit Substitution des Index l erreicht werden.

ähnlichen Methoden wie die Soll-Trajektorie geplant werden. Es wurde allerdings bereits auch ein spezialisiertes Verfahren basierend auf konvexer Optimierung vorgestellt [Pek18]. Außerdem ist eine kombinierte Planung der Soll- und Fail-Safe-Trajektorien in einem integrierten Planungsproblem denkbar.

Da die Fail-Safe-Trajektorie eine zuverlässige Rückfallebene bilden soll, unterliegt sie strengeren harten Anforderungen als die Soll-Trajektorie. Zunächst muss sie kollisionsfrei bzgl. der Worst-Case-Prädiktionen der Erreichbare-Mengen-Analyse sein. Sie darf also zu keinem Zeitpunkt eine Überschneidung mit den möglichen Belegungen durch andere Verkehrsteilnehmer aufweisen (Abschnitt 2.5.2). Diese harte Anforderung an die Fail-Safe-Trajektorie ist wesentlich für das Sicherheitskonzept in Abschnitt 4.2 und wird daher in Abschnitt 4.2.4 formalisiert.

Außerdem muss die Fail-Safe-Trajektorie analog zur Soll-Trajektorie technisch wie auch fahrdynamisch realisierbar sein. Da die im Planungsintervall κ bestimmte Fail-Safe-Trajektorie $\check{\boldsymbol{x}}^{\kappa}(t)$ allerdings, wie in Abb. 3.3 und 3.4 veranschaulicht, erst im Planungsintervall $\kappa + 1$ gewählt werden könnte, wird sie frühestens ab t_{k+2} eingeregelt. Daher muss $\check{\boldsymbol{x}}^{\kappa}(t)$ im Intervall $[t_k, t_{k+2}]$ der Soll-Trajektorie $\boldsymbol{x}^{\kappa}(t)$ gleichkommen, um ihre Realisierbarkeit zu gewährleisten:

$$\check{\boldsymbol{x}}^{\kappa}(t) = \boldsymbol{x}^{\kappa}(t), \qquad \forall t \in [t_k, t_{k+2}] \qquad \text{bzw.} \tag{3.2a}$$

$$\check{\boldsymbol{x}}_l^{\kappa} = \boldsymbol{x}_l^{\kappa}, \qquad \forall l \in [0,2]. \tag{3.2b}$$

HMI-Ausgaben

Die von einem Verhaltensbaustein veranlassten HMI-Ausgaben m_{HMI} sind in der Regel manöverabhängig und vorab definiert. Bei einem Fahrstreifenwechsel werden bspw. die entsprechenden Richtungsanzeiger aktiviert.

V2X-Nachrichten

Ähnlich verhält es sich mit den V2X-Nachrichten m_{V2X}, wobei diese bspw. die Intention des geplanten Manövers, die aktuell geplante Soll-Trajektorie $\boldsymbol{x}^{\kappa}(t)$ oder auch weitere Informationen beinhalten können.

Stellgrößentupel

Schließlich gibt die Funktion `Aktion` das geplante Manöver, wie in Abschnitt 3.2.2 festgelegt, als Tupel

$$\boldsymbol{u}^{\kappa} = (\boldsymbol{x}^{\kappa}, \check{\boldsymbol{x}}^{\kappa}, m^{\kappa}_{\text{HMI}}, m^{\kappa}_{\text{V2X}}) \tag{3.3}$$

an den aufrufenden Arbitrator zurück.

3.3 Arbitratoren

Die Arbitratoren sind in ihrer Theorie sowie Implementierung grundsätzlich generisch und anwendungsunabhängig. Es kann allerdings in sicherheitskritischen Anwendungen, wie dem automatisierten Fahren, vorteilhaft sein die Arbitration um einen Verifikationsschritt zu erweitern. Prinzipiell wäre eine Verifikation auch in den Start- und Fortsetzungs-Bedingungen der Verhaltensbausteine möglich, jedoch wäre hierfür bereits die ggf. rechenintensive Berechnung der Stellgrößen notwendig. Außerdem trägt eine Verifikation außerhalb der Verhaltensbausteine zur Entkopplung bei und garantiert, dass die gleiche Verifikationsmethode für alle Verhaltensbausteine eingesetzt wird.

Daher wird der *Verifikationsschritt* in die Arbitration integriert, indem die Stellgröße der `Aktion`-Funktion einer Verifikationsfunktion $\mathcal{V}$ unterzogen wird, bevor der Arbitrator entscheidet, ob er diese Option tatsächlich auswählen und diese Stellgröße zurückgeben soll. Scheitert nämlich die Verifikation, prüft der Arbitrator seine Alternativoptionen oder greift auf die Rückfallebene zurück. Abschnitt 4.2 erläutert, wie dadurch die Gültigkeit und Sicherheit der von einem Arbitrator zurückgegebenen Stellgröße gewährleistet wird.

Algorithmus 2 : Kosten-Arbitrator

```
1 function BesteAnwendbareAktion(Situation s)
2     Filtere anwendbare Optionen A ⊂ O
3     Sortiere anwendbare Optionen nach Kosten ihrer Aktion:
        A* = ⟨a_0, a_1, ...⟩ | cost(Aktion_{a_i}(s)) < cost(Aktion_{a_{i+1}}(s)) ∀a_i ∈ A
4     Wähle günstigste Intention θ = a_0 aus
5     Bestimme u = Aktion_θ(s)
6     return u
7 end
```

3.3.1 Kosten-Arbitrator

Eine weitere Anforderung, die sich aus den ersten Untersuchungen zur Verhaltensarbitration für automatisierte Fahrzeuge ergeben hatte, ist die Notwendigkeit in ausgewählten Arbitrationsebenen statt nach einer festgelegten Priorisierung oder Sequenz eine Verhaltensoption dynamisch nach dem zu erwartenden Nutzen zu wählen. Hierfür wurde in [Orz20] ein sog. *Kosten-Arbitrator* vorgestellt, der an dieser Stelle formalisiert werden soll.

Die Start- und Fortsetzungs-Bedingung des Kosten-Arbitrators gleicht dem des Prioritäts-Arbitrators:

$$\texttt{Start}_{\text{cost}}(\boldsymbol{s}_k) = \bigvee_{o \in \mathcal{O}} \texttt{Start}_o(\boldsymbol{s}_k) \tag{3.4}$$

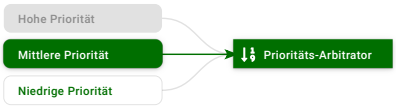
$$\texttt{Fortsetzung}_{\text{cost}}(\boldsymbol{s}_k) = \bigvee_{o \in \mathcal{O}} \texttt{Start}_o(\boldsymbol{s}_k) \vee \texttt{Fortsetzung}_{\theta_{k-1}}(\boldsymbol{s}_k) \tag{3.5}$$

In der `Aktion`-Funktion, die in Algorithmus 2 aufgeführt ist, filtert der Kosten-Arbitrator zunächst seine Optionen $\mathcal{O}$ nach ihrer Anwendbarkeit. Anschließend bestimmt er für jede der anwendbaren Optionen ihre Aktion, schätzt die zugehörigen Kosten und sortiert diese Optionen nach ihren Kosten. Schließlich wählt er aus dieser sortierten Liste anwendbarer Optionen A^* die günstigste als Intention und führt diese aus.

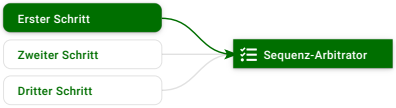
3.3.2 Übersicht

Neben der Definition von neuen Arbitratoren ermöglicht die in dieser Arbeit verwendete Implementierung zudem weitere Feineinstellungen der Auswahllogik. Beispielsweise kann für jede Option definiert werden, ob sie trotz wahrer Fortsetzungs-Bedingung unterbrechbar ist. Auf diese Weise kann u. a. der Abschluss-Arbitrator durch den Prioritäts-Arbitrator ersetzt werden.

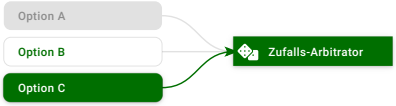
Fasst man schließlich die Arbitratoren aus Abschnitt 2.2.4 mit dem Kosten-Arbitrator zusammen, stehen beim Systementwurf die in Abb. 3.5 dargestellten Arbitrationsschemata zur Verfügung: der Prioritäts-, Sequenz-, Zufalls- und Kosten-Arbitrator.



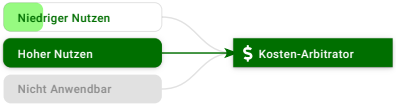
(a) Da der Verhaltensbaustein *Hohe Priorität* aktuell nicht anwendbar ist, wählt der Prioritäts-Arbitrator die Option *Mittlere Priorität*.



(b) Der Sequenz-Arbitrator ruft seine Optionen der Reihe nach auf.



(c) Der Zufalls-Arbitrator wählt zufällig eine seiner anwendbaren Optionen *Option B* und *C*.



(d) Der Kosten-Arbitrator wählt die anwendbare Option mit den geringsten zu erwartenden Kosten bzw. dem höchsten Nutzen.

Abbildung 3.5: Die in dieser Arbeit verwendeten Arbitrationsschemata.

4 Sichere und fehlertolerante Verhaltensarbitration

Das allgemeine Ziel des Sicherheitskonzepts dieser Arbeit ist es, in Anlehnung an [Rei10], kritische Fahrzeugzustände und Verkehrssituationen zu vermeiden und zugleich die daraus resultierenden Funktions- oder Leistungseinschränkungen zu minimieren. Im Rahmen dieser Arbeit wird eine Auswahl relevanter Verkehrsrisiken sowie möglicher spezifikations-, implementierungs- und störungsbedingter Fehlzustände in der Verhaltensgenerierung berücksichtigt. Fehlzustände in der eingesetzten Hardware (u. a. Sensorik, Aktorik und Rechnersysteme) oder Middleware (z. B. Robot Operating System (ROS)) sind hingegen nicht Inhalt dieser Arbeit, da hierzu bereits umfangreiche Literatur existiert [Ech90, Lap95, Dub13, Mon99, Kor20].

Dieses Kapitel verwendet, insbesondere beim Sicherheitskonzept in Abschnitt 4.2, eine möglichst formale und auf andere Anwendungen übertragbare Darstellung, bevor Kapitel 5 die konkrete Umsetzung im Kontext des automatisierten Fahrens darlegt.

Abschnitt 4.1 definiert zunächst die Sicherheitsziele dieser Arbeit. In Abschnitt 4.2 wird das dieser Arbeit zugrunde liegende Sicherheitskonzept bezüglich Ausfallsicherheit, Echtzeitfähigkeit und Verkehrssicherheit entworfen, um damit die genannten Sicherheitsziele zu gewährleisten. Schließlich fasst Abschnitt 4.3 die Kapitel 3 und 4 in einer Systemübersicht zusammen.

4.1 Sicherheitsziele

Für die Methode der Verhaltensarbitration lassen sich daraus konkrete Sicherheitsziele ableiten, die die Fehlertoleranz (Abschnitt 2.4) sowie die Verkehrssicherheit (Abschnitt 2.5) adressieren und im Folgenden für die Anwendung des automatisierten Fahrens festgelegt werden. An dieser Stelle wird zur besseren Lesbarkeit „die Verhaltensgenerierung mittels Arbitrationsverfahren“ mit „das System“ abgekürzt.

S1: Das System deckt notwendige Verhaltenskompetenzen ab

Zunächst muss die Verhaltensgenerierung ausreichend Verhaltenskompetenzen abdecken, um die zu erwartenden Situationen innerhalb der ODDs bewältigen zu können. Sollte dennoch eine nicht unterstützte Situation auftreten, muss die Verhaltensgenerierung dies erkennen und ein MRM ausführen, um das Fahrzeug in einen MRC zu überführen. Es sei darauf hingewiesen, dass MRMs zwar das vom betroffenen Fahrzeug ausgehende Risiko minimieren, allerdings dennoch eine höhere Verkehrsgefährdung darstellen können als ein spezialisiertes Fahrmanöver (bspw. wenn das Fahrzeug vor einer Baustelle zum Stehen kommt, statt sie gekonnt zu passieren). Daher müssen die ODDs möglichst genau bestimmt werden und zugleich möglichst vollständig von den Verhaltenskompetenzen abgedeckt werden, um solche nicht unterstützten Situationen innerhalb der ODDs zu vermeiden.

S2: Das System gewährleistet ein MRM bei Verlassen der ODDs

Sind die ODDs sorgfältig spezifiziert und umfänglich über Verhaltenskompetenzen abgedeckt, können sie z. B. durch einen Wetterumschwung oder eine sich unerwartet veränderte Szenerie dennoch verlassen werden. In einem solchen Fall muss die Verhaltensgenerierung diesen Umstand erkennen und das

Fahrzeug wie in Abschnitt 2.5 beschrieben in einen sicheren Zustand überführen. Je nach Situation können hierfür verschiedene MRMs infrage kommen.

S3: Das System ist robust gegen Ausfälle der Verhaltensbausteine

Während Arbitratoren einer vergleichsweise simplen Funktionslogik folgen und somit ihre robuste und zuverlässige Ausführung durch Komponententests oder gar einen formalen Beweis sichergestellt werden kann, decken Verhaltensbausteine hingegen komplexe Aufgaben ab und werden ggf. von unterschiedlichen Entwicklern in variierender Qualität implementiert. Daher sind seltene Ausfälle der Verhaltensbausteine in Form von Softwareabstürzen, insbesondere im frühen Entwicklungsstadium, nicht auszuschließen und müssen somit abgefangen werden.

S4: Das System vermeidet unzureichende Planungsfrequenz

Damit die von den Verhaltensbausteinen geplanten Trajektorien auch tatsächlich realisiert werden können, bspw. ein Haltemanöver zum Halt an der vorgesehenen Stelle führt, muss die gesamte Verhaltensgenerierung eines Serienfahrzeugs harte Echtzeitbedingungen erfüllen [Wör05]. Die Soll-Trajektorie des Planungsintervalls $\kappa = [t_k, t_{k+1})$ muss folglich spätestens zum Zeitpunkt t_{k+1} an die Regelung weitergereicht werden. In Versuchsfahrzeugen mit geschultem Sicherheitsfahrer kann diese Anforderung zu weichen Echtzeitbedingungen entschärft werden.

Gilt für die Verhaltensarbitration eine harte Echtzeitanforderung von bspw. 200 ms Laufzeit bzw. Periodendauer, müssen die Verhaltensbausteine eine strengere Echtzeitbedingung von entsprechenden 180 ms Laufzeit erfüllen. Somit wird den Arbitratoren Zeit für eine Erwägung aller infrage kommender

Manöveroptionen und der Verifikation ihrer Soll- und Fail-Safe-Trajektorien gegeben.

S5: Das System verhindert ungültige Stellgrößen

Der Einsatz Nichtlinearer Optimierung oder Methoden des Maschinellen Lernens innerhalb der Verhaltensbausteine birgt die Gefahr Trajektorien zu generieren, die kinematisch oder dynamisch nicht realisierbar sind. Daher muss die Verhaltensarbitration in der Lage sein, Trajektorien, die zu ungültigen Stellgrößen führen würden, zu erkennen und abzufangen.

S6: Das System vermeidet riskante Stellgrößen

Weiterhin muss sichergestellt werden, dass riskante Trajektorien, also solche, die zu kritischen Fahrsituationen führen können, wenn möglich verhindert werden. Es reicht allerdings nachzuweisen, dass die geplante Soll-Trajektorie noch im nächsten Planungsintervall einen Wechsel zur Fail-Safe-Trajektorie ermöglicht und letztere nachweislich kollisionsfrei ist. Hierfür können bspw. die prädizierten Worst-Case-Belegungen aus Abschnitt 2.5.2 ausgewertet werden.

4.2 Sicherheitskonzept

Das in diesem Abschnitt beschriebene Sicherheitskonzept baut auf den in Abschnitt 2.4 und Abschnitt 2.5 vorgestellten Konzepten zur Fehlertoleranz und Verkehrssicherheit sowie auf Maßnahmen zur Echtzeitfähigkeit auf. Zu den Methoden zur Verkehrssicherheit zählt der Systementwurf und die Verifikation, womit die Sicherheitsziele S1, S2 und S6 adressiert werden. Die Einhaltung der Echtzeitfähigkeit gewährleistet zudem das Sicherheitsziel S4. Die Verfahren zur Fehlertoleranz – insbesondere die Modularität des Arbitrationsgraphen und die daraus resultierende Entkopplung der Verhaltensbausteine sowie die Redundanz und Diversität von Verhaltensbausteinen – tragen

schließlich zur Ausfallsicherheit bei und realisieren somit die Sicherheitsziele S3 und S5.

Im Sinne fehlertoleranter Systeme sind der Systementwurf und die Sicherstellung der Echtzeitfähigkeit Maßnahmen zur Fehlervermeidung, während die Verifikation der Stellgrößen und die Methoden zur Ausfallsicherheit Maßnahmen zur Fehlererkennung und -behandlung sind.

4.2.1 Fehlervermeidung beim Systementwurf

Abbildung 4.1: Arbitrationsgraph mit drei Verhaltensbausteinen als Rückfallebene. *Continue Last Maneuver* setzt das zuletzt geplante Manöver fort, solange es noch ausführbar und nicht zu alt ist. *Fail Safe Fallback* führt die zuletzt geplante Fail-Safe-Trajektorie aus. Sollte sogar diese wegen unerwarteter Ereignisse nicht mehr sicher sein, führt *Emergency Stop* ein Nothaltemanöver aus.

Ein strukturierter Systementwurf trägt zur Fehlervermeidung bzgl. Verkehrssicherheit bei und bildet somit die Grundlage, um die genannten Sicherheitsziele in der Verhaltensgenerierung zu erreichen. Zunächst sind die ODDs an das Gesamtsystem zu definieren. Es ist also zu klären, wo und unter welchen

Bedingungen das automatisierte Fahrzeug betrieben werden soll. Anschließend werden in einem iterativen Prozess hieraus notwendige Verhaltenskompetenzen abgeleitet. Diese können initial auf Erfahrungswerten sowie Katalogen wie die der NHTSA und Waymo [Way20a] beruhen, müssen im weiteren Verlauf allerdings durch systematische Analyse von Realdaten laufend ergänzt werden.

Im nächsten Schritt werden Verhaltensbausteine entworfen, die jeweils eine oder mehrere Verhaltenskompetenzen abdecken. Dabei werden die jeweiligen ODDs in die Start- und Fortsetzungs-Bedingungen der Verhaltensbausteine integriert. Wurden alle Verhaltenskompetenzen berücksichtigt, liegt der Schluss nahe, dass die ODDs hinreichend adressiert wurden. Wenn sich jedoch die Betriebsbedingungen der Verhaltensbausteine nicht nennenswert überlappen, kann nur in seltenen Fällen zwischen Verhaltensbausteinen gewechselt und somit nicht die vollständigen ODDs bedient werden. Daher ist es sinnvoll detaillierte ODDs je Verhaltensbaustein festzulegen, um deren Überlappungen untersuchen und sicherstellen zu können.

Aus den ODDs ergibt sich zudem der Definitionsbereich, der durch Komponententests abgesichert werden muss. Hierbei erleichtert die Modularität, insbesondere die Aufspaltung in einzelne Verhaltensbausteine, deren Validierung: Die Verhaltensbausteine können aus dem Baum herausgenommen und in separaten kontrollierten Testumgebungen überprüft werden.

Außerdem sind die ODDs der Verhaltensbausteine neben den Verhaltenskompetenzen ein weiterer Ausgangspunkt bei der Erstellung eines Szenarienkatalogs zur Validierung der Verhaltensbausteine. Dabei ist darauf zu achten, das ganze Spektrum der jeweiligen ODD auszuschöpfen, um eine möglichst vollständige Abdeckung zu erreichen.

Eine lückenlose Abdeckung der ODDs und der dafür notwendigen Verhaltenskompetenzen wird allerdings aufgrund der Fülle und Komplexität insbesondere sehr seltener Ereignisse nicht möglich sein. Daher wird das aus bspw. unvorhergesehenen Situationen resultierende Restrisiko über eine Rückfallebene samt Verifikation der Fail-Safe-Trajektorien abgefangen. Damit in jedem Falle ein MRM verfügbar ist, muss die Wurzel des Arbitrationsgraphen

mindestens ein MRM bereithalten, das immer anwendbar ist (bspw. ein Nothaltemanöver).

Abb. 4.1 veranschaulicht die in dieser Arbeit eingeführte Rückfallebene am minimalen Arbitrationsgraphen für das automatisierte Fahren. Mit einem zwischengeschalteten Kosten-Arbitrator wurden Verhaltensbausteine zur Folgefahrt (*Follow Lane*), für Fahrstreifenwechsel (*Change Lane Left* und *Change Lane Right*) und zum Einparken (*Park Near Goal*) im Automated Driving Prioritäts-Arbitrator angeordnet. Bezüglich konkreter Funktionsweise und Implementierung dieser Verhaltensbausteine sei auf Kapitel 5 verwiesen. Zusätzlich ist an dieser Stelle die erste Rückfalloption *Continue Last Maneuver* eingebettet. Der Automated Driving Arbitrator bildet zugleich die Wurzel des Graphen und hält die Spezialverhalten *Park Near Goal* und *Slowly Pass Zebra* sowie die Rückfalloptionen *Fail Safe Fallback* und *Emergency Stop* mit geringerer Priorität bereit. Sollte wie angedeutet keine der priorisierten Optionen anwendbar sein, weil die ODDs verlassen wurden, wählt Automated Driving eine Option der Rückfallebene. Das Verhalten *Fail Safe Fallback* greift über das Umweltmodell auf die zuletzt geplante – ggf. noch komfortable – Fail-Safe-Trajektorie zurück und gibt diese aus. Wurden seit dem letzten Planungsintervall keine Annahmen der Verifikation verletzt, ist diese Fail-Safe-Trajektorie weiterhin nachweislich kollisionsfrei. Stellt sich jedoch heraus, dass diese nicht mehr beweisbar sicher ist, weil mind. eine der Annahmen verletzt wurde, kann der Automated Driving Arbitrator auf das Nothaltemanöver *Emergency Stop* zurückgreifen.

Durch einen solchen Systementwurf werden die ersten beiden Sicherheitsziele S1 und S2, nämlich notwendige Verhaltenskompetenzen abzudecken und bei Verlassen der ODDs ein MRM zu gewährleisten, umgesetzt.

4.2.2 Fehlervermeidung dank Echtzeitfähigkeit

Um das Sicherheitsziel S4 zu erreichen, also eine unzureichende Planungsfrequenz zu vermeiden, muss zum einen die Latenz der Verhaltensarbitration einschließlich der Rechenzeit der Verhaltensplanung der ausgewerteten Verhaltensoptionen gering gehalten werden. Insbesondere soll sie auch bei

steigender Anzahl an Verhaltensoptionen konstant bleiben, damit die Verhaltensarbitration skalierbar ist. Zum anderen muss die Verhaltensarbitration allerdings mindestens weiche Echtzeitanforderungen erfüllen, um das Sicherheitsziel S4 zu erfüllen.

Latenz

Eine effektive Maßnahme, um die Latenz eines Multithreading Systems zu reduzieren, ist es – bei ausreichend vorhandener Rechenleistung – rechenintensive Prozesse, wo möglich, zu parallelisieren. Bei der Verhaltensgenerierung zählt die Manöver- und Trajektorienplanung zu den rechenintensivsten Verarbeitungsschritten, während die Verhaltensentscheidung häufig auf einer a-priori Heuristik oder einer effizient umsetzbaren Evaluation der Verhaltensoptionen beruht. Auf die Verhaltensgenerierung mittels Arbitrationsgraphen übertragen bedeutet dies, dass die Funktionen der Verhaltensbausteine, insbesondere die `Aktion`-Funktion, rechenintensiv sind, während die Verhaltensarbitration einer an sich simplen und daher effizienten Rechenlogik folgt. Daher bietet es sich an, die Verhaltensbausteine parallel auszuwerten.

Allerdings steigt bei vollständiger Parallelisierung, d. h. einer parallelen Auswertung aller infrage kommenden Verhaltensoptionen, der Ressourcenaufwand linear mit der Anzahl dieser anwendbaren Optionen, womit das System nicht mehr skalierbar wäre. Letztlich ist ein Kompromiss aus Latenz, Ressourcenaufwand und Komfort gefragt: Keine Alternativoptionen auszuwerten würde bei Ausfällen sowie ungültigen oder unsicheren Trajektorien zur Ausführung der Fail-Safe-Trajektorie führen, die ggf. unverhältnismäßig konservativ und unkomfortabel ist. Dank der Redundanz und Diversität (Abschnitt 4.2.3) sind jedoch evtl. weitere anwendbare Alternativoptionen verfügbar, die die Situation sicher und komfortabel bewältigen können. Die Auswertung von Alternativoptionen ist daher aus Komfort- und somit auch Akzeptanzgründen unabdingbar. Die proaktiv volle Parallelisierung führt allerdings, wie bereits diskutiert, im ungünstigsten Fall zu einem linear mit der Anzahl an Verhaltensoptionen steigendem Ressourcenbedarf. Eine serielle Auswertung

von Alternativoptionen würde jedoch andererseits zu sich akkumulierender Laufzeit und somit hoher Latenz führen.

Es stellt sich somit vielmehr die Frage, wie und in welchem Umfang die Alternativoptionen ausgewertet werden sollen. In dieser Arbeit wird der folgende Kompromiss umgesetzt. Da die Start- und Fortsetzungs-Bedingungen wesentlich zur Verhaltensentscheidung sind und, durch ihre meist heuristische Umsetzung, eine vergleichsweise geringe Rechenlast verursachen, werden diese vollständig parallelisiert ausgewertet. Die eigentliche, rechenintensive Verhaltensplanung in den `Aktion`-Funktionen wird im Arbitrationsschritt hingegen nur für die n_{beste} besten anwendbaren Optionen $\mathcal{B} \subset \mathcal{A} \subset \mathcal{O}$ parallel ausgewertet, wobei n_{beste} bspw. auf maximal 3 festgelegt wird. Somit ist der Ressourcenaufwand an dieser Stelle begrenzt und dadurch unabhängig von der Gesamtanzahl an Verhaltensoptionen. Gleichzeitig gewährleistet die Auswertung von Alternativoptionen die Möglichkeit trotz Funktionsausfällen, nicht realisierbaren oder unsicheren Trajektorien dennoch ein sicheres und komfortables Verhalten zu generieren. Die Wahrscheinlichkeit, dass keine der Alternativoptionen $\mathcal{B}$ in der Lage ist, rechtzeitig eine gültige und sichere Trajektorie zu generieren, sinkt mit steigendem n_{beste}. Dank der Parallelisierung wird die Latenz zudem nicht durch die Anzahl der ausgewerteten Optionen n_{beste} beeinflusst.

Ereignis- oder Zeittrigger

Bzgl. Latenz ist neben der Parallelisierung außerdem entscheidend, nach welchem Schema – ereignis- oder zeitgetriggert – die Verhaltensentscheidung angestoßen wird. Bei einem Ereignistrigger wird die Verarbeitung, in diesem Falle die Verhaltensarbitration, durch ein Eingangssignal oder Ereignis, wie bspw. die Ankunft neuer Messdaten, angestoßen. Alternativ kann auch ein isochrones Zeitsignal, ein sog. Timer, als Trigger dienen.

Die einzelnen Bestandteile des Umweltmodells werden i. d. R. in unterschiedlichen Frequenzen aktualisiert. Während bspw. die Objekterkennung neue Ergebnisse typischerweise mit 5 to 10 Hz liefert, kann die Lokalisierung Frequenzen um die 150 Hz erreichen. Bei einer angestrebten Planungsfrequenz

von 5 to 10 Hz eignet sich daher der Eingang neuer Objektdaten als Triggerereignis für die Verhaltensarbitration. Um allerdings robust gegenüber einer ggf. gestörten Eingangsfrequenz zu sein, wird zusätzlich ein Timer mit 5 Hz eingesetzt. Somit werden neue Objektdaten unverzüglich und daher mit kleinstmöglicher Latenz zu neuen Verhaltensentscheidungen verarbeitet. Zugleich wird eine Mindestplanungsfrequenz von 5 Hz garantiert.

Echtzeitfähigkeit

Um die Echtzeitanforderungen zu erfüllen können klassische Methoden von Echtzeitsystemen angewandt werden [Wör05]. Dabei unterscheiden sich die Anforderungen in Industrie von denen in der universitären Forschung wesentlich. Im industriellen, seriennahen Kontext ist die harte Echtzeitfähigkeit zwingend notwendig, um den zuverlässigen und sicheren Betrieb über Millionen von Betriebsstunden hinweg zu gewährleisten. Folglich müssen alle Systemkomponenten, darunter das Betriebssystem, die eingesetzte Middleware sowie die Kommunikation der Soft- und Hardwaremodule untereinander, harten Echtzeitanforderungen genügen.

All diese Anforderungen können in Erprobungsfahrzeugen in der Regel nicht erfüllt werden, u. a. weil die Software auf gewöhnlichen Linux-Systemen ohne Echtzeitkernel betrieben werden [Meg05], die Recheneinheiten untereinander, mit der Sensorik oder mit der Aktorik über Ethernet (und somit meist mittels „Internet Protocol“) kommunizieren [Dan14] oder die ROS Middleware eingesetzt wird (die ebenfalls auf dem „Internet Protocol“ aufbaut)[1]. Daher werden im universitären Forschungskontext nur weiche Echtzeitanforderungen gestellt, solange ein geschulter Sicherheitsfahrer jederzeit die Kontrolle über das automatisierte Fahrzeug übernehmen kann.

Bei der angestrebten Planungsfrequenz von 5 to 10 Hz darf die Verarbeitungszeit bzw. Latenz für den gesamten Prozess der Verhaltensgenerierung- und

[1] Eines der Hauptziele von ROS2, dem Nachfolger des aktuellen de facto Standards ROS1, ist eine robuste und echtzeitfähige Neuentwicklung der populären Middleware, damit sie auch in industriellen Anwendungen eingesetzt werden kann [Zha17].

entscheidung folglich 200 ms nicht überschreiten. Die Verhaltensgenerierung wird daher bspw. auf 180 ms beschränkt, um in den verbleibenden 20 ms die Verhaltensentscheidung, samt Auswertung und Verifikation der infrage kommenden Trajektorien umsetzen zu können.

Zur Diagnose einer unzureichenden Reaktionszeit (vgl. Abschnitt 2.4) werden alle Funktionen der Verhaltensbausteine vom zugehörigen Arbitrator zeitbegrenzt und unterbrechbar aufgerufen. Liefert somit bspw. die `Aktion`-Funktion nicht rechtzeitig ein Ergebnis, wird der Aufruf unterbrochen und statt der Trajektorie ein definierter Fehlerwert $\eta_i^{\kappa} \in \mathbb{N}_0$ zurückgegeben. Die Rückgabe der `Aktion`-Funktion eines Verhaltensbausteins i wird hierfür um einen solchen potenziellen Fehlerwert zur sog. überwachten Stellgröße $\boldsymbol{\eta}_i^{\kappa}$ erweitert:

$$\boldsymbol{\eta}_i^{\kappa} = (\boldsymbol{u}_i^{\kappa}, \eta_i^{\kappa}), \tag{4.1}$$

wobei im Fehlerfall $\boldsymbol{u}_i^{\kappa} = \varnothing$ und sonst $\eta_i^{\kappa} = 0$ ist.

Durch die Auswertung der Alternativoptionen und dank der Fail-Safe-Trajektorie als Rückfallebene kann ein solcher Fehlerfall mittels Redundanz und Diversität im Arbitrationsschritt kompensiert werden (Abschnitte 4.2.3 und 4.2.4). Da eine unzureichende Rechenzeit i. d. R. nicht permanent ist, sondern bspw. durch kurzfristige Konvergenzprobleme verursacht wird, sollen an dieser Stelle keine weiteren Maßnahmen zur Fehlerbehandlung oder -ausgrenzung getroffen werden.

Durch die Eingrenzung der Latenz mittels Parallelisierung, den gleichzeitigen Einsatz von Ereignis- und Zeittriggern sowie die Festlegung und den Einsatz von Zeitschranken bei Funktionsaufrufen wird das Sicherheitsziel S4, nämlich eine unzureichende Planungsfrequenz zu vermeiden, erfüllt.

4.2.3 Ausfallsicherheit durch Fehlerkompensation

Einer der wesentlichen Beiträge dieser Arbeit besteht — wie in den Sicherheitszielen S3 und S5 formuliert — darin, die Verhaltensarbitration gegen Ausfälle der Verhaltensbausteine und ungültige Stellgrößen abzusichern. Beide Risiken können als Fehlzustände angesehen werden, sodass auch

hier Maßnahmen aus dem Gebiet der zuverlässigen und fehlertoleranten Systeme (Abschnitt 2.4) Anwendung finden sollen. Die Fehlerdiagnose und -behandlung übernehmen dabei die jeweiligen Arbitratoren. Bei der Fehlerbehandlung zahlt sich die konsequente Modularität und Entkopplung von Verhaltensbausteinen im Konzept der Verhaltensarbitration mittels Arbitrationsgraphen aus. Denn dadurch können Fehlzustände eingedämmt und eine drohende Fehlerfortplanzung verhindert werden.

Redundanz und Diversität

Die gängigste Variante der Fehlerbehandlung besteht in der Kompensation von Fehlern durch *Redundanz* und *Diversität* [Ech90]. Die Verhaltensarbitration bietet eine ideale Struktur, um damit trotz potenziellen Funktionsausfällen von Verhaltensbausteinen ein sicheres Verhalten zu generieren. Zunächst sollen die beiden Begriffe an einem einfachen Beispiel präzisiert werden, bevor sie im weiteren Verlauf dieses Abschnitts zum Einsatz kommen.

Redundanz Bei der Redundanz wird eine Komponente vervielfältigt bzw. *repliziert*, ohne ihre Funktionsweise zu verändern. Im Fall der Verhaltensarbitration wird hierzu ein Verhaltensbaustein mehrfach instanziiert und i. d. R. einem gemeinsamen Arbitrator zugewiesen.

Diversität Die Diversität – oder auch funktionelle Redundanz – beschreibt hingegen eine Funktion mit mehreren verschiedenartig implementierten Komponenten bzw. *Varianten* abzudecken. In der Verhaltensarbitration werden hierbei unterschiedliche Verhaltensbausteine, die die gleiche Verhaltenskompetenz adressieren, zum Arbitrationsgraphen hinzugefügt.

In beiden Fällen erkennt eine übergeordnete Instanz, wenn sich die Ausgaben der Komponenten unterscheiden und kann bspw. in sog. *2-von-3-Systemen* einzelne fehlerhafte Komponenten durch Mehrheitsentscheid kompensieren. Die Redundanz ist somit ein geeignetes Mittel gegen transiente als auch permanente Fehlzustände. Die Diversität ist zwar aufwendiger zu realisieren,

kann aber zusätzlich auch systematische Funktionsausfälle einzelner Komponenten ausgleichen.

Im automatisierten Fahren bietet sich der Einsatz von Redundanz u. a. bei Verhaltensbausteinen an, die in der Trajektorienplanung randomisierte Verfahren oder Initialisierungen nutzen. Konvergiert dann bspw. die Optimierung eines Verhaltensbausteins aufgrund einer ungünstigen Initialisierung nicht, liefern die Replikate mit einer anderen Initialisierung evtl. dennoch eine valide Trajektorie. Diversität eignet sich hingegen, um auch Entwurfsfehler in Verhaltensplanungsmethoden zu kompensieren. Scheitert bspw. der Frenet-basierte Planer eines Verhaltensbausteins an einer ungewöhnlichen Fahrbahngeometrie, kann ein anderer Verhaltensbaustein, der die Trajektorie im Euklidischen Raum berechnet, weiterhin gültige Stellgrößen erzeugen.

Der Einsatz beider Maßnahmen innerhalb der Verhaltensarbitration ist schließlich ohne weiteres möglich. Zudem könnten spezielle Arbitratoren mit Voting-Strategie *2-von-3-Systeme* umsetzen. In dieser Arbeit wird allerdings auf Voting-Mechanismen verzichtet und die Fehlerkompensation hauptsächlich mittels Diversität umgesetzt.

Ausfall eines Verhaltensbausteins

Zwar ist der Ausfall eines Verhaltensbausteins, bspw. in Form eines Progammabsturzes, in den Auswirkungen vergleichbar mit einem Verhaltensbaustein von zu hoher Latenz bzw. einem der die Echtzeitanforderungen nicht erfüllt (vgl. Abschnitt 4.2.2). Allerdings ist ein unbehandelter Absturz permanent und kann somit systemkritisch werden. Daher sollen Ausfälle explizit erkannt und gesondert behandelt werden.

Zur zuverlässigen Diagnose eines solchen Absturzes werden die Verhaltensbausteine mit einem sog. Heartbeat überwacht. Hierbei sendet jeder Verhaltensbaustein seinem übergeordneten Arbitrator in regelmäßigen Zeitabständen ein Signal, um anzuzeigen, dass er noch funktionsfähig ist. Bleibt dieser Heartbeat zu lange aus, kann der Arbitrator folglich davon ausgehen, dass der Verhaltensbaustein durch einen Absturz oder Verklemmung ausgefallen ist.

Wurde der Ausfall durch eine transiente Ursache ausgelöst, bspw. durch einen unerwartet ungewöhnlichen Zustand des Umweltmodells, kann dieser möglicherweise durch einen Neustart des Verhaltensbausteins behoben werden. Durch die damit verbundene Reinitialisierung und die sich in der Zwischenzeit veränderte Verkehrssituation, wird der Verhaltensbaustein mit hoher Wahrscheinlichkeit anschließend wieder funktionsfähig sein.

Ist die Fehlerursache allerdings persistent oder tritt zu häufig auf, wird der ausgefallene Verhaltensbaustein aus der Liste der Verhaltensoptionen $\mathcal{O}$ entfernt, um den Fehler auszugrenzen. Dadurch wird unterbunden, dass dieser einen Platz in der Menge der parallel auszuwertenden Optionen $\mathcal{B}$ belegt und damit die Auswertung einer besseren, weiterhin funktionsfähigen Option verhindert.

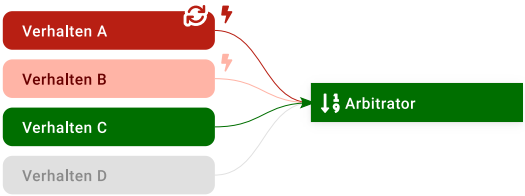


Abbildung 4.2: Redundanz und Diversität schützen den Arbitrationsgraphen vor Ausfällen von Verhaltensbausteinen (Fehlerkompensation). Ausgefallene Verhaltensbausteine werden zur Fehlerbehebung neu gestartet oder bei zu häufigen Ausfällen zur Fehlerausgrenzung deaktiviert. Beispiel im Bild: Das höchst priorisierte *Verhalten A* ist gerade ausgefallen und wird neu gestartet, *Verhalten B* wurde bereits wegen zu häufigen Ausfällen aus der Optionsliste des Arbitrators gestrichen, sodass dieser schließlich *Verhalten C* ausführt.

In beiden Fällen – der Fehlerbehebung und -ausgrenzung – werden die Maßnahmen allerdings erst zu einem späteren Planungsintervall wirksam. Der Neustart eines Verhaltensbausteins bspw. nimmt, je nach Komplexität der zu ladenden Planungskarte, mehrere Hundert Millisekunden in Anspruch. Daher ist es für die zuverlässige Verhaltensentscheidung notwendig, mögliche Fehler bereits zum Zeitpunkt ihres Auftretens zu kompensieren. Hier zeigt sich einer der Vorteile von Arbitrationsgraphen: Mit vergleichsweise geringem Mehraufwand können unterschiedliche Verhaltensbausteine, die dieselben Verhaltenskompetenzen adressieren, oder auch mehrfach die gleichen

Verhaltensbausteine als zusätzliche Verhaltensoptionen hinzugefügt werden. Die so realisierte Diversität und Redundanz ermöglicht es die Fehler anderer Verhaltensbausteine durch Auswertung von Alternativoptionen noch im betroffenen Planungsintervall zu kompensieren. Dank der Entkopplung der Verhaltensbausteine voneinander, hat ein Ausfall außerdem keinen Einfluss auf andere Verhaltensbausteine und setzt sich dank der Kapselung im Arbitrator nicht in der Verarbeitungskette fort. Abbildung 4.2 veranschaulicht diese Maßnahmen an einem einfachen Beispiel.

Durch die Fehlerdiagnose mittels Heartbeat, Fehlerbehebung mittels Neustart des Verhaltensbausteins, Fehlerausgrenzung durch Entfernen des Verhaltensbausteins und Fehlerkompensation durch Redundanz und Diversität wird das Sicherheitsziel S3 erreicht.

Ungültige Stellgrößen

Eine weitere mögliche Fehlerquelle sind ungültige oder nicht realisierbare Stellgrößen. Diese lassen sich, insbesondere bei Erprobungsfahrzeugen aus vielerlei Gründen nicht ausschließen. Zum einen führen bereits einfache Programmierfehler, wie bspw. nicht sachgemäß normierte Quaternionen, zu ungültigen Trajektorien. Zum anderen könnten die Soll-Trajektorien aus physikalischen Gründen, also aufgrund der Fahrzeugkinematik oder der Fahrdynamikgrenzen, nicht umsetzbar sein.

Die Fehlerdiagnose wird sowohl bzgl. der Gültigkeit als auch der Realisierbarkeit (sowie der Sicherheit in Abschnitt 4.2.4) mittels Verifikation umgesetzt. Hierfür wird die von einem Verhaltensbaustein i im Intervall κ zurückgegebene Stellgröße $\boldsymbol{u}_i^\kappa$ mit dem Verifikator $\mathcal{V}$ überprüft:

$$\nu_i^\kappa = \mathcal{V}(\boldsymbol{u}_i^\kappa) \tag{4.2a}$$

$$\boldsymbol{\nu}_i^\kappa = (\boldsymbol{\eta}_i^\kappa, \nu_i^\kappa), \tag{4.2b}$$

wobei die geprüfte Stellgröße $\boldsymbol{\nu}_i^\kappa$ aus der überwachten Stellgröße $\boldsymbol{\eta}_i^\kappa$ aus Gleichung (4.1) und einem Verifikationsergebnis $\nu_i^\kappa \in \mathbb{N}_0^3$ besteht.

Die Verifikation wird modular in den drei Einzelschritten $\mathcal{V}_{\text{gültig}}$, $\mathcal{V}_{\text{realisierbar}}$ und $\mathcal{V}_{\text{sicher}}$ durchgeführt und im Verifikator $\mathcal{V}$ vereint:

$$\mathcal{V}(\boldsymbol{u}_i^\kappa) = \left(\mathcal{V}_{\text{gültig}}(\boldsymbol{u}_i^\kappa), \mathcal{V}_{\text{realisierbar}}(\boldsymbol{u}_i^\kappa), \mathcal{V}_{\text{sicher}}(\boldsymbol{u}_i^\kappa)\right). \tag{4.3}$$

$\mathcal{V}_{\text{gültig}}$ prüft zunächst, ob die Trajektorie folgende Mindestanforderungen erfüllt:

- Die Trajektorie ist im korrekten Bezugssystem definiert.
- Die Trajektorie umfasst mindestens zwei Posen.
- Mindestens eine der Posen liegt in der Zukunft.
- Die Zeitabstände der Posen sind isochron.
- Die Quaternionen der Posen sind normalisiert.

Wird eine der Eigenschaften nicht erfüllt, gibt $\mathcal{V}_{\text{gültig}}(\boldsymbol{u}_i^\kappa) \in \mathbb{N}_0$ einen entsprechenden Fehlerwert zurück, sonst 0.

Der Verifikator $\mathcal{V}_{\text{realisierbar}}$ prüft die Trajektorie mittels kinematischen Einspurmodell auf folgende Grenzwerte:

- max. Geschwindigkeit,
- max. Beschleunigung,
- max. Ruck,
- max. Lenkwinkel,
- max. Drehrate und
- max. Querbeschleunigung.

Wird einer dieser Grenzwerte überschritten, gibt $\mathcal{V}_{\text{realisierbar}}(\boldsymbol{u}_i^\kappa) \in \mathbb{N}_0$ ebenfalls einen passenden Fehlerwert zurück, sonst 0.

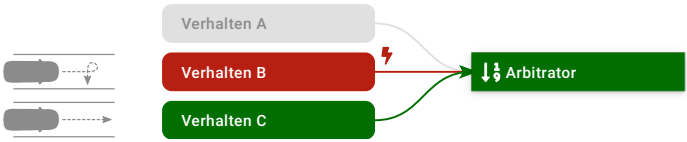


Abbildung 4.3: Ungültige Stellgrößen werden durch Verifikation vom Arbitrator erkannt und mittels Redundanz und Diversität kompensiert. Beispiel im Bild: Verhalten A mit höchster Priorität ist aktuell nicht anwendbar und Verhalten B hat eine ungültige Trajektorie zurückgegeben, sodass letztlich Verhalten C ausgeführt wird.

Da solche Stellgrößen i. d. R. transiente Ursachen haben, ist keine Fehlerbehebung oder -ausgrenzung vorgesehen. Stattdessen werden ungültige oder nicht-realisierbare Trajektorien, wie beim Ausfall von ganzen Verhaltensbausteinen, mittels Redundanz und Diversität von Verhaltensbausteinen kompensiert, wie in Abb. 4.3 beispielhaft dargestellt.

Folglich werden durch die Fehlerdiagnose mittels Verifikation und Fehlerkompensation durch Auswertung der Alternativoptionen ungültige oder nicht-realisierbare Manöver, wie durch das Sicherheitsziel S5 gefordert, erfolgreich verhindert.

4.2.4 Verkehrssicherheit durch Verifikation

Schließlich bildet das Sicherheitsziel S6, riskante, also unsichere Stellgrößen zu vermeiden, einen weiteren Kernbeitrag dieser Arbeit. Die Fehlerdiagnose erfolgt mittels Verifikation im Arbitrator, wobei die Sicherheit einer Trajektorie in dieser Arbeit über die Erreichbare-Mengen-Analyse (vgl. Abschnitt 2.5.2) überprüft wird. Es sei an dieser Stelle jedoch angemerkt, dass das Arbitrationsverfahren methodenunabhängig ist und somit auch andere Verfahren zur Verhaltensverifikation verwendet werden können oder gar sollten. Wegen der weitreichenden Annahmen der Erreichbare-Mengen-Analyse wäre eine Methode zu bevorzugen, die weichere Annahmen trifft[1],

[1] Beispielsweise indem sie einigen Verkehrsteilnehmern, wie Bussen, erlaubt aus ihrem Fahrstreifen herauszuragen.

sowie mögliche Regelverstöße (engl. „incompliant behavior“) und eine daraus potenziell resultierende Unfallschwere berücksichtigt. Die Formulierung einer geeigneten Verifikationsmethode soll allerdings nicht Teil dieser Arbeit sein. Daher wird aus Gründen der Anschaulichkeit, guten Anwendbarkeit und Nachvollziehbarkeit die Erreichbare-Mengen-Analyse verwendet.

Riskante Stellgrößen können bspw. dann entstehen, wenn ein Manöver zu knapp bemessen wurde (z. B. durch eine zu enge Lücke für einen Fahrstreifenwechsel), die zugrunde liegende Prädiktion den möglichen Verlauf der Szene unvollständig oder zu optimistisch abgebildet hat oder Unsicherheiten wie Verdeckungen und begrenzte Sensorabdeckung nicht berücksichtigt wurden.

Zur Diagnose, ob ein gegebenes Manöver sicher ist, wird die zugehörige Fail-Safe-Trajektorie $\check{\boldsymbol{x}}_i^{\kappa}$ aus Abschnitt 3.2.4 mit den Worst-Case-Belegungen $\mathcal{B}^{\kappa}_{\text{objekte}}$ aus dem Umweltmodell (Abschnitte 2.5.2 und 3.1) abgeglichen. Hierzu bestimmt der Verifikator $\mathcal{V}_{\text{sicher}}(\boldsymbol{u}_i^{\kappa})$ zunächst die zu erwartenden Ego-Belegungen der geplanten Fail-Safe-Trajektorie $\mathcal{B}^{\kappa}_{\text{ego,i}} = \mathcal{B}(\check{\boldsymbol{x}}_i^{\kappa})$. Diese werden für isochrone Zeitintervalle $\tau_l = [t_l, t_{l+1}]$ über den gesamten Planungshorizont $l \in [k, k+N]$ berechnet. Dabei werden die Posen der Fail-Safe-Trajektorie linear interpoliert, um von den diskreten Zeitschritten der Trajektorie auf kontinuierliche Zeitintervalle zu schließen.

Anschließend wird für jedes Zeitintervall geprüft, ob sich die Ego-Belegung $\mathcal{B}^{\kappa}_{\text{ego,i}}$ mit den Objekt-Worst-Case-Belegungen $\mathcal{B}^{\kappa}_{\text{objekte}}$ überschneidet:

$$\forall l \in [k, k+N] : \quad \varnothing \stackrel{?}{=} \mathcal{B}^{\kappa}_{\text{ego,i}}(\tau_l) \cap \mathcal{B}^{\kappa}_{\text{objekte}}(\tau_l). \tag{4.4}$$

Ist eine dieser Schnittmengen nicht leer, muss das Manöver als riskant eingestuft werden, sodass $\mathcal{V}_{\text{sicher}}(\boldsymbol{u}_i^{\kappa}) \in \mathbb{N}_0$ einen entsprechenden Fehlerwert zurückgibt. Ist das Manöver nachweislich sicher, wird hingegen 0 zurückgegeben.

Dank Gleichung (3.2b) folgt aus einer erfolgreichen Verifikation der Fail-Safe-Trajektorie $\check{\boldsymbol{x}}_i^{\kappa}$, dass auch die Soll-Trajektorie $\boldsymbol{x}_i^{\kappa}$ als sicher eingestuft werden kann. Denn sie stellt sicher, dass $\check{\boldsymbol{x}}_i^{\kappa}$ und $\boldsymbol{x}_i^{\kappa}$ bis zum Ende des nächsten Planungsintervalls t_{k+2} identisch sind. Bei sicherer Fail-Safe-Trajektorie hat

daher auch die Soll-Trajektorie im Zeitintervall $[t_k, t_{k+2}]$ keine Überschneidungen mit den Objekt-Worst-Case-Belegungen. Sollte zudem im nächsten Planungsintervall $\kappa+1$ keine neue sichere Soll-Trajektorie gefunden werden, kann die bereits verifizierte Fail-Safe-Trajektorie $\check{\boldsymbol{x}}_i^{\kappa}$ das Fahrzeug in einen MRC bringen.

Abbildung 4.4: Durch Verifikation auf Basis der Erreichbare-Mengen-Analyse werden zu riskante Manöver abgefangen und über Diversität und Redundanz kompensiert. Beispiel im Bild: Verhalten A hat höchste Priorität, allerdings hält die Trajektorie dieses Verhaltens nicht der Verifikation stand. Somit wählt der Arbitrator das sicherere Verhalten B.

Zur Fehlerbehandlung wird aufgrund der i. d. R. transienten Ursachen im Falle riskanter Stellgrößen ebenfalls auf Fehlerkompensation gesetzt. Sollte also $\boldsymbol{u}_i^{\kappa}$ nicht nachweislich kollisionsfrei bzgl. der Worst-Case-Prädiktionen sein, prüft der Arbitrator seine Alternativoptionen oder fällt auf spezialisierte MRMs zurück. Ein Minimalbeispiel hierzu ist in Abb. 4.4 dargestellt.

Somit führt die Verhaltensarbitration ausschließlich zugleich gültige, realisierbare und beweisbar sichere Manöver aus, also solche deren Verifikationsergebnis $\nu_i^{\kappa} = (0,0,0)^{\mathsf{T}}$ ist. Dies kann entweder eine der im aktuellen Planungsintervall κ evaluierten Optionen $\boldsymbol{u}_i^{\kappa} \in \mathcal{A}$ (vgl. Gleichung (2.3)) sein, wenn sie der Verifikation standhält, oder die bereits verifizierte Fail-Safe-Trajektorie des vorigen Planungsintervalls $\check{\boldsymbol{x}}^{\kappa-1}$ sein.

Startet das Fahrzeug zudem in einem sicheren Zustand $\boldsymbol{x}_0$, folgt die Verkehrssicherheit zum beliebigen Zeitpunkt $\boldsymbol{x}_k$ durch Induktion, solange die Annahmen der Erreichbare-Mengen-Analyse erfüllt sind. Sollte eine dieser Annahmen verletzt werden (weil bspw. ein Fußgänger unerwartet eine Straße kreuzt) und keine beweisbar sichere Option gefunden werden, wird dennoch die Fail-Safe-Trajektorie des vorigen Planungsintervalls als MRM ausgeführt, um das Risiko einer Kollision zu minimieren. Es kann sich zudem

anbieten für solche Fälle einen zusätzlichen noch konservativeren MRM Verhaltensbaustein zu entwerfen, der das Kollisionsrisiko ungeachtet dieser Annahmen minimiert. Schließlich muss der Arbitrator an der Wurzel des Graphen über mind. eine immer anwendbare Rückfalloption verfügen. Diese darf – als einzige Ausnahme – auch bei fehlgeschlagener Verifikation ausgeführt werden, damit das Sicherheitsziel S2, jederzeit ein MRM zur Verfügung zu stellen, erfüllt bleibt.

Dank der Fehlerdiagnose mittels Verifikation durch Erreichbare-Mengen-Analyse und der Auswertung von Alternativoptionen zur Fehlerkompensation wird das Sicherheitsziel S6 schließlich – unter den Annahmen der Erreichbare-Mengen-Analyse – erfüllt.

4.3 Systemübersicht

Dieser Abschnitt fasst die wesentlichen Beiträge dieses Kapitels zusammen. Insbesondere sei an dieser Stelle nochmals darauf hingewiesen, dass das vorgestellte Sicherheitskonzept ebenso auf andere Domänen übertragbar ist und somit nicht auf die Anwendung des automatisierten Fahrens beschränkt ist. Die Verhaltensentscheidung mittels Arbitrationsgraphen ist ohnehin domänenunabhängig, sodass neben den anwendungsspezifischen Verhaltensbausteinen, dem Umweltmodell und der Manöver- bzw. Stellgrößenrepräsentation, lediglich die Verifikatoren $\mathcal{V}(\boldsymbol{u}_i^{\kappa})$ aus Gleichung (4.3) auf die jeweilige Domäne übertragen werden müssen.

Entsprechend beschreibt Algorithmus 3 die Erweiterung des Arbitrationsverfahrens zu einer sicheren und robusten Verhaltensarbitration mittels Stellgrößenverifikation in einer möglichst generischen Form. Für eine gute Vergleichbarkeit ist die Formulierung hierbei an Algorithmus 1 angelehnt. Zunächst wird in Zeile 15 die aktuelle Situation $\boldsymbol{s}$ bestimmt. Ausgehend hiervon bestimmt der Wurzelarbitrator[1] seine beste anwendbare und sichere Aktion mit der Funktion `BesteAnwendbareAktion(`$\boldsymbol{s}$`)`. Hierzu filtert er aus seinen

[1] Der Arbitrator an der Wurzel des Arbitrationsgraphen.

Algorithmus 3 : Generisches Arbitrationsverfahren mit Verifikation

```
1  function BesteAnwendbareAktion(Situation s)
2      Filtere anwendbare Optionen A ⊂ O
3      Sortiere anwendbare Optionen A* = ⟨a_0, a_1, ...⟩ = strategie(A)
4      for a ∈ A* do
5          Bestimme Stellgröße u_a = Aktion_a(s)
6          Führe Verifikation v_a = V(u_a) durch
7          if Verifikation bestanden v_a = 0 then
8              return (u_a, v_a)
9          end
10     end
11     return (∅, NO_SAFE_OPTION)
12 end
13
14 while true do
15     Bestimme die aktuelle Situation s
16     Bestimme ν = (u, v) = BesteAnwendbareAktion(s)
17     if Verifikation bestanden v = 0 then
18         Führe u aus
19     end
20 end
```

Optionen[1] $\mathcal{O}$ diejenigen Optionen $\mathcal{A}$ heraus, die in der Situation $\boldsymbol{s}$ anwendbar sind. Diese wiederum sortiert er zu einer – je nach zugrundeliegender Strategie – absteigend sortierten Liste A^*. Nun wird für jede[2] Option $a \in A^*$ geprüft, ob ihre Stellgröße $\boldsymbol{u}_a = \texttt{Aktion}_a(\boldsymbol{s})$ einer Verifikation $\mathcal{V}(\boldsymbol{u}_a)$ standhält. Falls ja, wird diese als die bestmögliche anwendbare und zugleich sichere Option zurückgegeben. Besteht keine der Optionen den Verifikationsschritt, gibt der Arbitrator den Fehlerwert NO_SAFE_OPTION zurück. Gibt der Wurzelarbitrator schließlich eine Aktion $\boldsymbol{u}$ zurück, die die Verifikation bestanden hat, kann diese – unter Vorbehalt der Annahmen des verwendeten Verifikators $\mathcal{V}$ – ausgeführt werden.

[1] Jede Option kann, wie bereits im klassischen Arbitrationsverfahren, ein Verhaltensbaustein oder wiederum selbst ein Arbitrator sein. Solche untergeordneten Arbitratoren bestimmen ihre bestmögliche Option ebenfalls wie in der Funktion `BesteAnwendbareAktion(`$\boldsymbol{s}$`)` beschrieben.

[2] Die Logik zur parallelen Auswertung von insgesamt nur n_{beste} besten anwendbaren Optionen $\mathcal{B} \subset \mathcal{A} \subset \mathcal{O}$ aus Abschnitt 4.2.2 wurde an dieser Stelle vereinfachend durch eine sequenzielle Auswertung aller anwendbaren Optionen $\mathcal{A} \subset \mathcal{O}$ ersetzt.

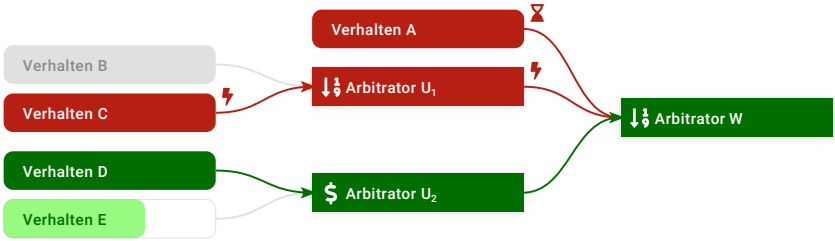


Abbildung 4.5: Beispielgraph für eine sichere und robuste Verhaltensarbitration. *Verhalten A* kann die Echtzeitanforderungen nicht erfüllen, *Verhalten B* ist in der aktuellen Situation nicht anwendbar und *Verhalten C* ist zu riskant, sodass *Verhalten D* (wegen seines höheren Nutzens gegenüber *Verhalten E*) ausgewählt und ausgeführt wird.

Abbildung 4.5 veranschaulicht das Konzept an einem Beispiel mit dem Wurzelarbitrator Arbitrator W, zwei untergeordneten Arbitratoren Arbitrator U_1 und U_2 sowie fünf Verhaltensbausteinen *Verhalten A* bis *E*. Die Optionen *Verhalten A, C, D* und *E* sind in der aktuellen Situation anwendbar, wobei *Verhalten A* höchste und *Verhalten C* zweithöchste Priorität hat und zudem *Verhalten D* höheren Nutzen gegenüber *Verhalten E* hat. Der Wurzelarbitrator Arbitrator W wertet zur Reduktion von Latenzen die drei vielversprechendsten Optionen (nämlich *Verhalten A, C* und *D*) aus. *Verhalten A* liefert allerdings nicht rechtzeitig ein Ergebnis und verfehlt damit die Echtzeitanforderungen. *Verhalten C* gibt zwar rechtzeitig eine Trajektorie zurück, allerdings ist diese laut Verifikation zu riskant. *Verhalten D* liefert jedoch rechtzeitig eine gültige, realisierbare und sichere Trajektorie, sodass diese schließlich ausgeführt wird.

5 Verhaltensbausteine in der Anwendung

Während die NHTSA und Waymo bereits zahlreiche Verhaltenskompetenzen definierten [Way20a], stellt Abschnitt 5.1 vor, wie Verhaltensbausteine entworfen werden können, um eine Auswahl grundlegender Fahrmanöver für das automatisierte Fahren im urbanen Raum zu realisieren. Abschnitt 5.2 beschreibt die Verhaltensbausteine der Rückfallebenen, die für eine robuste und sichere Verhaltensarbitration von wesentlicher Bedeutung sind. Kapitel 6 wird daraufhin aufzeigen, wie diese in Arbitrationsgraphen eingebettet werden und schließlich in der Praxis Anwendung finden.

5.1 Grundlegende Fahrmanöver

Die hier vorgestellten Verhaltensbausteine wurden insbesondere auf die Befahrung der Karlsruher Adenauer-Route mit dem Erprobungsfahrzeug *Bertha* ausgelegt (weitere Details siehe Abschnitt 6.1), sind im Allgemeinen jedoch auch auf andere Strecken und Fahrzeuge übertragbar. Die Verhaltensbausteine sind folglich geografisch auf die zuvor festgelegte Strecke und, um eine hinreichende Qualität der Lokalisierung und Umgebungserfassung zu gewährleisten, bzgl. Umweltbedingungen auf sonnig trockenes Wetter bis einschließlich gemäßigten Regen bei Tag beschränkt. Aus Gründen der Übersichtlichkeit wird im Folgenden darauf verzichtet, diesen auf alle Verhaltensbausteine zutreffenden Teil der ODDs in den jeweiligen Start- und Fortsetzungs-Bedingungen zu wiederholen.

5.1.1 Ausparken

Die Fahrt eines automatisierten Fahrzeugs beginnt i. d. R. mit dem Ausparken aus einer Parklücke oder Garage. Ziel des Ausparkens ist es, das Fahrzeug in eine geeignete Pose auf einen Fahrstreifen zu bewegen, damit die Fahrt anschließend ausgehend von diesem Fahrstreifen beginnen kann. Dieser sog. Startfahrstreifen ist entweder vorab definiert worden oder muss ausgehend von der Parkpose dynamisch bestimmt werden.

Bei großen Parkplätzen oder gar Parkhäusern, wie sie bspw. am Einzelhandel, an Flughäfen und Stadien zu finden sind, bezieht sich der Startfahrstreifen auf einen ggf. virtuellen Fahrstreifen innerhalb des Parkplatzes. Somit sind noch weitere Verhaltensbausteine notwendig, die das Fahrzeug nach dem Ausparken aus der Parklücke über den Parkplatz navigieren und schließlich in den Straßenverkehr einfädeln. Auf jene weiteren Verhaltensbausteine soll in diesem Abschnitt allerdings nicht weiter eingegangen werden. Abbildung 5.1 veranschaulicht entsprechend das Ausparken aus einer Parklücke parallel zur Fahrbahn.

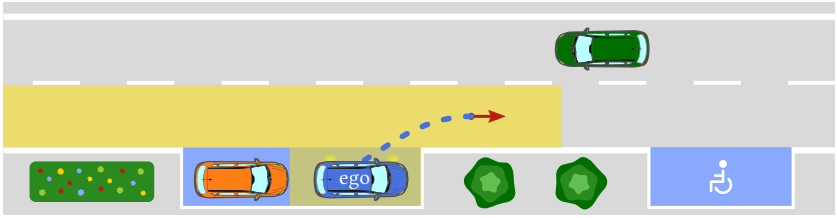


Abbildung 5.1: Ausparken aus einer parallel zur Fahrbahn angeordneten Parklücke. Zum sicheren Ausparken in die rot gekennzeichnete Zielpose muss mind. die gelb markierte Fläche einsehbar sein. Dadurch kann sichergestellt werden, dass die notwendige Rangierfläche frei ist und kein herannahender Verkehr im Startfahrstreifen gefährdet wird. Eine mögliche Soll-Trajektorie ist in Blau dargestellt, der Fahrtrichtungsanzeiger kündigt das Manöver an.

Start-Bedingung

Wie in Abschnitt 3.2.3 diskutiert, leiten sich die Start- und Fortsetzungs-Bedingungen im automatisierten Fahren von den ODDs des Verhaltens ab. Der

Verhaltensbaustein zum Ausparken ist folglich unter folgenden Bedingungen anwendbar:

Start-Bedingung zum Ausparken

Der Verhaltensbaustein zum Ausparken kann aufgerufen werden, wenn

- das Ego-Fahrzeug im Stillstand ist,
- sich auf einem Parkplatz oder einer anderweitigen Freifläche befindet, die unmittelbar an den Startfahrstreifen angrenzt,
- es zudem von einer freien, hinreichend großen Rangierfläche umgeben ist und
- der zum Rangieren notwendige Bereich, einschließlich eines hinreichend langen Teil des Startfahrstreifens, einsehbar und
- frei von anderen Verkehrsteilnehmern ist.

Die notwendige Rangierfläche hängt dabei von den Fahrzeugdimensionen, kinematischen Grenzen wie des maximalen Lenkwinkels, der Start- und Endpose sowie der verwendeten Methode zur Pfadplanung ab. Wie weit zudem der Startfahrstreifen einsehbar sein muss, hängt von der dort zugelassenen Höchstgeschwindigkeit ab, um es einem ggf. herannahenden Fahrzeug zu ermöglichen rechtzeitig zum Stillstand zu kommen. Nimmt man in einer 30er-Zone bspw. eine zumutbare Bremsbeschleunigung von $1\,\frac{\mathrm{m}}{\mathrm{s}^2}$ an, müssen mind.

$$\frac{v_{\max}^2}{2a_{\mathrm{comf}}} = \frac{\left(30\,\frac{\mathrm{km}}{\mathrm{h}}\right)^2}{2 \cdot 1\,\frac{\mathrm{m}}{\mathrm{s}^2}} \approx \frac{\left(8{,}3\,\frac{\mathrm{m}}{\mathrm{s}}\right)^2}{2\,\frac{\mathrm{m}}{\mathrm{s}^2}} \approx 34{,}4\,\mathrm{m} \tag{5.1}$$

einsehbar sein. Bezieht man noch eine Reaktionszeit von 1 s an, erhöht sich der Sicherheitsabstand auf $34{,}4\,\mathrm{m} + 8{,}3\,\mathrm{m} = 42{,}7\,\mathrm{m}$.

Da die Sicherheit der Einparktrajektorie im Rahmen des Sicherheitskonzepts überprüft wird (Abschnitt 4.2.4), bilden die Start- und Fortsetzungs-Bedingungen an dieser Stelle nur eine heuristische Einschätzung ab. Um allerdings zu vermeiden, dass die Verifikation Einparktrajektorien regelmäßig aufgrund von zu optimistischen Start- und Fortsetzungs-Bedingungen ablehnen muss,

sollten die in den Verhaltensbausteinen eingesetzten Sicherheitsabstände großzügig bemessen werden.

Fortsetzungs-Bedingung

Die Fortsetzungs-Bedingung zum Ausparken orientiert sich stark an der Start-Bedingung, wobei sie den Einparkvorgang beendet, sobald die Zielpose näherungsweise erreicht wurde:

Fortsetzungs-Bedingung zum Ausparken

Der Verhaltensbaustein zum Ausparken sollte fortgesetzt werden, solange

- sich das Ego-Fahrzeug auf einem Parkplatz, einer anderweitigen Freifläche, die unmittelbar an den Startfahrstreifen angrenzt, oder dem Startfahrstreifen selbst befindet,
- der zum Rangieren notwendige Bereich, einschließlich eines hinreichend langen Teil des Startfahrstreifens, einsehbar und
- frei von anderen Verkehrsteilnehmern ist und
- die Zielpose nicht hinreichend genau erreicht worden ist.

Verhaltensgenerierung

Gegeben einer Start- und Zielpose steht eine Vielzahl von Methoden zur Pfad- oder Trajektorienplanung zur Verfügung. Eine Übersicht zu Trajektorienplanern für das Ein-/Ausparken gibt u. a. [Ban17].

Bspw. nutzt [Sch16] einen dreistufigen Ansatz, um möglichst einfache und platzsparende Trajektorien zu generieren. Im ersten Schritt werden Start- und Zielpose unter Anwendung der Reeds-Shepp Vorschriften [Ree90] verbunden. So entsteht ein Pfad aus zwei Liniensegmenten, die ohne Richtungswechsel mit einem Kreissegment verbunden sind. Ist dieser Pfad nicht realisierbar, wird im zweiten Schritt ein ebenfalls deterministischer Reeds-Shepp Pfad mit zwei Richtungswecheln bestimmt. Führt auch dieser nicht kollisionsfrei zur Zielpose, wird der Pfad mittels Hybrider A*-Graphensuche [Dol10] ermittelt.

[Ban18] erweitert die Pfadplanung mittels A*-Graphensuche und Bewegungsprimitiven um Unsicherheiten insbesondere in der Lokalisierung und Regelung, sodass auch in beengten Räumen sichere Trajektorien robust generiert werden.

Für die Anwendung im Realversuch ist folglich der dreistufige Ansatz aus [Sch16], mit der Erweiterung um Unsicherheiten aus [Ban18] zu empfehlen.

Abgesehen von der Soll-Trajektorie ist beim Ausparken wichtig, als Teil der HMI den Fahrtrichtungsanzeiger richtig zu setzen. Eine beispielhafte Soll-Trajektorie samt Fahrtrichtungsanzeiger ist in Abb. 5.1 veranschaulicht.

5.1.2 Einparken

Zum Schluss einer Fahrt soll das Fahrzeug meist in einer Parklücke oder Garage, bei einem Mobilitätsdienst zumindest vorübergehend in einer Haltebucht, geparkt werden. Ausgehend von der Haltepose im letzten Fahrstreifen der Route, dem sog. Zielfahrstreifen, und einer angestrebten Zielparkpose wird eine Einparktrajektorie bestimmt. Dabei wurde die Zielparkpose entweder bereits festgelegt oder muss dynamisch bestimmt werden. Auf großen Parkplätzen oder in Parkhäusern wird dem Einparkvorgang die Parkplatzsuche als eigener Verhaltensbaustein vorangestellt, wobei hierauf im Folgenden nicht näher eingegangen wird. Stattdessen stellt Abb. 5.2 das Einparken in eine parallel zur Fahrbahn angeordnete Parklücke dar.

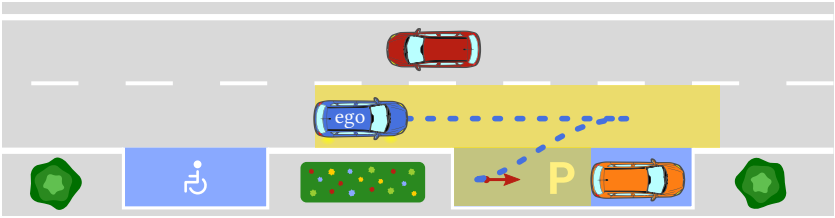


Abbildung 5.2: Im Zielfahrstreifen angekommen, soll das Ego-Fahrzeug in die gewählte Parklücke mit in rot dargestellter Zielpose einparken. Die gelb hervorgehobene Fläche muss einsehbar und frei von Hindernissen sein, um ein sicheres Rangieren zu ermöglichen. Eine bspw. mittels Reeds-Shepp bestimmte Soll-Trajektorie ist in Blau dargestellt, während der Fahrtrichtungsanzeiger zum Einparken blinkt.

In ihren Start- und Fortsetzungs-Bedingungen sowie der Verhaltensgenerierung ähnelt das Einparkverhalten stark dem im vorigen Abschnitt beschriebenen Verhaltensbaustein zum Ausparken.

Start-Bedingung

Muss zum Einparken, bspw. in eine direkt an den Zielfahrstreifen angrenzende Parklücke, kein anderer Fahrstreifen gekreuzt werden, vereinfacht sich im Vergleich zum Ausparken die Start-Bedingung. In diesem Fall muss lediglich die Rangierfläche einsehbar und frei sein:

Start-Bedingung zum Einparken

Der Verhaltensbaustein zum Einparken kann aufgerufen werden, wenn

- das Ego-Fahrzeug nahezu im Stillstand ist,
- sich auf einem Zielfahrstreifen befindet, der unmittelbar an einen Parkplatz oder eine anderweitige zum Parken geeignete Freifläche grenzt,
- es zudem von einer hinreichend großen Rangierfläche umgeben ist und
- diese einsehbar sowie
- frei von anderen Verkehrsteilnehmern ist.

Fortsetzungs-Bedingung

Die Fortsetzungs-Bedingung vereinfacht sich ebenfalls im Vergleich zum Ausparken:

Fortsetzungs-Bedingung zum Einparken

Der Verhaltensbaustein zum Einparken sollte fortgesetzt werden, solange

- sich das Ego-Fahrzeug weiterhin auf dem Zielfahrstreifen oder dem angesteuerten Parkplatz bzw. der angesteuerten Parkfläche befindet,
- der zum Rangieren notwendige Bereich einsehbar und
- frei von anderen Verkehrsteilnehmern ist sowie
- die angestrebte Zielparkpose nicht hinreichend genau erreicht worden ist.

Verhaltensgenerierung

Für die Pfadplanung zum Einparken eignen sich Methoden wie sie in Abschnitt 5.1.1 vorgestellt wurden. Darunter fallen u. a. deterministische Reeds-Shepp Pfade sowie die Hybride A*-Graphensuche.

Anschließend kann das Längsverhalten entlang des so bestimmten Pfades bspw. mittels Direkter Optimierung eines Mehrfachintegrators unter quadratischen Kosten bestimmt werden [Pek18].

Analog zum Ausparken muss auch beim Einparken der Fahrtrichtungsanzeiger als Teil der HMI gesetzt werden. Abbildung 5.2 visualisiert eine Beispieltrajektorie und den entsprechend aktivierten Fahrtrichtungsanzeiger.

5.1.3 Folgefahrt

Die Folgefahrt, also einem Fahrstreifen und dem darin ggf. vorausfahrenden Verkehr zu folgen, bildet die Grundlage des geregelten Straßenverkehrs. Sie gehört zugleich zu den zeitlich betrachtet häufigsten und entsprechend auch zu den am besten untersuchten Fahrmanövern [Win15, Bar15].

Unter kontrollierten Bedingungen, also bei u. a. wohldefinierten Fahrstreifen wie bspw. auf Autobahnen, wird die Längsregelung zur Folgefahrt bereits seit 1995 in Serienfahrzeugen als Assistenzsystem verbaut (genannt Adaptive Cruise Control (ACC)) [Win15]. Das erste vollwertige Level 3 System wird hingegen voraussichtlich erst 2022 Serienreife erreichen. Dann soll der, auf zunächst 60 $\frac{\text{km}}{\text{h}}$ beschränkte, sog. *DRIVE PILOT*, die Quer- und Längsführung auf Autobahnen übernehmen können [Mer21, Gre21a].

Bei der Folgefahrt im urbanen Raum, müssen im Längsverhalten zusätzlich zu vorausfahrendem Verkehr Stopp- und Haltelinien berücksichtigt werden, auch wenn spezialisierte Verhalten existieren sollten, die diese Halte- und Vorfahrtmanöver adressieren. Dadurch wird zum einen vermieden diese Stopp- und Haltelinien zu schnell anzufahren bis ein entsprechender Spezialbaustein ausgeführt wird. Zum anderen wird hiermit insbesondere sichergestellt, dass der Verhaltensbaustein zur Folgefahrt auch selbst an diesen (ggf. virtuellen)

Halte- und Stopplinien anhält und somit das Fahrzeug unabhängig von anderen Verhaltensbausteinen in einen MRC bringen kann.

Start-Bedingung

Zur Folgefahrt orientiert sich das Ego-Fahrzeug am aktuellen Fahrstreifen, der entweder aus Sensordaten erfasst wurde oder aus einer Planungskarte bekannt ist. Im Längsverhalten kann die Folgefahrt sowohl mit als auch ohne vorausfahrenden Verkehr durchgeführt werden, sodass sich hieraus keine weiteren Einschränkungen an die Start-Bedingung ergeben:

Start-Bedingung zur Folgefahrt

Der Verhaltensbaustein zur Folgefahrt kann aufgerufen werden, solange

- sich das Ego-Fahrzeug innerhalb eines Fahrstreifens der Route befindet,
- unabhängig davon, ob andere Fahrzeuge im selben Fahrstreifen vorausfahren oder nicht.

Fortsetzungs-Bedingung

Die Folgefahrt kann immer dann fortgesetzt werden, wenn sich das Ego-Fahrzeug in einem Fahrstreifen der Route befindet. Da eben dies die Aufgabe der Folgefahrt ist, nämlich das Ego-Fahrzeug im Fahrstreifen zu halten, und keine sonstigen Anforderungen zum Fortsetzen der Folgefahrt erfüllt sein müssen, sollte der Verhaltensbaustein also i. d. R. immer fortgesetzt werden können. Als Vorsichtsmaßnahme wird in der Fortsetzungs-Bedingung dennoch geprüft, ob sich das Fahrzeug weiterhin im Fahrstreifen befindet.

Eine interessante Fragestellung betrifft das Ende eines Fahrstreifens: Soll die Folgefahrt auch dann aufrufbar sein oder fortgesetzt werden können, wenn der Fahrstreifen in Kürze endet? Soll also der Verhaltensbaustein zur Folgefahrt auch ein notwendiges Haltemanöver am Ende eines Fahrstreifens umsetzen? Eine Option wäre spezielle Verhaltensbausteine für solche Haltemanöver zu entwerfen, die am Ende des Fahrstreifens mit höherer Priorität übernehmen. Analog zur den Stopp- und Haltelinien innerhalb eines Fahrstreifens

werden in dieser Arbeit allerdings auch Haltemanöver am Ende eines Fahrstreifens als Teil der Folgefahrt realisiert. Somit bringt der Verhaltensbaustein *Folgefahrt* das Fahrzeug am Ende eines Fahrstreifens zum Halten und kann aktiv bleiben bis ein anderer Verhaltensbaustein ausgeführt wird:

Fortsetzungs-Bedingung zur Folgefahrt

Die Fortsetzungs-Bedingung des Verhaltensbausteins zur Folgefahrt stimmt mit seiner Start-Bedingung überein. Er kann also fortgesetzt werden, solange

- sich das Ego-Fahrzeug innerhalb eines Fahrstreifens der Route befindet,
- unabhängig davon, ob andere Fahrzeuge im selben Fahrstreifen vorausfahren oder nicht.

Verhaltensgenerierung

Zur Verhaltens- und Trajektorienplanung der Folgefahrt wird zunächst, abhängig von der aktuellen Position und der gewählten Route, der aktuelle Fahrkorridor bestimmt. Hierzu werden ausgehend von dem aktuellen Ego-Lanelet aufeinander folgende Lanelets konkateniert solange sie der Route folgen und innerhalb des Planungshorizonts liegen. Innerhalb dieses Korridors wird daraufhin ggf. unter Zuhilfenahme der Korridormittellinie die Soll-Trajektorie bestimmt. Hierzu eignet sich bspw. die Nichtlineare Optimierung, wie in Abschnitt 2.3.3 beschrieben. Abbildung 5.3 veranschaulicht zusammenfassend wie aus den grauen Lanelets ein Fahrkorridor in Blau bestimmt wird und hieraus, unter Berücksichtigung des vorausfahrenden Fahrzeugs, schließlich eine blau gestrichelte Soll-Trajektorie geplant wird.

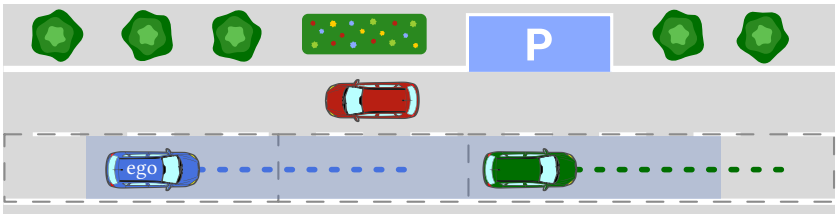


Abbildung 5.3: Die Folgefahrt hält das Ego-Fahrzeug im Fahrstreifen bzw. -korridor (blau unterlegt) und berücksichtigt vorausfahrenden Verkehr sowie Halte- und Stopplinien. Die zugrunde liegenden Lanelets sind grau gestrichelt dargestellt, die entsprechende Soll-Trajektorie ist blau gestrichelt.

Der Einbezug der Haltemanöver in den Verhaltensbaustein zur Folgefahrt führt bspw. am Ende der Route dazu, dass die Folgefahrt das Fahrzeug zum Halten bringt und anschließend das Einparkverhalten übernehmen kann. Abbildung 5.4 verdeutlicht eine solche Szene. Die Route endet in diesem Beispiel neben der Zielparklücke. Infolge dessen kommt die rote Soll-Trajektorie am Ende des ebenfalls dort endenden Fahrkorridors zum Stehen.

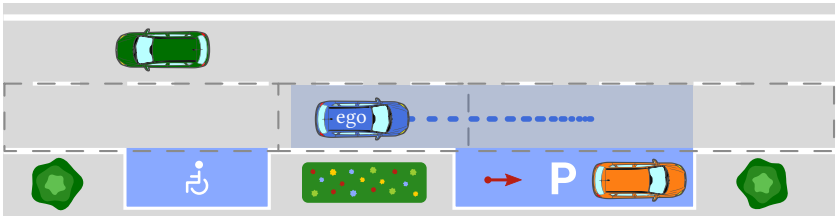


Abbildung 5.4: Am Ende der Route wird der Fahrkorridor trotz nachfolgender Lanelets abgeschnitten. So wird erreicht, dass die Soll-Trajektorie der Folgefahrt (blau) dort zum Stehen kommt und bspw. ein Einparkmanöver übernehmen kann.

An Stellen, an welchen ein Fahrstreifenwechsel notwendig ist, aber aktuell bspw. wegen starkem Verkehrsaufkommen nicht ausgeführt werden kann, endet der Fahrkorridor der Folgefahrt virtuell an der letztmöglichen Position zum Fahrstreifenwechsel, wie in Abb. 5.5 dargestellt. Dadurch wird der Verhaltensbaustein zur Folgefahrt die Geschwindigkeit reduzieren und nicht

an der angepeilten Ausfahrt vorbeifahren. Sollte der Fahrstreifenwechsel dennoch nicht rechtzeitig gelingen, kann es, je nach Verkehrsaufkommen, sinnvoll sein nach einer Weile die Route neu zu planen, um ein Blockieren der Fahrbahn zu vermeiden.

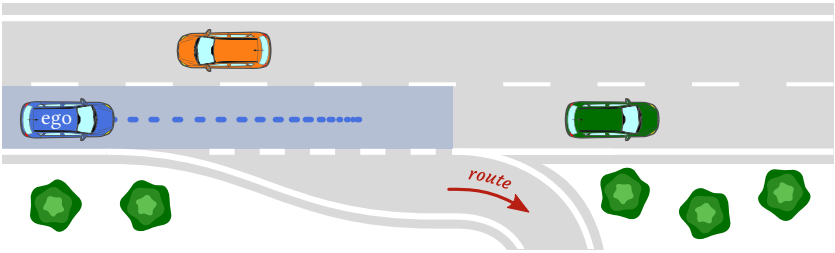


Abbildung 5.5: Ist zum Folgen der Route ein Fahrstreifenwechsel notwendig, wird der Fahrkorridor der Folgefahrt an der letztmöglichen Stelle abgeschnitten, an welcher noch ein Fahrstreifenwechsel möglich ist.

5.1.4 Kreuzung passieren

An Ein- und Ausfahrten, Fußgängerüberwegen, Kreuzungen sowie Engstellen muss die Vorfahrt anderer Verkehrsteilnehmer berücksichtigt werden. Diese Vorfahrtsituationen sind in der Praxis entweder implizit durch Verkehrsregeln wie der sog. *Rechts vor Links Regel*[1] und der Vorschrift bzgl. abgesenkten Bordsteinen[2] oder explizit durch Verkehrszeichen bzw. Fahrbahnmarkierungen geregelt. Wie in Abschnitt 3.1 erläutert, sind eben jene Verkehrs- und Vorfahrtsregeln in der Planungskarte, auf die der Verhaltensbaustein zugreifen kann, abgebildet.

Da die Folgefahrt bereits an den vorfahrtsbezogenen Stopp- und Haltelinien hält, wird die Aufgabe des Haltens an dieser Stelle tatsächlich dem Verhaltensbaustein *Folgefahrt* überlassen. Darüber hinaus übernimmt *Kreuzung*

[1] „An [nicht besonders geregelten] Kreuzungen und Einmündungen hat die Vorfahrt, wer von rechts kommt." (§8 Absatz 1 Satz 1 StVO)

[2] „Wer [...] über einen abgesenkten Bordstein hinweg auf die Fahrbahn einfahren [...] will, hat sich dabei so zu verhalten, dass eine Gefährdung anderer Verkehrsteilnehmer ausgeschlossen ist; erforderlichenfalls muss man sich einweisen lassen." (§10 Satz 1 StVO)

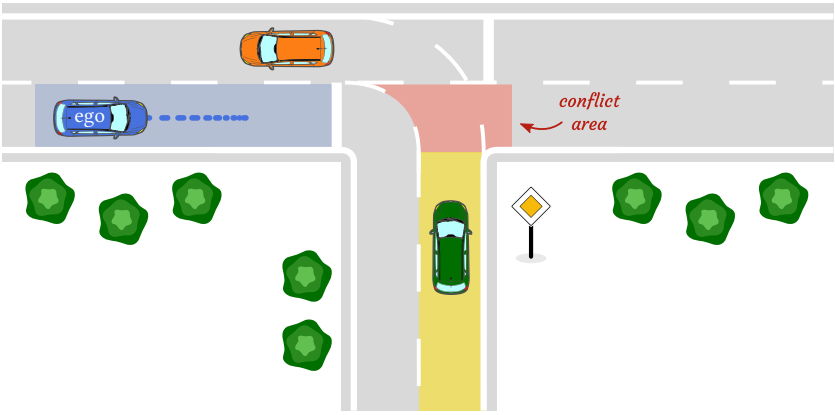
passieren allerdings die Entscheidung, ob die Vorfahrt hinreichend gewährt wurde und die Kreuzung[1] nun sicher und legal passiert werden kann. Dabei unterscheidet sich *Kreuzung passieren* im Querverhalten nicht von *Folgefahrt*, deaktiviert allerdings im Längsverhalten die betreffende virtuelle Haltelinie, um sie zu queren.

Alternativ hätte ein Verhaltensbaustein entworfen werden können, der sowohl den Haltevorgang sowie das anschließend sichere Passieren der Haltelinie durchführt. In diesem Falle wäre für den Anwender allerdings nicht ersichtlich wann und warum die Haltelinie passiert wird. Mit *Kreuzung passieren* ist die Entscheidungsgrundlage hingegen explizit in der Start-Bedingung definiert, was zu einer guten Nachvollziehbarkeit beiträgt. Zudem ist in diesem Fall am Arbitrationsgraphen bzw. am aktuell aktiven Verhaltensbaustein erkennbar, wenn die Kreuzung bzw. Haltelinie passiert werden soll. Abb. 5.6a und 5.6b vergleichen zur Veranschaulichung *Folgefahrt* und *Kreuzung passieren* an derselben Szene.

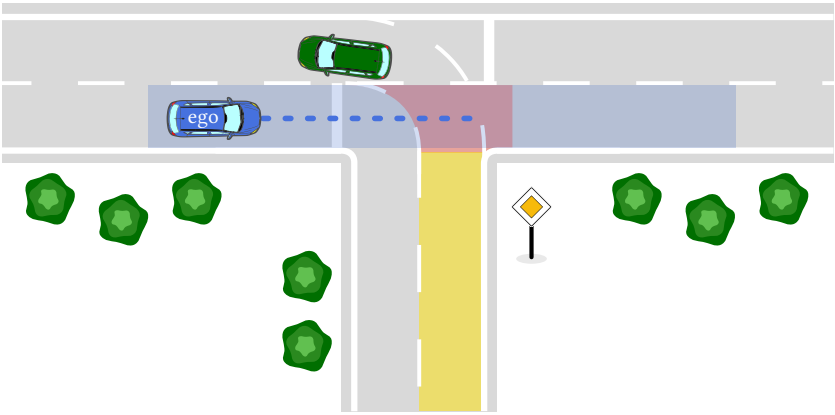
Start-Bedingung

Auf Basis der Planungskarte prüft *Kreuzung passieren*, ob der Ego-Fahrstreifen innerhalb des Planungshorizonts mit anderen Fahrstreifen in Konflikt steht und ob zu erwarten ist, dass vorfahrtsberechtigte Verkehrsteilnehmer die zugehörige Konfliktzone passieren werden. Abbildung 5.6b veranschaulicht eine Beispielszene, wobei die rot markierte Fläche die zu berücksichtigende Konfliktzone kennzeichnet. In Gelb ist außerdem der Bereich markiert, der mindestens einsehbar sein muss, um die Situation sicher einschätzen zu können. Wie groß bzw. lang diese Fläche sein muss, hängt ähnlich zum Ausparken in Abschnitt 5.1.1 von der zulässigen Maximalgeschwindigkeit in diesem Fahrstreifen und der zumutbaren Bremsbeschleunigung ab.

[1] Einschließlich Ein- und Ausfahrten, Fußgängerüberwegen sowie Engstellen.



(a) Die *Folgefahrt* (Abschnitt 5.1.3) hält immer auch an vorfahrtsbezogenen Haltelinien.



(b) Erst wenn der gelb markierte Bereich einsehbar und die rot gefärbte Konfliktzone frei von Prädiktionen vorfahrtsberechtigter Verkehrsteilnehmer ist, kann die Kreuzung sicher und legal passiert werden, sodass *Kreuzung passieren* anwendbar wird.

Abbildung 5.6: *Kreuzung passieren* übernimmt die Entscheidung, ob und wann die Kreuzung passiert werden kann.

Bei mehreren aufeinander folgenden Vorfahrtszenarien betrachtet *Kreuzung passieren* nur die Vorfahrtsituation der nächstliegenden Haltelinie (im Folgenden als aktuelle Haltelinie bezeichnet). So wird eine Haltelinie nach der anderen passiert, ohne Gefahr zu laufen eine nachfolgende Haltelinie zu schnell anzusteuern oder gar unberechtigt zu überqueren.

Start-Bedingung zum Passieren einer Kreuzung

Der Verhaltensbaustein *Kreuzung passieren* kann aufgerufen werden, wenn

- sich das Ego-Fahrzeug innerhalb eines Fahrstreifens der Route befindet,
- sich im vor dem Ego-Fahrzeug gelegenen Fahrkorridor eine Haltelinie mit zugehörigen vorfahrtsberechtigten Fahrstreifen befindet,
- diese Fahrstreifen hinreichend weit einsehbar sind und
- die zugehörigen Konfliktzonen frei von Prädiktionen vorfahrtsberechtigter Verkehrsteilnehmer sind.

Fortsetzungs-Bedingung

Solange kein Verkehrsteilnehmer die Kreuzung unerwartet schnell erreichen könnte, um sein Vorfahrtsrecht doch einzufordern, kann und sollte *Kreuzung passieren* fortgesetzt werden, auch um ein Blockieren der Kreuzung zu vermeiden. Um jedoch ein oszillierendes Verhalten bei Objekten an der Entscheidungsgrenze zu vermeiden, sollte die Entscheidungsgrenze der Fortsetzungs-Bedingung im Stil einer Hysterese näher an der Kreuzung liegen als die der Start-Bedingung.

Fortsetzungs-Bedingung zum Passieren einer Kreuzung

Der Verhaltensbaustein *Kreuzung passieren* sollte fortgesetzt werden, solange

- sich das Ego-Fahrzeug innerhalb eines Fahrstreifens der Route befindet,
- die zur aktuellen Haltelinie zugehörigen vorfahrtsberechtigten Fahrstreifen, unter Berücksichtigung einer Hysterese, hinreichend weit einsehbar sind,
- die aktuelle Haltelinie samt entsprechender Konfliktzonen noch nicht überquert wurden und
- keine der Prädiktionen vorfahrtsberechtigter Verkehrsteilnehmer erwarten lässt, dass ein solcher Verkehrsteilnehmer eine Konfliktzone erreicht bevor das Ego-Fahrzeug sie verlassen hat.

Verhaltensgenerierung

Den Fahrkorridor bestimmt *Kreuzung passieren* wie der Verhaltensbaustein zur Folgefahrt, d. h. durch Aneinanderkettung aufeinander folgender Lanelets, mit dem Unterschied, dass der Korridor anschließend nicht an der aktuellen Haltelinie abgeschnitten wird. An darauf folgenden Stopp- und Haltelinien wird der Korridor wiederum regulär abgeschnitten. Ein resultierender Beispielkorridor wird in Abb. 5.6b einschließlich der zugehörigen Soll-Trajektorie dargestellt.

5.1.5 Gefahrenstelle langsam queren

Nähert sich das Fahrzeug einer Gefahrenstelle ist es angebracht, diese, solange sie nicht vollständig blockiert ist, nur mit reduzierter Geschwindigkeit zu passieren. Als Gefahrenstelle gilt bspw. eine Baustellenausfahrt, ein mit Warndreieck gekennzeichneter Unfallbereich oder auch freie Fußgängerüberwege[1].

Die entsprechende Ziel-Geschwindigkeit ist dabei abhängig von der zulässigen Maximalgeschwindigkeit, den aktuellen Wahrnehmungsgrenzen und der

[1] Solange sich VRUs in der Nähe eines Fußgängerüberweges aufhalten, übernimmt das *Vorfahrt gewähren* Manöver aus Abschnitt 5.1.4.

Prädiktionsunsicherheit. Abgesehen davon unterscheidet sich die Manöverbeschreibung allerdings nicht von der Folgefahrt, sodass der Verhaltensbaustein *Gefahrenstelle langsam queren* als Spezialisierung des Verhaltensbausteins *Folgefahrt* entworfen werden kann. Damit nutzt man die Modularität und Wiederverwendbarkeit der Verhaltensbausteine aus, welche zu den wesentlichen Stärken des Arbitrationsverfahrens zählen. Während *Folgefahrt* eine robuste Basis liefert, können darauf aufbauend beliebige Spezialverhalten implementiert werden, die gezielte Szenarien adressieren. Dies dient u. a. einer besseren Wartbarkeit der Verhaltensbausteine, aber auch einer höheren Transparenz welches Manöver aktuell ausgeführt wird.

Start-Bedingung

Die Start-Bedingung des Verhaltensbausteins *Gefahrenstelle langsam queren* prüft zusätzlich zur Start-Bedingung der Folgefahrt, ob innerhalb des Planungshorizonts des aktuellen Fahrstreifens eine Gefahrenstelle vorliegt:

Start-Bedingung zum Queren einer Gefahrenstelle

Der Verhaltensbaustein zum Queren einer Gefahrenstelle kann aufgerufen werden, sobald

- sich das Ego-Fahrzeug innerhalb eines Fahrstreifens der Route befindet,
- sich eine Gefahrenstelle im Fahrkorridor befindet und
- die Fahrbahn zugleich nicht vollständig blockiert ist bzw.,
- im Falle eines Fußgängerüberweges, keine VRUs im Umfeld sind.

Fortsetzungs-Bedingung

Die Fortsetzungs-Bedingung des *Gefahrenstelle langsam queren* Verhaltensbausteins gleicht seiner Start-Bedingung:

Fortsetzungs-Bedingung zum Queren einer Gefahrenstelle

Der Verhaltensbaustein *Gefahrenstelle langsam queren* kann fortgesetzt werden, solange

- sich das Ego-Fahrzeug innerhalb eines Fahrstreifens der Route befindet,
- sich eine Gefahrenstelle im Fahrkorridor befindet und
- die Fahrbahn zugleich nicht vollständig blockiert ist bzw.,
- im Falle eines Fußgängerüberweges, keine VRUs im Umfeld sind.

Verhaltensgenerierung

Den Fahrkorridor übernimmt *Gefahrenstelle langsam queren* vom Verhaltensbaustein *Folgefahrt*, wobei allerdings die hinterlegte Wunschgeschwindigkeit an der Gefahrenstelle angepasst wird. Anschließend wird die Trajektorie mit dem gleichen Verfahren wie in Abschnitt 5.1.3, also bspw. Nichtlinearer Optimierung, bestimmt. Ggf. kann es außerdem angebracht sein, das Warnblinklicht als HMI zu aktivieren. Abbildung 5.7 veranschaulicht das Verhalten.

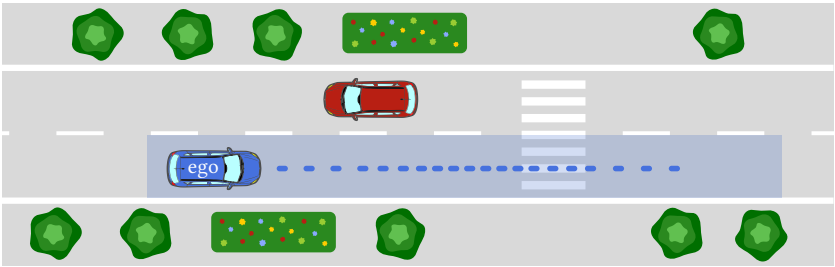


Abbildung 5.7: An potenziellen Gefahrenstellen fährt das Ego-Fahrzeug mit reduzierter Geschwindigkeit.

5.1.6 Fahrstreifenwechsel

Ein weiteres wichtiges grundlegendes Fahrmanöver ist der Fahrstreifenwechsel, der im Laufe einer Fahrt häufig notwendig wird, um der Route zu folgen. Fahrstreifenwechsel können folglich bei Ein- und Ausfahrten, an Kreuzungen zur Wahl der Fahrrichtung, oder wenn der Ego-Fahrstreifen aus sonstigen Gründen endet, durchgeführt werden. Hauptsächlich unterscheidet sich ein Fahrstreifenwechsel von der Folgefahrt also im Querverhalten, berücksichtigt allerdings im Längsverhalten auch Verkehrsteilnehmer auf dem Zielfahrstreifen. Weist letzterer eine ausreichend große Lücke zum Einscheren auf, kann der konservative Fahrstreifenwechsel vollzogen werden.

Bei dichtem Kolonnenverkehr oder auch dem sog. Reißverschlussverfahren wäre ein konservativer Fahrstreifenwechsel hingegen nicht möglich. Hierfür müsste ein weiterer Verhaltensbaustein entworfen werden, der die in diesem Falle nicht zu vernachlässigende Interaktion zwischen den Verkehrsteilnehmern berücksichtigt. An dieser Stelle beschränkt sich diese Arbeit jedoch auf konservative Fahrstreifenwechsel und verweist hinsichtlich des Einordnens in dichtem Verkehr u. a. auf den POMDP Ansatz aus [Hub18b] sowie auf [Bou19].

Der Verhaltensbaustein *Fahrstreifenwechsel* wird zunächst richtungsunabhängig definiert und in der Anwendung jeweils einmal als *Fahrstreifenwechsel links* und *Fahrstreifenwechsel rechts* instanziiert. Die entsprechende Richtung wird im Folgenden als Zielrichtung bezeichnet.

Start-Bedingung

Sofern auf dem Zielfahrstreifen andere Verkehrsteilnehmer sind, ist beim Fahrstreifenwechsel darauf zu achten, dass sowohl zum vorausfahrenden als auch zum nachfolgenden Objekt ein angemessener Abstand gehalten wird. Dieser Abstand setzt sich zum einen aus einem räumlichen und einem zeitlichen Mindestabstand zusammen, die jeweils für den vorausfahrenden bzw. nachfolgenden Verkehrsteilnehmer unterschiedlich festgelegt werden können. Diese Voraussetzungen werden in der Start-Bedingung überprüft:

Start-Bedingung zum Fahrstreifenwechsel

Der Verhaltensbaustein *Fahrstreifenwechsel* kann aufgerufen werden, wenn

- sich das Ego-Fahrzeug innerhalb eines Fahrstreifens der Route befindet,
- in Zielrichtung ein dazu benachbarter Fahrstreifen existiert,
- ein Fahrstreifenwechsel auf diesen Zielfahrstreifen erlaubt ist und
- ggf. auf diesem Fahrstreifen vorhandene Verkehrsteilnehmer in ausreichendem räumlichen sowie zeitlichen Abstand zum Ego-Fahrzeug sind.

Fortsetzungs-Bedingung

Zweck des Fahrstreifenwechsels ist es, das Ego-Fahrzeug auf den Zielfahrstreifen zu bringen, sodass der Verhaltensbaustein so lange fortgesetzt werden sollte bis das Ego-Fahrzeug vollständig auf dem Zielfahrstreifen angelangt ist. Gelingt dies nicht, bspw. weil die angepeilte Lücke doch zu eng wird, muss der Verhaltensbaustein *Fahrstreifenwechsel* den Vorgang kontrolliert abbrechen und das Ego-Fahrzeug vollständig auf den Startfahrstreifen bringen.

Fortsetzungs-Bedingung des Fahrstreifenwechsels

Der Fahrstreifenwechsel sollte fortgesetzt werden, solange

- sich das Ego-Fahrzeug innerhalb des Start- und/oder Zielfahrstreifens befindet,
- ggf. auf dem Zielfahrstreifen vorhandene Verkehrsteilnehmer in ausreichendem räumlichen sowie zeitlichen Abstand zum Ego-Fahrzeug sind und
- das Ego-Fahrzeug den Zielfahrstreifen, bzw. im Falle eines Abbruchs den Startfahrstreifen, vollständig erreicht hat.

Verhaltensgenerierung

Der Fahrkorridor zum Fahrstreifenwechsel besteht im naiven Ansatz aus drei aufeinander folgenden Segmenten: Zunächst folgt er ausschließlich dem

Startfahrstreifen, dann umfasst das Segment des eigentlichen Fahrstreifenwechsels beide Fahrstreifen und zuletzt folgt der Korridor nur noch dem Zielfahrstreifen. In einer solchen Implementierung würden allerdings die Korridorgrenzen und infolgedessen auch die Mittellinie am Anfang und Ende des mittleren Segments einen Sprung aufweisen. Da Trajektorienplaner aber häufig ebenjene Mittellinie als Referenz nutzen, würde diese Unstetigkeit die Konvergenz des Planers beeinträchtigen oder gar verhindern. Daher werden, wie in Abb. 5.8 veranschaulicht, zwei zusätzliche Segmente am Anfang und Ende des mittleren Korridorsegments eingefügt, die mittels Interpolation einen kontinuierlichen Übergang der Korridorgrenzen und somit auch der Mittellinie sicherstellen.

Anschließend wird, wie auch bei den anderen korridorbasierten Fahrmanövern, innerhalb dieses Fahrkorridors eine Trajektorie bspw. mittels Nichtlinearer Optimierung geplant. Dabei sind die Längen der einzelnen Segmente parametrisierbar, um sie je nach Anwendung und verwendetem Trajektorienplaner optimal aufeinander abstimmen zu können. Außerdem werden auch beim Fahrstreifenwechsel Stopp- und Haltelinien berücksichtigt, um bspw. eine rote Ampel des Zielfahrstreifens nicht zu schnell anzufahren.

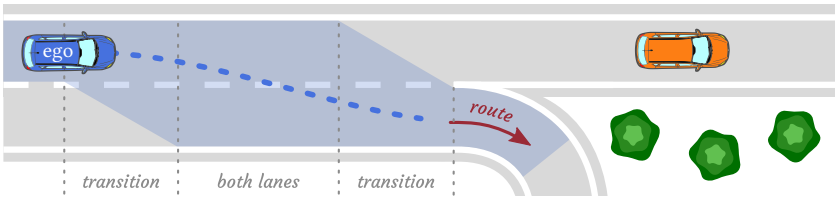


Abbildung 5.8: Der Fahrkorridor des Fahrstreifenwechsels unterstützt die Trajektorienplanung durch einen kontinuierlichen Übergang der Fahrkorridorgrenzen von Start- zu Zielfahrstreifen.

5.2 Fahrmanöver der Rückfallebenen

Für eine robuste und sichere Verhaltensarbitration sind die Verhaltensbausteine der Rückfallebene von wesentlicher Bedeutung. Sie übernehmen die Fahrfunktion, wenn die grundlegenden Verhaltensbausteine ausfallen. Um allerdings auch auf der Rückfallebene ein möglichst robustes und sicheres Verhalten zu gewährleisten, werden auch ihre Verhaltensbausteine der Verifikation unterzogen. Lediglich bei einem Verhaltensbaustein, dem *Nothalt*, wird das Verifikationsergebnis nicht berücksichtigt, um jederzeit mind. ein MRM zu garantieren. Dieser Abschnitt beschreibt die in dieser Arbeit genutzten Rückfallebenen.

5.2.1 Letztes Manöver fortsetzen

Das optimistischste Rückfallverhalten *Letztes Manöver fortsetzen* gibt die im vorigen Zeitintervall ausgeführte Soll-Trajektorie zurück, solange sie noch realisierbar und nicht zu alt ist. Die Berücksichtigung des Manöveralters, also der Dauer seitdem es geplant wurde, verhindert dabei sich zu lange auf einen alten Stand des Umweltmodells zu verlassen. Zudem unterliegt *Letztes Manöver fortsetzen* der Stellgrößen-Verifikation aus Abschnitt 4.2, was ihre Gültigkeit, Realisierbarkeit und Sicherheit gewährleistet.

Start-Bedingung zum Fortsetzen der letzten Soll-Trajektorie

Der Verhaltensbaustein *Letztes Manöver fortsetzen* kann aufgerufen werden, wenn

- im vorigen Zeitintervall bereits ein Manöver ausgeführt wurde,
- dieses Manöver nicht älter als ein festgelegtes Maximalalter ist und
- die zugehörige Soll-Trajektorie noch realisierbar ist.

Fortsetzungs-Bedingung zum Fortsetzen der letzten Soll-Trajektorie

Der Verhaltensbaustein *Letztes Manöver fortsetzen* kann fortgesetzt werden, wenn

- der Verhaltensbaustein bereits aktiv ist,
- das im vorigen Zeitintervall ausgeführte Manöver nicht älter als ein festgelegtes Maximalalter ist und
- die zugehörige Soll-Trajektorie noch realisierbar ist.

Die Verhaltensgenerierung beschränkt sich für diesen Verhaltensbaustein zunächst darauf, die Soll-Trajektorie $\boldsymbol{x}^{\kappa-1}$, die HMI-Ausgaben $m_{\mathrm{HMI}}^{\kappa-1}$ und die V2X-Nachrichten $m_{\mathrm{V2X}}^{\kappa-1}$ des Manövers aus dem vorigen Planungsintervall $\kappa-1$ zu übernehmen. Wichtig ist allerdings eine neue Fail-Safe-Trajektorie $\check{\boldsymbol{x}}^{\kappa}$ zu bestimmen, um sicherzustellen, dass diese – wie auch in Abschnitt 3.2.4 und Gleichung (3.2b) diskutiert – im nächsten Zeitintervall noch realisierbar ist. Hierzu muss sie in den ersten drei geplanten Posen mit der Soll-Trajektorie übereinstimmen:

$$\boldsymbol{x}^{\kappa} = \boldsymbol{x}^{\kappa-1} \tag{5.2a}$$

$$\check{\boldsymbol{x}}_l^{\kappa} = \boldsymbol{x}_l^{\kappa} \quad \forall l \in [0,2]. \tag{5.2b}$$

5.2.2 Fail-Safe-Trajektorie

Die zweite Rückfallebene knüpft unmittelbar an die im Rahmen des Sicherheitskonzepts formulierte Forderung an, dass jeder Verhaltensbaustein zusätzlich zur Soll-Trajektorie auch eine ausfallsichere Alternativ-Trajektorie, die sog. Fail-Safe-Trajektorie, bestimmen muss (Abschnitt 3.2.4). Diese Fail-Safe-Trajektorie muss u. a. der Verifikation bzgl. Verkehrssicherheit standhalten. Entsprechend kann die Rückfallebene *Fail-Safe* auf die, im vorigen Planungsintervall $\kappa-1$ geplante, Fail-Safe-Trajektorie $\check{\boldsymbol{x}}^{\kappa-1}$ zurückgreifen und direkt als ihre Soll-Trajektorie ausgeben.

Auch *Fail-Safe* wird vom übergeordneten Arbitrator der Verifikation unterzogen, um sicherzustellen, dass diese Fail-Safe-Trajektorie nur dann ausgeführt wird, solange die Annahmen ihrer Verifikation weiterhin erfüllt sind. Daher beschränkt sich die Start-Bedingung lediglich darauf, zu prüfen, ob überhaupt

eine Fail-Safe-Trajektorie $\check{\boldsymbol{x}}^{\kappa-1}$ vorliegt. Ob sie weiterhin gültig, realisierbar und sicher ist, wird hingegen im Verifikationsschritt geprüft.

Start-Bedingung zum Fail-Safe Verhalten

Der Verhaltensbaustein *Fail-Safe* kann aufgerufen werden, wenn

- im vorigen Zeitintervall bereits ein Manöver ausgeführt wurde.

Da der Verhaltensbaustein *Fail-Safe* in jedem Fall die vorige Fail-Safe-Trajektorie $\check{\boldsymbol{x}}^{\kappa-1}$ als neue Fail-Safe-Trajektorie $\check{\boldsymbol{x}}^{\kappa}$ zurückgibt, kann er anschließend durchgehend fortgesetzt werden:

Fortsetzungs-Bedingung zum Fail-Safe Verhalten

Der Verhaltensbaustein *Fail-Safe* kann immer dann fortgesetzt werden, wenn

- der Verhaltensbaustein bereits aktiv ist.

Zusammenfassend übernimmt *Fail-Safe* in seiner `Aktion`-Funktion die im vorigen Planungsintervall $\kappa - 1$ geplante Fail-Safe-Trajektorie $\check{\boldsymbol{x}}^{\kappa-1}$ als aktuelle Soll- und Fail-Safe-Trajektorie:

$$\boldsymbol{x}^{\kappa} = \check{\boldsymbol{x}}^{\kappa-1} \tag{5.3a}$$

$$\check{\boldsymbol{x}}^{\kappa} = \check{\boldsymbol{x}}^{\kappa-1} \tag{5.3b}$$

Zudem kann es, je nach Gefährdungslage, sinnvoll sein, das Warnblinklicht über die HMI-Ausgaben $m^{\kappa}_{\mathrm{HMI}}$ zu aktivieren.

5.2.3 Nothalt

Als dritte Rückfallebene stellt der Verhaltensbaustein *Nothalt* eine Vollbremsung auf geradem Pfad zur Verfügung. Da er zugleich die letzte Rückfallebene bildet, ist es entscheidend, dass der *Nothalt* in jeder Situation aufgerufen werden kann. Für die Trajektorienplanung in Kartesischen oder Frenet-Koordinaten muss jedoch zumindest der aktuelle Fahrzeugzustand $\boldsymbol{x}$ (bspw. durch ein Odometer) bekannt sein:

Start-Bedingung zum Nothalt

Der Verhaltensbaustein zum Nothalt kann jederzeit aufgerufen werden, solange der aktuelle Fahrzeugzustand $\boldsymbol{x}$ bekannt ist. Der Nothalt kann insbesondere auch dann aufgerufen werden, wenn u. a.

- das Ego-Fahrzeug die Strecke verlassen hat,
- die Planungskarte nicht mehr verfügbar ist oder
- Teile der Wahrnehmung, Lokalisierung oder Prädiktion ausgefallen sind.

Steht auch der Fahrzeugzustand $\boldsymbol{x}$ nicht zur Verfügung, weil z. B. sogar die Fahrzeugodometrie ausgefallen ist, ist es Aufgabe der Regelung diesen Ausfall zu erkennen und direkt über die Aktorstellgrößen (Soll-Beschleunigung und -Lenkwinkel) eine Vollbremsung zu initiieren. Alternativ müssten die Verhaltensbausteine ihre Soll- und Fail-Safe-Trajektorien als zeitlichen Verlauf der Aktorstellgrößen modellieren. Diese könnten zwar ohne die Kenntnis des aktuellen Fahrzeugzustands geplant werden. Allerdings kann ohne diesen nicht gewährleistet werden, dass die Trajektorien physikalisch realisierbar sind.

Fortsetzungs-Bedingung zum Nothalt Verhalten

Der Verhaltensbaustein *Nothalt* kann immer dann fortgesetzt werden, wenn

- der Verhaltensbaustein bereits aktiv ist.

In seiner `Aktion`-Funktion bestimmt der Verhaltensbaustein *Nothalt* eine Trajektorie, die ausgehend vom aktuellen Zustand $\boldsymbol{x}$ das Fahrzeug bei konstanter Orientierung mit maximal realisierbarer Bremsbeschleunigung bis zum Stillstand bringt. Aufgrund der Havarie wird zudem das Warnblinklicht über die HMI-Ausgaben $m^{\kappa}_{\mathrm{HMI}}$ aktiviert.

6 Evaluation und Validierung

Dieses Kapitel stellt die Validierung und Evaluation der im Rahmen dieser Arbeit entworfenen Verhaltensarbitration vor. Abschnitt 6.1 beschreibt dabei zunächst die Validierung der Verhaltensgenerierung mittels Arbitrationsgraphen im Realversuch. Anschließend werden in Abschnitt 6.2 die Erweiterungen aus Kapitel 4 hin zur robusten und sicheren Verhaltensarbitration in Simulation evaluiert.

6.1 Validierung im Realversuch

Abbildung 6.1: Arbitrationsgraphen übernehmen seit Sommer 2020 die Verhaltensentscheidung des Versuchsfahrzeugs *Bertha*.

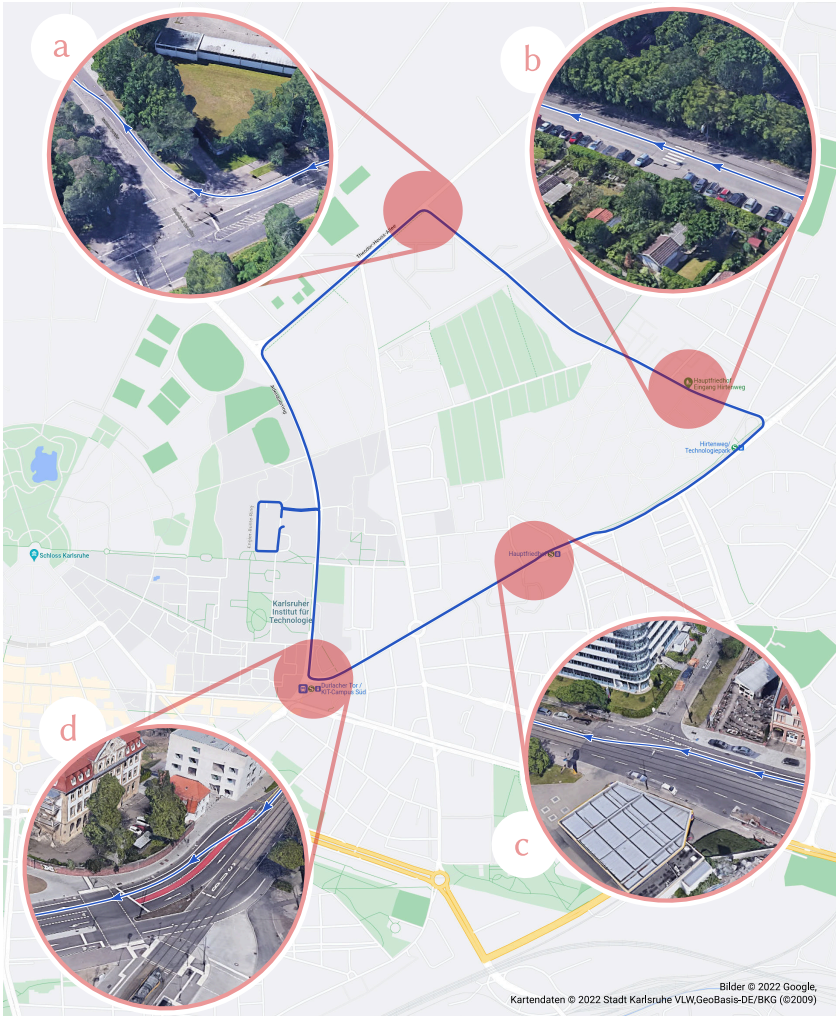


Abbildung 6.2: Die 6 km lange Teststrecke umfasst urbane sowie landstraßenähnliche Etappen. **(a)** An der Kreuzung der Theodor-Heuss-Allee mit der Rintheimer Querallee sind zwei Fahrstreifenwechsel notwendig. **(b)** Auf der Rintheimer Querallee ist ein Fußgängerüberweg mit „Zebrastreifen“ zu beachten. **(c)** Die Strecke verläuft ab dem Hauptfriedhof auf den Straßenbahngleisen. **(d)** An der Kreuzung Durlacher Tor muss u. a. ein parallel verlaufender Radweg gekreuzt werden.

Eine frühe Implementierung der Verhaltensarbitration für automatisierte Fahrzeuge wurde auf dem Versuchsfahrzeug *Bertha* [Taş18b], dargestellt in Abb. 6.1, integriert und ist seit Sommer 2020 in regelmäßigen Testfahrten im Straßenverkehr im Einsatz [Orz20]. Sie hat den bisherigen Endlichen Zustandsautomaten ersetzt und vereinfacht seitdem die Entwicklung und Integration neuer Manöver in Form weiterer Verhaltensbausteine. Der Ansatz im Fahrzeug nutzt eine korridorbasierte Manöverrepräsentation, die im Zuge dieser Arbeit (Abschnitt 3.2.2) aber auf Soll- und Fail-Safe-Trajektorien geändert wurde, um die in Abschnitt 4.2 eingeführte Verifikation zu ermöglichen.

6.1.1 Teststrecke

Die in Abb. 6.2 veranschaulichte Hauptversuchsstrecke beginnt auf dem Campus des Karlsruher Instituts für Technologie (KIT) und erstreckt sich über 6 km entlang urbaner sowie landstraßenähnlicher Etappen, bevor sie wieder am Ausgangspunkt endet. Die Strecke beinhaltet einige herausfordernde Passagen: Sie passiert zwölf Kreuzungen (u. a. Abb. 6.2a und 6.2d), eine Stelle mit Reißverschlussverfahren, einen Fußgängerüberweg mit „Zebrastreifen“ (Abb. 6.2b), drei Fußgängerüberwege mit Lichtsignalanlage und zwei Verkehrsinseln. Zudem kreuzt sie einen parallel verlaufenden Radweg vor der Kreuzung am Durlacher Tor (Abb. 6.2d). Eine weitere Herausforderung ergibt sich im Bereich der Haid-und-Neu-Straße (Abb. 6.2c) zwischen Hauptfriedhof und Durlacher Tor, da sich dort Straßenbahn und Fahrzeuge denselben Fahrstreifen teilen. Zusätzlich sind entlang der ganzen Route mehrere Fahrstreifenwechsel notwendig, um der Streckenführung zu folgen.

6.1.2 Arbitrationsgraph

Abbildung 6.3 stellt die im Realversuch verwendeten Verhaltensbausteine des Arbitrationsgraphen dar. Den überwiegenden Anteil der Fahraufgabe übernimmt der Verhaltensbaustein *Follow Lane*, der die Folgefahrt samt Haltemanövern an Ampeln und Vorfahrtsituationen realisiert. Die Freigabe der Haltelinien erfolgte aus Sicherheitsgründen jedoch noch manuell. Die beiden *Change Lane* Verhaltensbausteine setzen jeweils Fahrstreifenwechsel nach links

bzw. rechts um. Diese berücksichtigen, wie in Abschnitt 5.1.6 diskutiert, auch Stopp- und Haltelinien des Zielfahrstreifens, um ein vorausschauendes Verhalten zu erzeugen. Der Kosten-Arbitrator Urban Driving wählt zwischen diesen drei Verhaltensbausteinen das Manöver, dessen langfristige Routingkosten minimal sind[1]. In der Nähe des angesteuerten Parkplatzes wird schließlich das Einparkmanöver *Park Near Goal* anwendbar und nach Priorität aktiviert. Als Beispiel, wie das Sicherheitsziel S1 (ein MRM bei Verlassen der ODDs zu garantieren) umgesetzt wurde, wird die Rückfallebene *Emergency Stop* nach Abschluss des Einparkmanövers aktiv und hält das Fahrzeug im Stillstand.

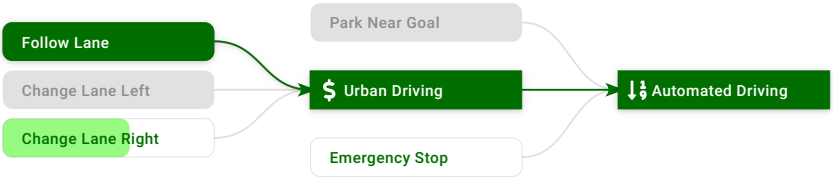


Abbildung 6.3: Arbitrationsgraph zur Verhaltensentscheidung für automatisiertes Fahren mit dem Versuchsfahrzeug *Bertha*. Die Breite eines hellgrünen Hintergrundes stellt den normierten Nutzen einer anwendbaren Option dar. Je höher der Nutzen, desto größer die hellgrüne Fläche.

6.2 Simulative Evaluation

In diesem Abschnitt wird die in Kapitel 3 überarbeitete Manöverrepräsentation und Verhaltensgenerierung sowie die Erweiterung der Verhaltensarbitration um das in Kapitel 4 eingeführte Sicherheitskonzept evaluiert. Dabei wird die Implementierung insbesondere hinsichtlich der Umsetzung folgender Sicherheitsziele untersucht:

S5 Das System verhindert ungültige Stellgrößen.

S6 Das System vermeidet riskante Stellgrößen.

[1] Alternativ zur Minimierung von Kosten kann auch der Nutzen maximiert werden.

Die folgenden beiden Abschnitte stellen zunächst die Testumgebung und den Arbitrationsgraph des Referenzverhaltens samt der verwendeten Verhaltensgenerierung vor. Die Abschnitte 6.2.3 und 6.2.4 präsentieren anschließend den jeweiligen Evaluationsansatz hinsichtlich der Sicherheitsziele S5 und S6 sowie die entsprechenden Ergebnisse.

6.2.1 Testumgebung

Für die simulative Evaluation wird dieselbe Teststrecke wie im Realversuch verwendet (Abschnitt 6.1.1). Als Simulationsumgebung kommt dabei CoInCar-Sim zum Einsatz [Nau18]. Wesentliche Vorteile dieser Simulationsumgebung sind der Multi-Agenten-Ansatz, der es ermöglicht die Interaktion zwischen Verkehrsteilnehmern zu simulieren, sowie die ROS und lanelet2 Schnittstellen, die den Schnittstellen des Versuchsfahrzeugs *Bertha* gleichen (Abb. 6.1). Somit kann eine in CoInCar-Sim entworfene Verhaltensgenerierung mit nur geringem Aufwand auf das Versuchsfahrzeug übertragen und dort ausgeführt werden. Genauso wird sie umgekehrt genutzt, um Fehler der Verhaltensgenerierung des Versuchsfahrzeugs in Simulation zu rekonstruieren, gefahrlos zu testen und zu beheben.

CoInCar-Sim schafft zwar die Voraussetzungen für interaktive Verhaltenssimulationen, stellt allerdings selbst nur nicht-interaktive Verhaltensmodelle bereit, die lediglich vorab definierten Trajektorien folgen. Um daher eine realistischere Simulation zu erzielen, wird das vorhandene Verhaltensmodell anderer Verkehrsteilnehmer so umgestaltet, dass diese auch auf das Ego-Fahrzeug reagieren. Hierzu wird die in dieser Arbeit vorgestellte Verhaltensarbitration nicht nur für das Ego-Fahrzeug, sondern auch im Verhaltensmodell aller anderen Verkehrsteilnehmer eingesetzt. Dabei können sich Parameter, wie bspw. Wunschgeschwindigkeiten oder angestrebte Zeitabstände, von Fahrzeug zu Fahrzeug unterscheiden. Die übrigen CoInCar-Sim Module, u. a. die Simulation der Sensorik und Aktorik sowie das Zeitmanagement, bleiben jedoch unberührt. Die in den Abschnitten 6.2.3 und 6.2.4 zur Evaluation entworfenen Stresstests und Modifikationen werden zudem nur in die Verhaltensarbitration des Ego-Fahrzeugs eingebettet.

6.2.2 Implementierung

Die Implementierung setzt die in Kapitel 3 beschriebene Verhaltensarbitration für automatisierte Fahrzeuge samt des Umweltmodells und der überarbeiteten Manöverrepräsentation in C++ und ROS um. Die Verhaltensbausteine sind zudem an die in Kapitel 5 vorgestellten Fahrmanöver angelehnt. Zur Verhaltensgenerierung der einzelnen Verhaltensbausteine wird also für die Soll-Trajektorie zunächst, wie in Kapitel 5 und [Orz20] erläutert, ein Fahrkorridor bestimmt. Um den Fokus der Evaluation auf die Verhaltensentscheidung samt des Verifikationsverfahrens zu legen und hierbei ungewollte Einflüsse der Trajektorienplanung auszuschließen, wird für letztere ein vereinfachtes, deterministisches Verfahren eingesetzt: Hierzu wird die Mittellinie des Korridors mit Splines interpoliert und, unter Anwendung des IDMs, eine Trajektorie entlang dieses Referenzpfades für einen Planungshorizont von 8 s bestimmt. Zuletzt wird die Fail-Safe-Trajektorie im Zeitintervall $[t_k, t_{k+2}]$, wie in Abschnitt 3.2.4 insbesondere bzgl. Gleichung (3.2b) diskutiert, der Soll-Trajektorie gleichgesetzt und ab t_{k+2} als Vollbremsung entlang dieser Soll-Trajektorie bestimmt.

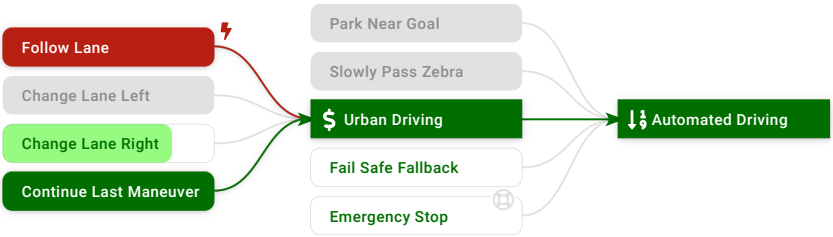


Abbildung 6.4: Beispielzustand Arbitrationsgraphen aus der Evaluation mit den drei Rückfallebenen *Continue Last Maneuver*, *Fail Safe Fallback* und *Emergency Stop*. Die Verhaltensoption *Follow Lane* hat die Verifikation nicht bestanden, sodass alternativ *Continue Last Maneuver* ausgewählt wird. Das *Change Lane Right* Verhalten ist zwar ebenfalls anwendbar, hat allerdings einen geringeren Nutzen.

Zur Umsetzung des in Kapitel 4 eingeführten Sicherheitskonzepts werden die drei Verifikatoren aus Gleichung (4.3) eingesetzt. Wie in Abb. 6.4 dargestellt,

wurden die drei Rückfallebenen aus Abschnitt 5.2 umgesetzt: Zunächst verhindert *Continue Last Maneuver*, bei kurzweiligen Fehlern eines der Verhaltensbausteine innerhalb des Kosten-Arbitrators Urban Driving, eine Oszillation zwischen mehreren Verhaltensoptionen. Hierfür übernimmt *Continue Last Maneuver* die Soll-Trajektorie des im vorigen Zeitintervall bestimmten Manövers, falls sie weiterhin sicher ausführbar und nicht zu alt ist. Außerdem wird die Fail-Safe-Trajektorie aktualisiert, um Gleichung (3.2b) wieder zu erfüllen. Beispielhaft konnte in Abb. 6.4 *Follow Lane* nicht der Verifikation standhalten. Statt einen zwar möglichen, allerdings nicht zielführenden Fahrstreifenwechsel einzuleiten, wird *Continue Last Maneuver* ausgewählt und hält das Fahrzeug mit der zuvor geplanten Trajektorie im eigenen Fahrstreifen.

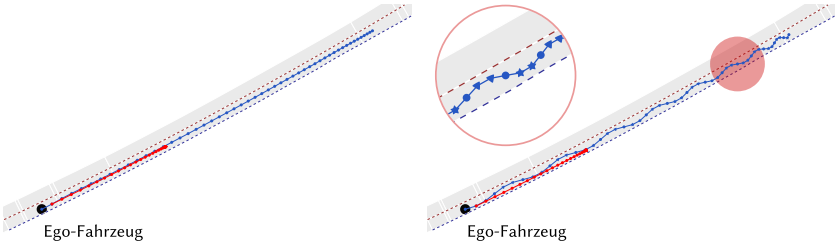
Als zweite Rückfallebene kommt *Fail Safe Fallback* zum Tragen, falls keiner der regulären Verhaltensbausteine sicher anwendbar und das letzte Manöver zu alt ist, um es mit *Continue Last Maneuver* auszuführen. Wie bereits der Name verdeutlicht, führt *Fail Safe Fallback* die Fail-Safe-Trajektorie des zuletzt erfolgreich verifizierten Manövers aus. Es sei angemerkt, dass auch diese Trajektorie vom übergeordneten Prioritäts-Arbitrator Automated Driving in jedem Planungsintervall verifiziert wird.

Sollte schließlich eine der zugrunde liegenden Annahmen, bspw. durch Regelverstöße anderer Verkehrsteilnehmer verletzt werden, kann auch die Verifikation von *Fail Safe Fallback* fehlschlagen. In diesem Fall übernimmt die letzte und restriktivste Rückfallebene *Emergency Stop* und führt einen sofortigen Nothalt aus. *Emergency Stop* wird als einziger Verhaltensbaustein keiner Verifikation unterzogen, um in jeder Situation ein MRM zur Verfügung zu stellen (entspricht dem Sicherheitsziel S2), auch wenn es in dem Moment ggf. nicht den strengen Anforderungen der Erreichbare-Mengen-Analyse genügt.

Die folgenden Abschnitte untersuchen, welchen Einfluss ungültige sowie riskante Soll-Trajektorien auf die Verhaltensgenerierung mit und ohne Sicherheitskonzept haben. Die soeben beschriebene Implementierung, die die Verifikation im Arbitrationsverfahren nutzt, wird dabei als *Sichere Verhaltensarbitration* bezeichnet. Der Begriff *Optimistische Verhaltensentscheidung* bezieht sich hingegen auf eine Verhaltensarbitration ohne Sicherheitskonzept, vergleichbar mit der aus [Orz20].

6.2.3 Valide Verhaltensentscheidung

In diesem Abschnitt wird die Verhaltensarbitration hinsichtlich des Sicherheitsziels S5 untersucht, also ob und wie sie ungültige Stellgrößen abfängt und in dem Fall auf Alternativoptionen zurückgreift. Hierbei ist zu erwarten, dass die *Optimistische Verhaltensentscheidung* ungültige Soll-Trajektorien ungeprüft durch den Arbitrationsgraph durchreicht und als Ergebnis der Verhaltensentscheidung ausgibt. Dies kann – je nachdem, wie schwerwiegend die Soll-Trajektorie beschädigt ist – zu instabilem und somit gefährlichem Fahrverhalten führen. Bei der *Sicheren Verhaltensarbitration* sollte der Verifikationsschritt aus Abschnitt 4.2.3 ungültige Trajektorien hingegen mittels $\mathcal{V}_{\text{gültig}}$ und $\mathcal{V}_{\text{realisierbar}}$ erkennen. Entsprechend ist zu erwarten, dass die Arbitration in dem Fall einen anderen Verhaltensbaustein auswählt und weiterhin gültige und realisierbare Trajektorien ausgibt. Außerdem sollte das Fahrzeug die gesamte Route absolvieren können, solange die regulären[1] Verhaltensbausteine nicht dauerhaft ungültige Stellgrößen ausgeben.



(a) Regulär geplantes Manöver von *Follow Lane*. Soll-Trajektorie in Blau, Fail-Safe-Trajektorie in Rot.

(b) Künstlich beschädigte Soll-Trajektorie mit u. a. zu hoher Krümmung. Dreiecke markieren nach links versetzte Stützpunkte, Sterne wurden nach rechts versetzt, Kreise blieben unberührt.

Abbildung 6.5: Die Soll-Trajektorie wird in zufälligen Planungsintervallen künstlich beschädigt, um den Verifikations- und Rückfall-Mechanismus der Verhaltensarbitration auszulösen.

[1] Reguläre Verhaltensbausteine sind solche, die nicht Teil der Rückfallebenen sind.

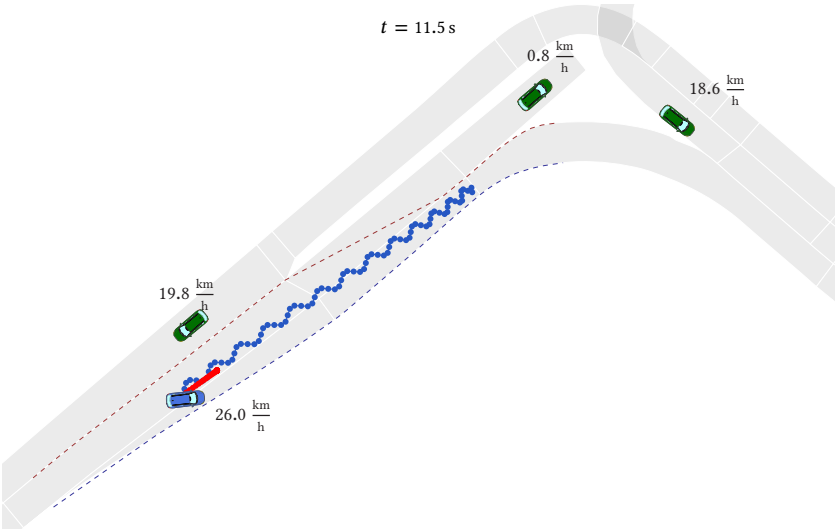
Unter den idealisierten Bedingungen einer Simulation führt die Implementierung der Verhaltensbausteine nicht ohne weiteres zu gravierenden Fehlern, wie ungültigen oder nicht realisierbaren Soll-Trajektorien. Daher werden die Soll-Trajektorien der regulären Verhaltensbausteine in diesem Teil kontrolliert beschädigt, um die Fehlerkompensation auszulösen und die Wirkweise der Rückfallebenen untersuchen zu können. Die Soll-Trajektorien $\boldsymbol{x}_i^{\kappa}$ werden mit festgelegter Fehlerwahrscheinlichkeit $\mathrm{p}(\mathrm{ungültig}_i)$ reproduzierbar beschädigt oder gänzlich intakt gelassen. Im Falle der Beschädigung werden 2/3 der Stützpunkte mit einem großen lateralen Sprung versetzt, die Hälfte hiervon jeweils in Fahrtrichtung links bzw. rechts. Infolgedessen weist die resultierende Trajektorie an diesen Stellen u. a. sehr hohe Krümmungswerte auf und ist in kinematischer Hinsicht nicht realisierbar. Abbildung 6.5 vergleicht die regulär geplante Soll-Trajektorie des *Follow Lane* Verhaltensbausteins mit der künstlich beschädigten Variante. Die Evaluation bzgl. Sicherheitsziel S5 besteht nun aus folgenden beiden Teilen:

Stresstest Im Stresstest wird die Verhaltensarbitration in ausgewählten Szenarien einer außergewöhnlich hohen Fehlerwahrscheinlichkeit von $\mathrm{p}(\mathrm{Ausfall}_i) = 50\,\%$ ausgesetzt. Die künstlich versetzten Punkte der Soll-Trajektorie weisen einen lateralen Versatz von 0,5 m auf.

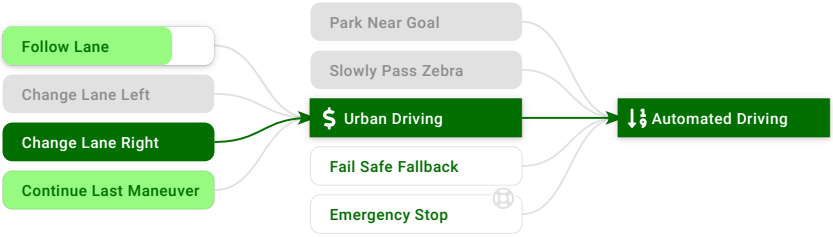
Dauerlauf Im Dauerlauf wird das Systemverhalten entlang der geplanten Route mit einer geringeren Fehlerwahrscheinlichkeit von $\mathrm{p}(\mathrm{Ausfall}_i) = 10\,\%$ und ebenfalls 0,5 m lateralem Versatz der Punkte untersucht.

Ergebnisse *Stresstest Optimistische Verhaltensentscheidung*

Die *Optimistische Verhaltensentscheidung* wurde dem *Stresstest* im Szenario „Kreuzung Theodor-Heuss-Allee mit Rintheimer Querallee“ (vgl. Abb. 6.2a) ausgesetzt. Die Ergebnisse hierzu sind in den Abb. 6.6 und 6.7 dargestellt. Das Ego-Fahrzeug nähert sich zu Beginn, von der Theodor-Heuss-Allee kommend, aus südwestlicher Richtung, auf einem Abschnitt mit einem Fahrstreifen je Fahrtrichtung. Anschließend muss es auf den Abbiegefahrstreifen wechseln, diesem etwa weitere 50 m folgen, um sich dann in den vorfahrtsberechtigten Verkehrsfluss der Rintheimer Querallee einzuordnen.

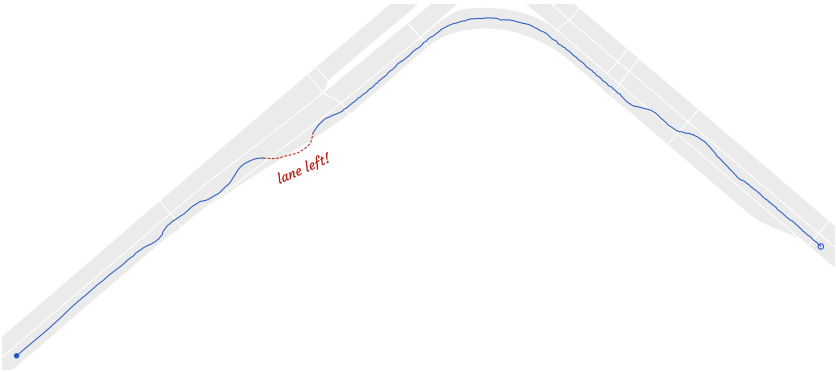


(a) Die *Optimistische Verhaltensentscheidung* führt bei der Arbitration keine Verifikation der Stellgrößen durch, sodass auch die künstlich beschädigten Trajektorien ausgeführt werden. Im *Stresstest* bringen diese das blaue Ego-Fahrzeug in heftiges Überschwingen. Ab dem dargestellten Zeitpunkt verlässt das Ego-Fahrzeug sogar zeitweise den Fahrkorridor (vgl. Abb. 6.7a und 6.7c).

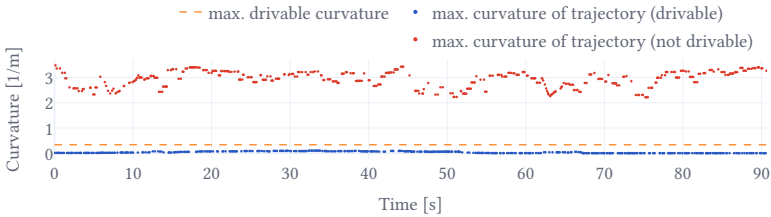


(b) Die *Optimistische Verhaltensentscheidung* führt ihre Verhaltensoptionen ausschließlich auf Basis ihrer Start- und Fortsetzungs-Bedingungen aus. Daher wird zu diesem Zeitpunkt das *Change Lane Right* Verhalten ausgeführt, obwohl es in diesem Moment eine nicht realisierbare Trajektorie ausgibt.

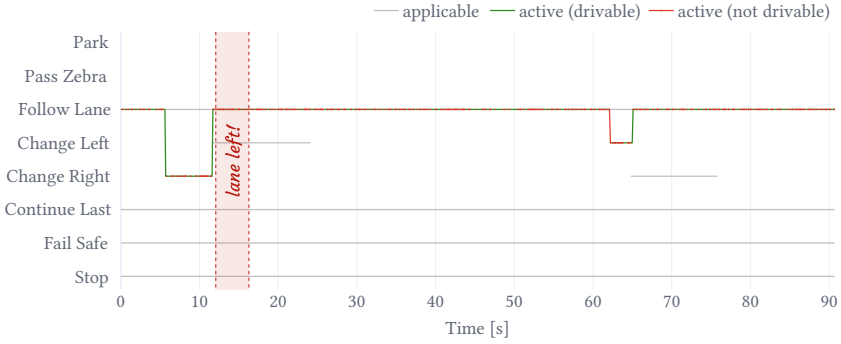
Abbildung 6.6: Beispielsituation der *Optimistischen Verhaltensentscheidung* im *Stresstest* im Szenario „Kreuzung Theodor-Heuss-Allee mit Rintheimer Querallee“ zum Zeitpunkt $t = 11{,}5\,\mathrm{s}$ (bzgl. Szenariostart).



(a) Der vom Ego-Fahrzeug gefahrene Pfad. Aufgrund der künstlich beschädigten Trajektorien, kommt das Ego-Fahrzeug in heftiges Überschwingen und verlässt zeitweise seinen Fahrkorridor (rot gestrichelt markiert).



(b) Maximale Krümmung der zum jeweiligen Zeitpunkt geplanten Trajektorie. Die künstlich beschädigten Trajektorien weisen u. a. zu hohe Krümmungen auf.



(c) Da die *Optimistische Verhaltensarbitration* keine Verifikation durchführt, führt sie die mutmaßlich beste Option auch dann aus, wenn ihre Trajektorie aufgrund zu hoher Krümmungen nicht realisierbar ist.

Abbildung 6.7: Zeitverlauf der *Optimistischen Verhaltensarbitration* im *Stresstest* im Szenario „Kreuzung Theodor-Heuss-Allee mit Rintheimer Querallee“.

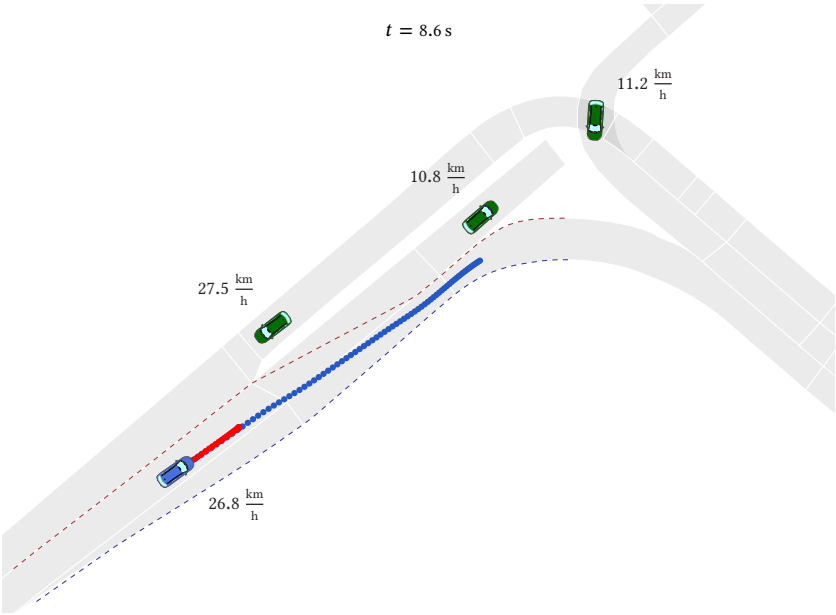
Abbildung 6.6 zeigt dabei die Situation zum Zeitpunkt $t = 11{,}5\,\mathrm{s}$ (bzgl. Szenariostart) in Vogelperspektive sowie den Zustand des Arbitrationsgraphen. Dabei wird deutlich, dass die *Optimistische Verhaltensentscheidung* keine Verifikation durchführt und daher die künstlich beschädigten Trajektorien, in diesem Fall des *Change Lane Right* Verhaltens, ausführt (vgl. Abb. 6.7c).

Dadurch kommt das Ego-Fahrzeug gefährlich ins Schwingen und verlässt sogar zeitweise den Fahrkorridor mit bis zu drei Rädern, wie in Abb. 6.7a hervorgehoben. Außerdem führt die lateral alternierende Trajektorie u. a. wegen ihrer hohen Krümmungswerte in Abb. 6.7b dazu, dass der auf glatte, krümmungs- und ruckarme Trajektorien ausgelegte Längsregler seine Referenzgeschwindigkeit auf durchschnittlich $9{,}9\,\frac{\mathrm{km}}{\mathrm{h}}$ reduziert. Infolgedessen erreicht das Ego-Fahrzeug den Zielfahrstreifen des Szenarios erst nach über 90 s.

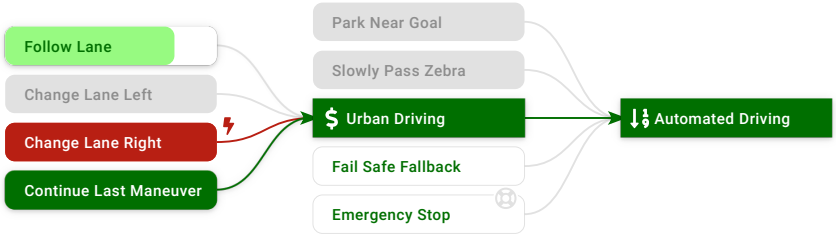
Das beobachtete instabile und riskante Verhalten in diesem *Stresstest* ist eine direkte Folge dessen, dass die *Optimistische Verhaltensentscheidung* die erstbeste anwendbare Option ausführt, ohne dabei die zugehörigen Soll- und Fail-Safe-Trajektorien auf Realisierbarkeit oder Sicherheit hin zu überprüfen. Dadurch hängt die Leistungsfähigkeit der *Optimistischen Verhaltensentscheidung*, insbesondere hinsichtlich der Qualität und Sicherheit des generierten Verhaltens, komplett von der Qualität und Sicherheit der eingesetzten Verhaltensbausteine ab.

Ergebnisse *Stresstest Sichere Verhaltensentscheidung*

Dieser Abschnitt untersucht, ob der Verifikationsschritt der *Sicheren Verhaltensentscheidung* das Ego-Verhalten stabilisieren und absichern kann. Hierzu wird die *Sichere Verhaltensentscheidung* ebenfalls dem *Stresstest* im Szenario des vorigen Abschnitts unterzogen, mit gleichem Anfangszustand des Ego-Fahrzeugs und aller weiteren beteiligten Verkehrsteilnehmer. Auch der Arbitrationsgraph samt verfügbarer Verhaltensoptionen stimmt mit dem der *Optimistischen Verhaltensentscheidung* überein. Der wesentliche Unterschied besteht darin, dass die Arbitratoren der *Sicheren Verhaltensentscheidung* die Soll- und Fail-Safe-Trajektorien einer Verifikation, in diesem Fall hinsichtlich ihrer Gültigkeit und Realisierbarkeit (vgl. Abschnitt 4.2.3), unterziehen. Die

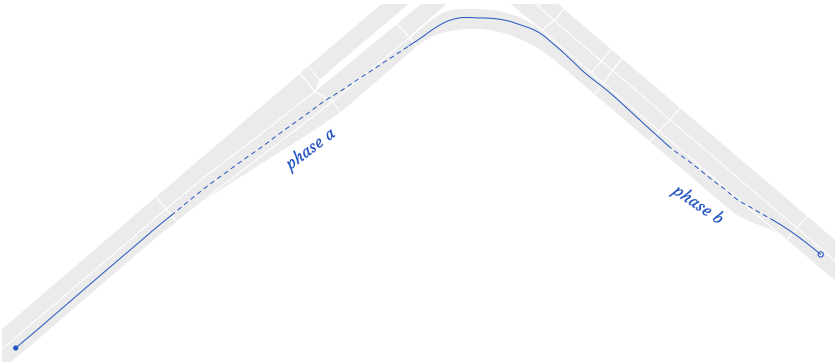


(a) Trotz der Fehlerwahrscheinlichkeit von $\mathrm{p}(\mathrm{Ausfall}_i) = 50\,\%$ hält die *Sichere Verhaltensarbitration* das Ego-Fahrzeug stabil und kollisionsfrei im Fahrkorridor (siehe auch im Gesamtverlauf von Abb. 6.9a).

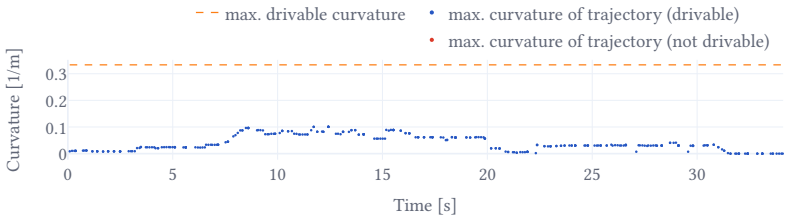


(b) Die *Sichere Verhaltensarbitration* erkennt die beschädigte Soll-Trajektorie des *Change Lane Right* Verhaltens und setzt stattdessen (über das *Continue Last Maneuver* Verhalten) die zuvor ausgeführte Trajektorie fort.

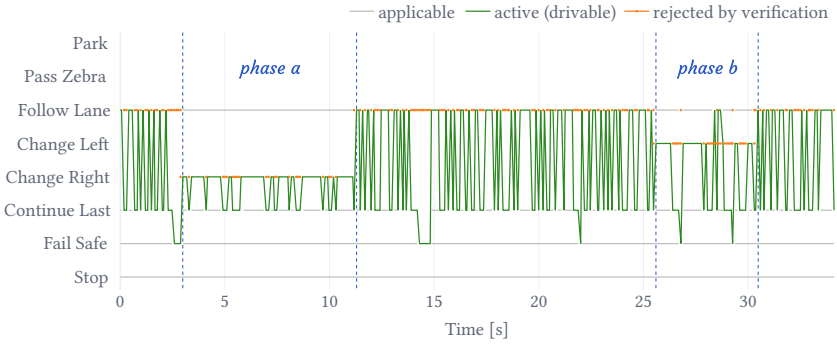
Abbildung 6.8: Beispielsituation der *Sicheren Verhaltensarbitration* im *Stresstest* im Szenario „Kreuzung Theodor-Heuss-Allee mit Rintheimer Querallee“ zum Zeitpunkt $t = 8{,}6\,\mathrm{s}$ (bzgl. Szenariostart).



(a) Gefahrener Ego-Pfad mit hervorgehobenen Abschnitten der Fahrstreifenwechsel (vgl. Abb. 6.9c).



(b) Maximale Krümmung der zum jeweiligen Zeitpunkt geplanten Trajektorie. Dank der Verifikation werden Trajektorien verhindert, die u. a. aufgrund ihrer Krümmung kinematisch nicht realisierbar wären.



(c) Wegen der hohen Ausfallrate greifen die Arbitratoren häufig auf die Rückfallebenen *Continue Last Maneuver* und *Fail Safe Fallback* zurück. Dadurch ergibt sich ein dennoch ruhiges und sicheres Fahrverhalten.

Abbildung 6.9: Zeitverlauf der *Sicheren Verhaltensarbitration* im *Stresstest* im Szenario „Kreuzung Theodor-Heuss-Allee mit Rintheimer Querallee“.

Arbitration wählt also die beste Manöveroption aus, die zugleich die Verifikation mittels $\mathcal{V}_{\text{gültig}}$ und $\mathcal{V}_{\text{realisierbar}}$ erfolgreich absolviert. Die Verifikation bzgl. Verkehrssicherheit wird an dieser Stelle ausgelassen und erst im späteren Abschnitt 6.2.4 untersucht.

Die Ergebnisse der *Sicheren Verhaltensentscheidung* im *Stresstest* sind in den Abb. 6.8 und 6.9 visualisiert. Zunächst ist in Abb. 6.8b zu beobachten, dass die Verifikation im Urban Driving Arbitrator die zum Zeitpunkt $t = 8{,}6\,\text{s}$ künstlich beschädigte Trajektorie des *Change Lane Right* Verhaltens verweigert. Da der Fahrstreifenwechsel auf den Abbiegefahrstreifen in diesem Moment den größten Nutzen hinsichtlich Routingkosten aufweist, fällt der Urban Driving Arbitrator als Nächstes nicht auf die Folgefahrt *Follow Lane* zurück. Stattdessen überprüft er zunächst, ob die bereits im vorigen Planungsintervall ausgeführte Trajektorie der *Continue Last Maneuver* Rückfallebene der Verifikation standhält. Da sich diese Trajektorie als weiterhin gültig und realisierbar erweist, wählt der Urban Driving Arbitrator schließlich *Continue Last Maneuver* aus und setzt somit den Fahrstreifenwechsel mit der Trajektorie des vorigen Planungsintervalls fort.

Im Zeitverlauf des Gesamtszenarios in Abb. 6.9c sind die vielen Ausfälle, in Form künstlich beschädigter und somit nicht realisierbarer Trajektorien, erkennbar. Dabei fällt die *Sichere Verhaltensentscheidung* meist auf das *Continue Last Maneuver* Verhalten zurück. Einige Male, wenn die zuletzt geplante Trajektorie zu alt ist (vgl. Abschnitt 5.2.1), wird die Fail-Safe-Trajektorie des letzten Manövers ausgeführt. Solange allerdings eine der regulären Alternativoptionen ausführbar ist, wird diese statt der niedriger priorisierten *Fail Safe Fallback* Option ausgewählt. Bspw. ist zum Zeitpunkt $t = 26{,}8\,\text{s}$, nach einem längeren Ausfall des bzgl. Routingkosten besten Verhaltens *Change Lane Left*, die Rückfalloption *Continue Last Maneuver* nicht mehr anwendbar. Da die Folgefahrt in diesem Augenblick ebenfalls nicht realisierbar ist, wird der *Fail Safe Fallback* ausgeführt. Sobald allerdings in den Zeitpunkten $t = \{28{,}4\,\text{s}, 28{,}6\,\text{s}$ und $28{,}7\,\text{s}\}$ *Follow Lane* als weitere Alternative zur Verfügung steht, wird die komfortablere Folgefahrt statt der Fail-Safe-Trajektorie ausgeführt.

Die *Sichere Verhaltensentscheidung* kann schließlich dank der Rückfallebenen *Continue Last Maneuver* und *Fail Safe Fallback* sowie in seltenen Fällen auch

dank der Diversität[1] ein ruhiges Fahrverhalten ohne schwingende Trajektorie realisieren. Die Krümmungen der ausgegebenen Trajektorien in Abb. 6.9b bleiben jederzeit weit unter dem Grenzwert. Gleichzeitig erreicht es den Zielfahrstreifen mit einer Durchschnittsgeschwindigkeit von 26,2 $\frac{\text{km}}{\text{h}}$ und ohne Kollision oder Verlassen des Fahrkorridors in 34,1 s. Der hohe Grad an Robustheit und Stabilität des Systemverhaltens fällt insbesondere im direkten Vergleich zum Verhalten ohne Verifikation, also der *Optimistischen Verhaltensarbitration* auf (Abb. 6.9a vs. Abb. 6.7a).

Ergebnisse *Dauerlauf*

Im *Dauerlauf* startet das Ego-Fahrzeug auf dem KIT Campus und soll die gesamte Route von 6,04 km Länge absolvieren. Dabei muss es u. a. Fahrstreifenwechsel durchführen, Kreuzungen und Fußgängerüberwege passieren sowie zum Schluss auf dem Campusgelände in eine festgelegte Parklücke einparken (ausführlichere Streckenbeschreibung in Abschnitt 6.1.1). Da CoInCar-Sim allerdings keine Verkehrsflusssimulation umfasst, können andere Verkehrsteilnehmer nicht dynamisch im Laufe der Simulation initialisiert werden, um mit dem Ego-Fahrzeug in Interaktion zu kommen. Andererseits ist bei einer solch langen Fahrt nicht genau absehbar, wann das Ego-Fahrzeug welche interessante Stelle erreichen wird, sodass eine manuelle Festlegung des Anfangszustands aller Verkehrsteilnehmer nicht praktikabel ist. Daher verläuft der *Dauerlauf* ohne Einbindung anderer Verkehrsteilnehmer und fokussiert sich auf das dynamische Fahrverhalten ohne Interaktion, wodurch die beiden Dauerläufe der *Optimistischen* und *Sicheren Verhaltensentscheidung* wiederum sehr gut miteinander vergleichbar sind.

Abbildungen 6.10 bis 6.12 stellen die Ergebnisse der beiden Dauerläufe dar. Das Ego-Fahrzeug kommt wegen der künstlich beschädigten Trajektorien (Abb. 6.11a), wie in Abb. 6.10 hervorgehoben, ohne Verifikation (also bei der *Optimistischen Verhaltensentscheidung*) ins Schwingen und verlässt mehrmals seinen Fahrkorridor. Außerdem gerät es dadurch zum Schluss ungewollt

[1] Wenn bspw. in einer Situation sowohl *Follow Lane* als auch *Change Lane Left* anwendbar sind.

auf den rechten Fahrstreifen, sodass es anschließend nicht mehr rechtzeitig auf den linken Abbiegefahrstreifen zum KIT Campus wechseln kann. Daher bleibt es an der letzten Kreuzung stehen, statt die Route abzuschließen. Für die zurückgelegten 5,47 km benötigt das Ego-Fahrzeug insgesamt 1365 s $\widehat{=}$ 22 min 45 s.

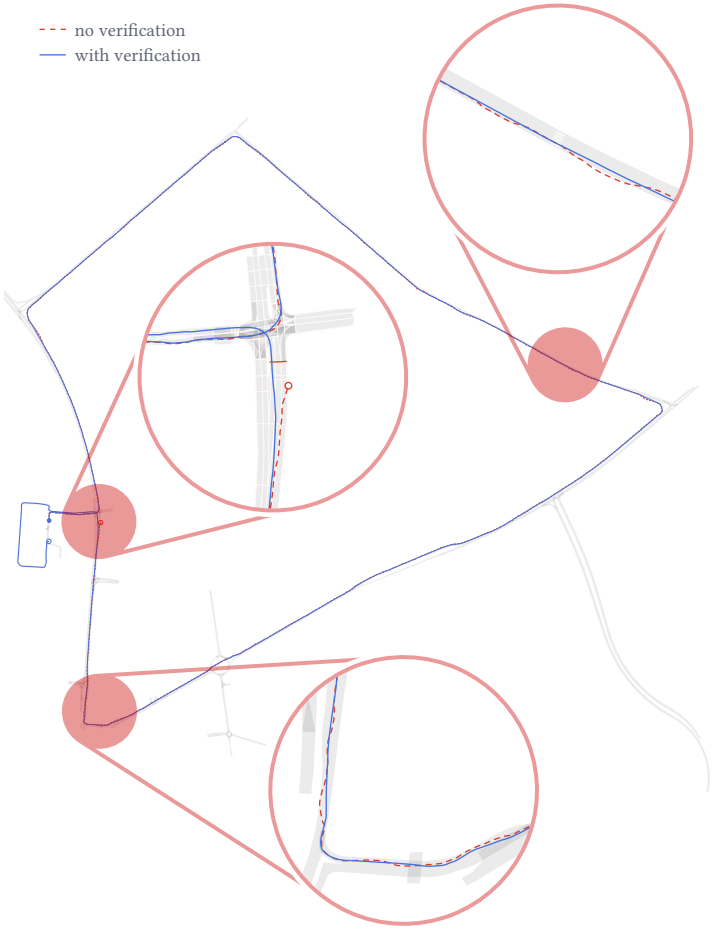
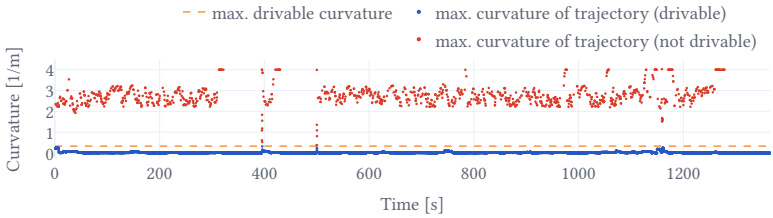
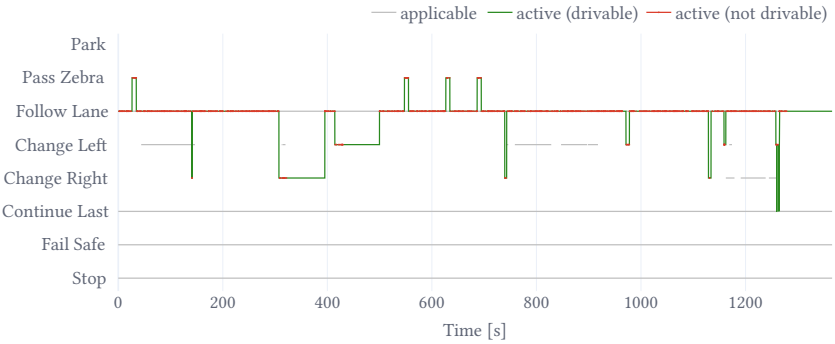


Abbildung 6.10: Gefahrener Pfad der *Optimistischen Verhaltensentscheidung* (rot gestrichelt) im Vergleich zur *Sicheren Verhaltensentscheidung* (blau durchgezogen).



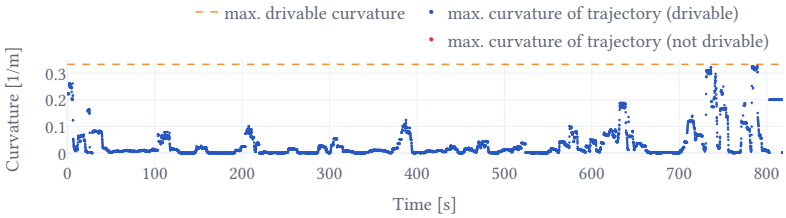
(a) Maximale Krümmung der zum jeweiligen Zeitpunkt geplanten bzw. künstlich beschädigten Trajektorie.



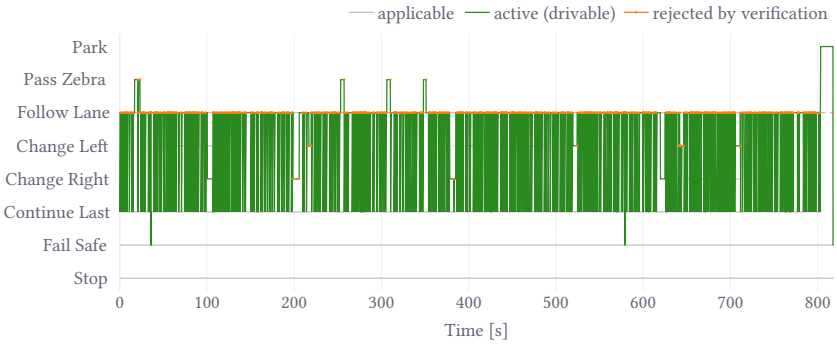
(b) Da die *Optimistische Verhaltensentscheidung* im Arbitrationsverfahren keine Verifikation nutzt, führt es auch die künstlich beschädigten Trajektorien aus. Dadurch gerät das Ego-Fahrzeug mehrmals aus dem Fahrkorridor und wechselt einige Male ungewollt auf einen benachbarten Fahrstreifen.

Abbildung 6.11: Zeitverlauf der *Optimistischen Verhaltensentscheidung* im *Dauerlauf*.

Die *Sichere Verhaltensentscheidung* kann das Fahrzeug hingegen dank des Verifikationsverfahrens stabilisieren, im Fahrkorridor bleiben und die gesamte Route sicher absolvieren. Hierzu fängt es Trajektorien mit zu hoher Krümmung ab (Abb. 6.12a) und wechselt in diesen Fällen auf Alternativoptionen. Dadurch wirkt die Verhaltensentscheidung im Zeitverlauf von Abb. 6.12b auf den ersten Blick wesentlich unruhiger als in Abb. 6.11b. Bei genauerer Betrachtung ist jedoch erkennbar, dass die allermeisten Entscheidungswechsel zwischen der aktuell geeignetsten regulären Verhaltensoption und dem *Continue Last Maneuver* stattfinden. Dieses setzt vorübergehend das Manöver des jeweils vorigen Planungsintervalls fort, was einen stabilisierenden Effekt auf das Gesamtverhalten hat. Folglich bleibt das Ego-Fahrzeug im Fahrkorridor und absolviert die gesamte Route innerhalb von 818 s $\hat{=}$ 13 min 38 s.



(a) Dank der Verifikation werden Trajektorien mit jeweils zu hoher maximaler Krümmung verhindert.



(b) Sobald die Verifikation erkennt, dass die künstlich beschädigte Trajektorie nicht realisierbar ist, wechselt der jeweilige Arbitrator auf die nächstbeste Alternativoption, die der Verifikation standhält. Meist fällt die Arbitration somit auf die *Continue Last Maneuver* Rückfallebene zurück.

Abbildung 6.12: Zeitverlauf der *Sicheren Verhaltensentscheidung* im *Dauerlauf*.

Im direkten Vergleich der beiden Dauerläufe wird zunächst ersichtlich, dass zum einen eine Fehlerwahrscheinlichkeit von $p(\text{Ausfall}_i) = 10\,\%$ zu instabilem und daher riskantem Fahrverhalten führen kann. Zudem erkennt die Verifikation im Arbitrationsschritt diese Fehler (Abb. 6.12a vs. Abb. 6.11a) und stabilisiert das Gesamtverhalten über die Rückfall- und Alternativoptionen.

6.2.4 Sichere Verhaltensentscheidung

Dieser Abschnitt untersucht die Verhaltensarbitration darauf, ob und wie sie riskante Stellgrößen abfängt und in dem Fall auf Alternativoptionen zurückgreift (Sicherheitsziel S6). Die *Optimistische Verhaltensentscheidung* selektiert zwischen den Manöveroptionen ausschließlich auf Grundlage

ihrer Start- bzw. Fortsetzungs-Bedingungen. Daher ist zu erwarten, dass sie ggf. auch riskante Manöveroptionen auswählt, bei deren Ausführung es im schlimmsten Fall zu einem Unfall kommen kann. Je nachdem wie kritisch die aktuelle Situation ist, kann es somit – insbesondere, wenn die Start- oder Fortsetzungs-Bedingung einer der Verhaltensoptionen zu optimistisch entworfen wurde – zu einer gefährlichen Situation oder gar einer Kollision kommen. Dem entgegengesetzt sollte die *Sichere Verhaltensentscheidung* diese Manöveroptionen durch die Verifikation mit $\mathcal{V}_{\text{sicher}}$ als zu riskant erkennen und abweisen. Somit werden nur Manöver ausgeführt, deren Fail-Safe-Belegung sich nicht mit den Worst-Case-Belegungen anderer Verkehrsteilnehmer überlappen. Folglich sollten, solange die Annahmen der Erreichbare-Mengen-Analyse erfüllt bleiben (Abschnitt 2.5.2), keine Kollisionen vom Ego-Fahrzeug verursacht werden.

Die Verifikation bzgl. Verkehrssicherheit wurde in ausgewählten Szenarien evaluiert. Deren Anfangssitationen sind jeweils so konstruiert, dass sie zu einer riskanten Situation führen. Im präsentierten Szenario startet das Ego-Fahrzeug zu Beginn der Rintheimer Querallee, wo ein Fahrstreifenwechsel notwendig wird (vgl. Abb. 6.2a). Dabei erreicht es den zweispurigen Abschnitt in einem Moment, indem ein anderes Fahrzeug in knapper Distanz auf dem Zielfahrstreifen folgt (Abb. 6.13a). Hierbei wurden die Start- und Fortsetzungs-Bedingungen des *ChangeLane* Verhaltensbausteins so manipuliert, dass der Fahrstreifenwechsel auch bei solch knappen Längsabständen eine wahre Start-Bedingung hat. Folglich ist der Fahrstreifenwechsel in diesem kritischen Moment anwendbar, während sich die Belegungen der entsprechenden Fail-Safe-Trajektorie mit den Worst-Case-Belegungen des anderen Verkehrsteilnehmers überschneiden. Außerdem bleibt die Fortsetzungs-Bedingung so lange wahr, bis der Zielfahrstreifen erreicht wurde, ohne durchgehend zu prüfen, ob die Lücke weiterhin groß genug ist.

Ergebnisse *Optimistische Verhaltensentscheidung*

Die Simulationsergebnisse für die riskante Situation zu Beginn der Rintheimer Querallee sind für die *Optimistische Verhaltensentscheidung* in Abb. 6.13

und 6.14 dargestellt. Dabei bietet Abb. 6.13a einen guten Überblick über die Situation: Nach Abbiegen in die Rintheimer Querallee befindet sich das Ego-Fahrzeug auf dem rechten zweier Fahrstreifen in Fahrtrichtung, wobei dieser in absehbarer Distanz endet. Gleichzeitig folgt auf dem linken Fahrstreifen ein weiteres, schnelleres Fahrzeug etwa 5,8 m hinter dem Ego-Fahrzeug.

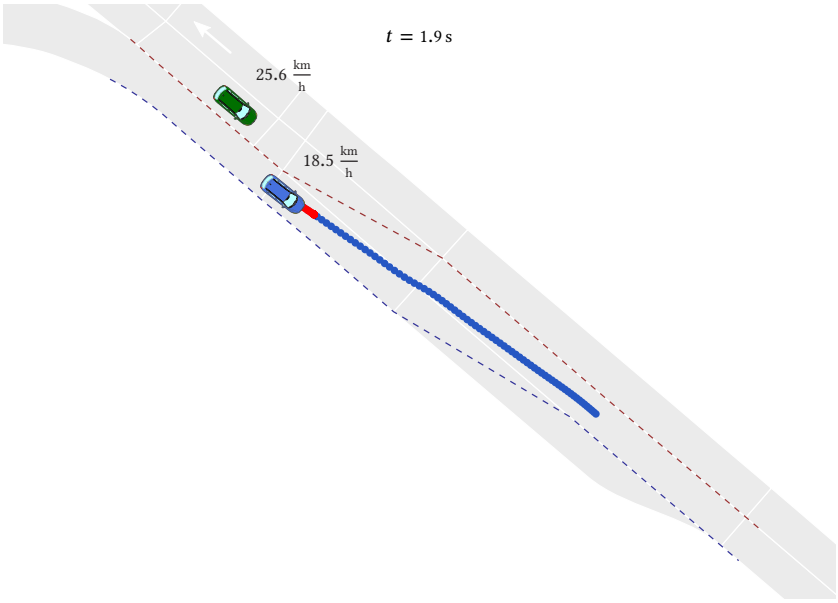
Trotz dieses geringen Abstands deutet das modifizierte *Change Lane Left* Verhalten zum Zeitpunkt $t = 1{,}9\,\mathrm{s}$ in seiner Start-Bedingung an, in der vorliegenden Situation anwendbar zu sein. Da zudem die *Optimistische Verhaltensentscheidung* in ihrer Arbitration keinen Verifikationsschritt durchführt, wählt sie die hinsichtlich Routingkosten beste Option *Change Lane Left* aus (Abb. 6.13b), obwohl sich die Belegung der zugehörigen Fail-Safe-Trajektorie mit den prädizierten Worst-Case-Belegungen der Objekte bereits überschneiden (Abb. 6.14b).

Da das grüne Fahrzeug Vorfahrt hat, reagiert es im weiteren Verlauf zudem nicht auf das Ego-Fahrzeug. Gleichzeitig bleibt das Ego-Fahrzeug durch die Fortsetzungs-Bedingung beim Fahrstreifenwechsel, sodass es zum Zeitpunkt $t = 5{,}3\,\mathrm{s}$, wie in Abb. 6.14a veranschaulicht, zu einer seitlichen Kollision kommt. Der zeitliche Verlauf der Verhaltensentscheidung samt anwendbarer und aktiver Optionen kann zusätzlich in Abb. 6.14c nachvollzogen werden.

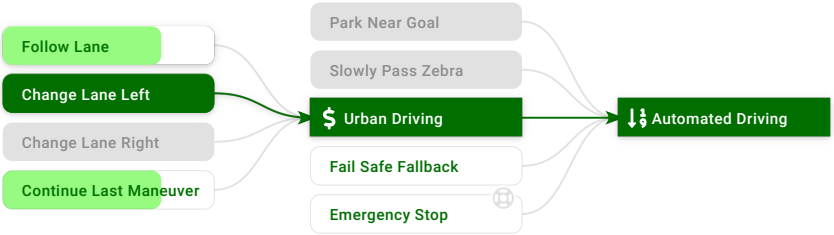
Diese Kollision resultiert u. a. aus folgenden drei Gründen.

1. Die Start-Bedingung des modifizierten *Change Lane Left* Verhaltens ist zu optimistisch und missachtet die vorliegende Vorfahrtsituation.
2. In der Fortsetzungs-Bedingung wird das Verhalten anderer Verkehrsteilnehmer nicht berücksichtigt, insbesondere ob die Abstände zu vorausfahrenden und nachfolgenden Fahrzeugen weiterhin ausreichend sind.
3. Die Arbitratoren führen in diesem Modus keinerlei Verifikation der Soll- oder Fail-Safe-Trajektorien durch, mit der sie die Gefahr erkennen und eine sicherere Option wählen könnten.

Folglich erzeugt *Change Lane Left* ein riskantes Manöver, das vom Urban Driving und Automated Driving Arbitrator ungeprüft ausgeführt wird.

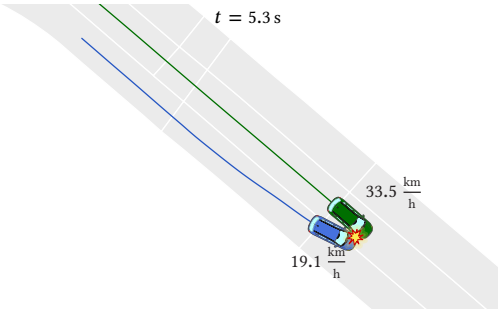


(a) Das blaue Ego-Fahrzeug erreicht einen zweispurigen Abschnitt, an dem ein Fahrstreifenwechsel notwendig ist. Dabei nähert sich hinter dem Ego-Fahrzeug ein vorfahrtsberechtigtes, schnelleres und beschleunigendes Fahrzeug in Grün. Die *Optimistische Verhaltensentscheidung* führt dennoch den Fahrstreifenwechsel aus.



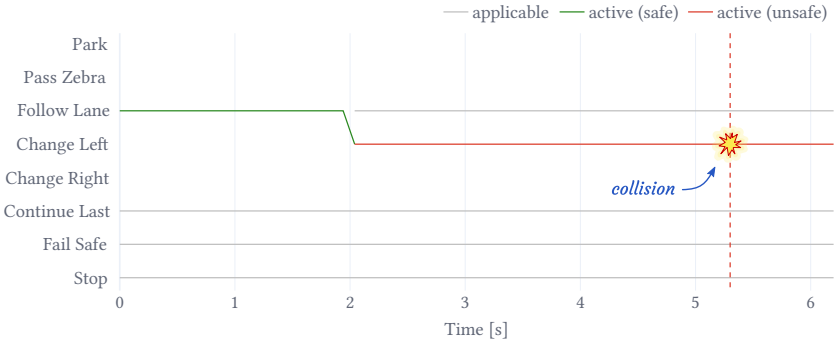
(b) Die Verhaltensbausteine *Follow Lane*, *Change Lane Left*, *Continue Last Maneuver*, *Fail Safe Fallback* und *Emergency Stop* sind aktuell anwendbar. Dabei hat *Change Lane Left* höheren Nutzen und wird ausgewählt, weil es der Route folgt. Die *Optimistische Verhaltensentscheidung* führt allerdings keine Verifikation der Stellgrößen durch, sodass dieses riskante Manöver in Abb. 6.14a zur Kollision führt.

Abbildung 6.13: Beispielsituation der *Optimistischen Verhaltensentscheidung* in der Evaluation bzgl. Verkehrssicherheit im Szenario „Riskanter Fahrstreifenwechsel in der Rintheimer Querallee“ zum Zeitpunkt $t = 1{,}9\,\mathrm{s}$ (bzgl. Szenariostart).



(a) Da das grüne Fahrzeug des Zielfahrstreifens nicht auf das blaue Ego-Fahrzeug reagiert und der räumliche sowie zeitliche Abstand in der Start-Bedingung des modifizierten *Change Lane Left* Verhaltens besonders knapp bemessen ist, kommt es zum Zeitpunkt $t = 5{,}3\,\mathrm{s}$ zu einer seitlichen Kollision.

(b) Die Abstände zwischen den Belegungen der Ego-Fail-Safe-Trajektorie und den Worst-Case-Belegungen der Objekte deuten spätestens ab $t = 2{,}1\,\mathrm{s}$ auf das Kollisionsrisiko hin. Auch der Abstand zwischen der aktuellen Ego- und der Objekt-Hülle sinkt kontinuierlich bis zur Kollision bei $t = 5{,}3\,\mathrm{s}$.



(c) Der Fahrsteifenwechsel wird bereits früh eingeleitet und trotz kleiner werdender Lücke fortgeführt.

Abbildung 6.14: Zeitverlauf der *Optimistischen Verhaltensentscheidung* im Szenario „Riskanter Fahrstreifenwechsel in der Rintheimer Querallee“.

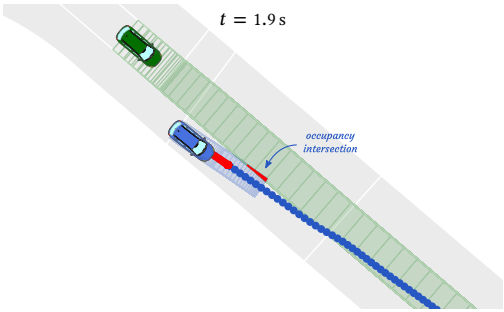
Ergebnisse *Sichere Verhaltensentscheidung*

Die *Sichere Verhaltensentscheidung* nutzt das in Kapitel 4 eingeführte Verifikationsverfahren, um ebenjene riskanten Manöver zu erkennen und in dem Fall auf Alternativoptionen auszuweichen. Von diesem Unterschied in der Verhaltensarbitration abgesehen, sind die Verhaltensbausteine sowie Anfangssitation in diesem sowie vorherigen Abschnitt identisch. Entsprechend zeigen Abb. 6.15 und 6.16 die Simulationsergebnisse derselben riskanten Situation in der Rintheimer Querallee für die *Sichere Verhaltensentscheidung* mit Verifikation.

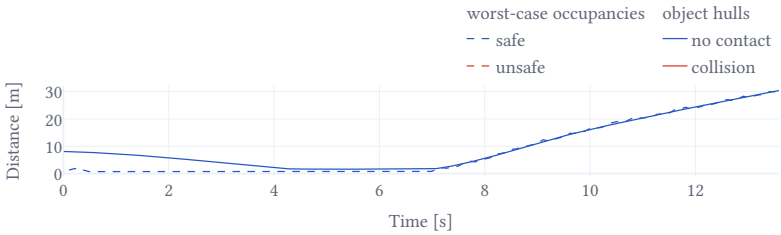
Auch dieses Szenario beginnt zunächst mit der Folgefahrt *Follow Lane* (Abb. 6.15c). Zum Zeitpunkt $t = 1{,}9\,\mathrm{s}$ wird das *Change Lane Left* Verhalten anwendbar, allerdings erkennt der Urban Driving Arbitrator diesmal im Verifikationsschritt, dass es nicht der Erreichbare-Mengen-Analyse standhält: Die Belegung der zugehörigen Fail-Safe-Trajektorie schneidet die Worst-Case-Belegungen des grünen Fahrzeugs, wie in Abb. 6.15a veranschaulicht. Folglich wird *Change Lane Left* als unsicher markiert und die nächstbeste Option *Follow Lane* ausgewählt (Abb. 6.16a und 6.16b).

Während das grüne Fahrzeug am Ego-Fahrzeug vorbeifährt, ist die Start-Bedingung des *Change Lane Left* Verhaltens wieder inaktiv. Erst zum Zeitpunkt $t = 8{,}2\,\mathrm{s}$ ist das grüne Fahrzeug ausreichend weit vorausgefahren, sodass *Change Lane Left* anwendbar wird. Dann weisen die Belegungen der Fail-Safe-Trajektorie auch keine Überlappung mit den Worst-Case-Belegungen des grünen Fahrzeugs auf, sodass *Change Lane Left* sicher ist und ausgeführt wird. Wie in Abb. 6.15b visualisiert, werden im Laufe des Szenarios nur Verhalten ausgewählt, die einen positiven Abstand zu anderen Objekten garantieren.

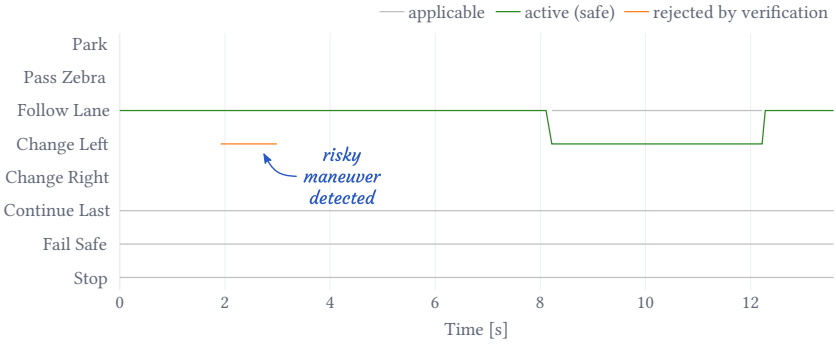
Zusammenfassend hat die *Sichere Verhaltensentscheidung* mithilfe des Verifikators $\mathcal{V}_{\text{sicher}}$ erkannt, dass das *Change Lane Left* Manöver während $t \in [1{,}9\,\mathrm{s}, 3{,}0\,\mathrm{s}]$ riskant sein könnte und ist auf die *Follow Lane* Alternative ausgewichen. Schließlich wurde so die Kollision aus Abb. 6.14a erfolgreich verhindert. Gleichzeitig wurde ein natürliches Fahrverhalten erzeugt, indem das Ego-Fahrzeug zunächst die Geschwindigkeit gedrosselt und sich anschließend hinter dem grünen Fahrzeug auf dem linken Fahrstreifen eingereiht hat.



(a) Belegungen der *Change Lane Left* Fail-Safe-Trajektorie in Blau, Belegungen der Worst-Case-Prädiktionen des anderen Fahrzeugs in Grün. Da sich die Belegungen überschneiden, besteht das *Change Lane Left* Manöver nicht den Verifikationsschritt des übergeordneten Urban Driving Arbitrators (vgl. Abb. 6.16b).

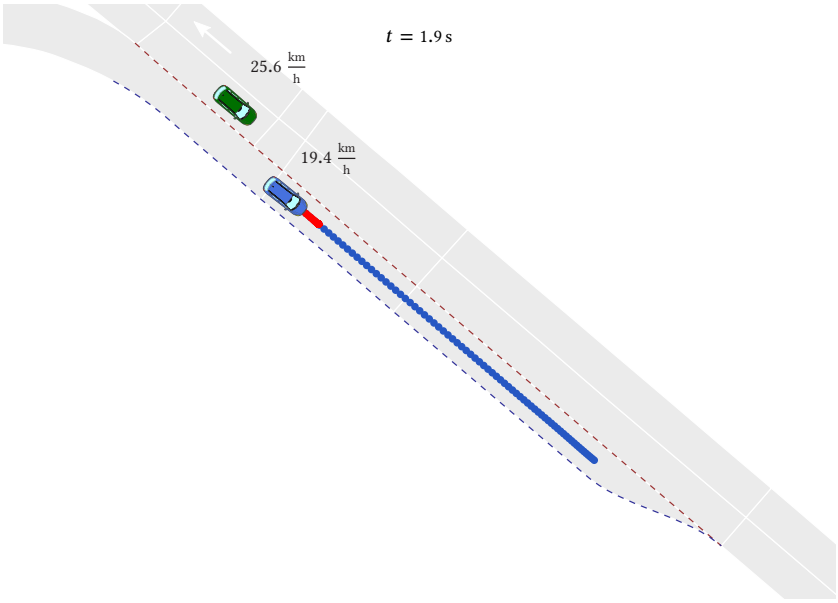


(b) Die Abstände zwischen den Ego-Fail-Safe- und den Objekt-Worst-Case-Belegungen, sowie zwischen den aktuellen Ego- und Objekt-Hüllen, bleiben während des Überholvorgangs durchgehend größer als 0,6 m.

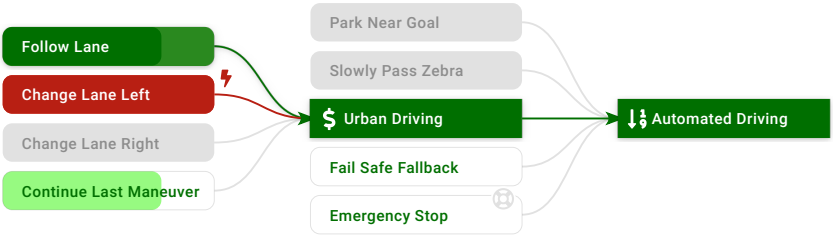


(c) Da *Change Lane Left* anwendbar, aber unsicher ist, wird stattdessen *Follow Lane* ausgeführt. Nachdem das grüne Fahrzeug das Ego-Fahrzeug bei $t = 8{,}2\,\mathrm{s}$ passiert, wird der Fahrstreifenwechsel sicher durchgeführt.

Abbildung 6.15: Zeitverlauf der *Sicheren Verhaltensentscheidung* im Szenario „Riskanter Fahrstreifenwechsel in der Rintheimer Querallee“.



(a) Da das *Change Lane Left* Manöver zum Zeitpunkt $t = 1{,}9\,\mathrm{s}$ zu riskant ist, führt die *Sichere Verhaltensentscheidung* zunächst die Folgefahrt *Follow Lane* aus.



(b) Neben den Rückfallebenen *Continue Last Maneuver*, *Fail Safe Fallback* und *Emergency Stop* sind auch die beiden regulären Verhaltensbausteine *Follow Lane* und *Change Lane Left* anwendbar. Zwar weist *Change Lane Left* einen höheren Nutzen hinsichtlich Routingkosten auf, besteht allerdings nicht die Verifikation. Folglich wird *Follow Lane*, die nächstbeste Option, die die Verifikation besteht, ausgeführt.

Abbildung 6.16: Beispielsituation der *Sicheren Verhaltensentscheidung* im Szenario „Riskanter Fahrstreifenwechsel in der Rintheimer Querallee“ zum Zeitpunkt $t = 1{,}9\,\mathrm{s}$ (bzgl. Szenariostart).

7 Zusammenfassung und Ausblick

In dieser Arbeit wurde eine Systemarchitektur zur sicheren und robusten Verhaltensentscheidung eines automatisierten Fahrzeugs entworfen, die es zudem ermöglicht, vielfältige Methoden der Verhaltensplanung miteinander zu kombinieren. Hierzu wurden grundlegende Verhaltensbausteine mittels generischen Arbitratoren hierarchisch in einem Graphen angeordnet. Die Einbettung der Erreichbare-Mengen-Analyse und Fail-Safe-Trajektorien zur Verifikation der geplanten Trajektorien, sowie diverse Rückfallebenen, garantieren dabei das notwendige Sicherheitsniveau.

Dieses Kapitel fasst die vorgeschlagene Methodik zusammen, bevor es auf die wesentlichen Vor- und Nachteile, insbesondere gegenüber den weitverbreiteten Endlichen Zustandsautomaten, eingeht. Zuletzt werden noch offene Forschungsfragen skizziert und ein Ausblick auf mögliche Verbesserungen gegeben.

7.1 Zusammenfassung

Zu Beginn dieser Arbeit wurde die Methode der Verhaltensarbitration auf das automatisierte Fahren übertragen. Hierzu wurde ein geeignetes *Umweltmodell* und eine Szenario-unabhängige *Manöverrepräsentation* definiert. Das Umweltmodell umfasst statische sowie dynamische Elemente, die die aktuelle Verkehrssituation beschreiben. Darunter fallen u. a. die hochgenaue Planungskarte (sog. Lanelets), der Zustand des Ego-Fahrzeugs sowie Beobachtungen und Prädiktionen anderer Verkehrsteilnehmer. Die zuletzt geplanten

Soll- und Fail-Safe-Trajektorien werden ebenfalls im Umweltmodell hinterlegt. Die von den Verhaltensbausteinen geplanten Manöver werden schließlich durch die Soll- und Fail-Safe-Trajektorie, HMI Ausgaben und V2X Nachrichten repräsentiert.

Anschließend wurde das generische Arbitrationsverfahren um eine ebenso generische *Verifikationslogik* erweitert, sodass nur solche Verhaltensoptionen ausgeführt werden, die einer Verifikation standhalten. Ausgehend von der aktuellen Situation bestimmt ein Arbitrator hierfür zunächst welche seiner Verhaltensbausteine anwendbar sind. Entsprechend seiner Priorisierungsstrategie prüft er anschließend die von dem jeweiligen Verhaltensbaustein zurückgegebene Stellgröße mittels domänenspezifischer Verifikatoren. Sobald eine der Verhaltensoptionen diese Verifikation besteht, gibt der Arbitrator die entsprechende Stellgröße aus. So wird sichergestellt, dass kein ungültiges oder unsicheres Verhalten ausgeführt wird.

Ergänzend wurden drei *Verifikatoren* für das automatisierte Fahren definiert, um potenzielle Manöver auf Gültigkeit, Realisierbarkeit und Verkehrssicherheit zu überprüfen. Ein Manöver wird als gültig eingestuft, sofern seine Trajektorien bzgl. Koordinatensystem und Repräsentation korrekt definiert sind. Mittels kinematischen Einspurmodells wird die Fahrbarkeit und somit Realisierbarkeit der Trajektorien bestimmt. Die Sicherheit wird über eine Erreichbare-Mengen-Analyse verifiziert. Hierzu wird die Belegung der Fail-Safe-Trajektorie mit den Worst-Case-Belegungen aller vorfahrtsberechtigten Verkehrsteilnehmer verglichen. Erst wenn diese keine Überlappungen aufweisen, wird dieses Manöver als nachweislich sicher eingestuft.

Hierauf aufbauend wurde schließlich ein *Sicherheitskonzept* entworfen, das die Ausfallsicherheit und Verkehrssicherheit der Verhaltensgenerierung gewährleistet. Dazu setzt es zur Fehlervermeidung auf einen sorgfältigen Systementwurf und auf Echtzeitfähigkeit. Durch Verifikation werden Fehler, wie ausgefallene Verhaltensbausteine oder zu riskante Fahrmanöver, bereits im Entscheidungsprozess erkannt. Das Arbitrationsverfahren kompensiert solche Fehler schließlich mittels Redundanz und Diversität in Form von Rückfallebenen. Die erste Rückfallebene gibt die zuletzt ausgeführte Soll-Trajektorie

zurück, solange diese realisierbar und nicht zu alt ist sowie der Verifikation standhält. Ist dies nicht mehr möglich, führt die nächste Ebene die letzte Fail-Safe-Trajektorie aus, sofern sie weiterhin beweisbar sicher ist. Die dritte Rückfallebene stellt ein Nothaltemanöver zur Verfügung und darf als einzige Verhaltensoption auch dann ausgeführt werden, wenn ihre Verifikation fehlschlägt. So kann jederzeit ein MRM bereitgestellt werden.

Zuletzt wurde das Sicherheitskonzept in der CoInCar Simulationsumgebung auf einer für Realversuche genutzten Karlsruher Teststrecke evaluiert. Die Implementierung wurde insbesondere hinsichtlich der Sicherheitsziele untersucht, ungültige und riskante Stellgrößen zu verhindern. Hierfür wurde das Verhaltensmodell des Ego-Fahrzeugs verschiedenen Stresstests und Modifikationen unterworfen. Im ersten Teil wurde die Soll-Trajektorie einzelner Verhaltensbausteine mit einer festgelegten Fehlerwahrscheinlichkeit reproduzierbar beschädigt. Hierbei konnte gezeigt werden, dass die Verhaltensentscheidung dank der Fehlererkennung und -kompensation auch bei hohen Ausfallraten in der Lage war, ein sicheres und stabiles Fahrverhalten zu erzeugen. Im zweiten Teil wurde das Sicherheitskonzept in konstruierten, besonders riskanten Szenarien bzgl. der Verkehrssicherheit untersucht. Dank der Verifikation mit der Erreichbaren-Mengen-Analyse konnten Manöver, die ohne das vorgestellte Sicherheitskonzept zu einer Kollision geführt hätten, rechtzeitig erkannt und stattdessen eine sichere Alternativoption ausgeführt werden.

7.2 Diskussion

Die in dieser Arbeit vorgestellte Methode zur Verhaltensentscheidung kombiniert die Arbitrationsgraphen mit Maßnahmen aus dem Bereich der zuverlässigen und fehlertoleranten Systeme. Hierdurch ergeben sich u. a. folgende Vorteile des Ansatzes, die teilweise auch in [Orz20] präsentiert wurden:

Unterstützung unterschiedlicher Planungsmethoden Die Verhaltensarbitration bietet eine geeignete Möglichkeit, unterschiedliche Planungsmethoden innerhalb einer Architektur zu vereinen. Diese können

sowohl verschiedene Fahrkompetenzen und Szenarien adressieren, als auch dieselbe Fahrkompetenz mittels vielfältiger, sich ergänzender Methoden umsetzen. In der Evaluation wurden bspw. zwei verschiedene Trajektorienplaner eingesetzt. Ein vereinfachter Ansatz auf IDM-Basis adressierte urbane korridorbasierte Manöver, während für das Parkmanöver mit der RRT* Methode *Hybrid Curvature* Trajektorien geplant wurden.

Konsequente Modularität und Skalierbarkeit Verhaltensbausteine bestimmen in ihren Start- und Fortsetzungs-Bedingungen selbst, wann sie anwendbar sind. Dadurch bleiben Arbitratoren anwendungsunabhängig und benötigen kein Wissen über die Voraussetzungen zum Aufruf einzelner Verhaltensbausteine. Diese Entkopplung der Situationsinterpretation von der Auswahllogik erhöht die Wiederverwendbarkeit, Testbarkeit und Wartbarkeit der Verhaltensbausteine sowie Arbitratoren. Die konsequente Modularität führt außerdem zu einer guten Skalierbarkeit sowohl bzgl. Systemkomplexität als auch der Laufzeit.

Bottom-Up Entwurf Ausgehend von den ODDs und adressierten Verhaltenskompetenzen werden im Systementwurf zunächst grundlegende Verhaltensbausteine entwickelt. Diese werden anschließend im Bottom-Up Verfahren zu einem hierarchischen Arbitrationsgraphen zusammengesetzt. Somit können die Verhaltensbausteine voneinander unabhängig entworfen und entwickelt werden. Komplexes Systemverhalten, wie mehrfach aufeinanderfolgende Fahrstreifenwechsel um den Abbiegefahrstreifen zu erreichen, entsteht zudem aus einfachen Bausteinen, ohne jedes dieser Szenarien durch spezialisierte Entscheidungs- oder Planungsverfahren explizit adressieren zu müssen.

Iterativ erweiterbar In der Weiterentwicklung eines Systems können Arbitrationsgraphen stets iterativ erweitert werden. Um neue Verhaltensbausteine hinzuzufügen, müssen lediglich ihre Start- und Fortsetzungs-Bedingungen definiert, sowie eine geeignete Platzierung innerhalb des Arbitrationsgraphen festgelegt werden. Dank der starken Entkopplung

sind keine Änderungen an einem der anderen Verhaltensbausteine erforderlich.

Nachvollziehbares Verhalten Der streng modulare Aufbau sowie die konsequente Trennung von Situationsinterpretation und Auswahllogik trägt im Vergleich zu Endlichen Zustandsautomaten oder klassischen verhaltensbasierten Systemen zu einer deutlich besseren Verständlichkeit bei. Sowohl die Start- und Fortsetzungs-Bedingungen der Verhaltensbausteine als auch die Auswahllogik der Arbitratoren sind transparent und nachvollziehbar. Dadurch kann der hierarchische Entscheidungsprozess schnell erfasst und im zeitlichen Verlauf gut nachvollzogen werden.

Robustes und sicheres Verhalten Geben Verhaltensbausteine fehlerhafte Stellgrößen zurück oder fallen sogar komplett aus, stellt die Verifikationslogik sicher, dass auf Alternativoptionen sowie die Rückfallebenen zurückgegriffen wird. Ggf. auftretende Ausfälle können dadurch frühzeitig eingegrenzt werden und beeinträchtigen nicht die Stabilität des Gesamtsystems. In der Anwendung des automatisierten Fahrens werden somit ungültige, nicht realisierbare oder riskante Trajektorien abgefangen und ein robustes und sicheres Fahrverhalten sichergestellt.

Mehrschichtiges Sicherheitskonzept Die vorgestellte Verifikationslogik ermöglicht es, Stellgrößen vor der Ausführung hinsichtlich vielfacher Kriterien zu überprüfen. Für den Fehlerfall können zudem mehrere abgestufte Rückfallebenen definiert werden, sodass das System bei Ausfällen reibungslos degradiert. Im automatisierten Fahren werden die Trajektorien hinsichtlich ihrer Gültigkeit, Realisierbarkeit und Verkehrssicherheit untersucht. In den Rückfallebenen wird zunächst versucht, die zuletzt ausgeführte Trajektorie fortzusetzen. Ist diese nicht mehr sicher, kann im Regelfall auf die Fail-Safe-Trajektorie zurückgegriffen werden. Muss allerdings auch diese wegen unerwarteter Ereignisse als nicht mehr sicher eingestuft werden, wird schließlich eine Notbremsung eingeleitet.

7.3 Ausblick

Der Einsatz von Arbitrationsgraphen im automatisierten Fahren stellt einen vielversprechenden Ansatz für eine sichere und robuste Verhaltensentscheidung dar. Dennoch gibt es in der Verhaltensplanung, -entscheidung und -verifikation noch zahlreiche offene Forschungsfragen und Verbesserungspotential.

In der Trajektorienplanung würde es sich anbieten, die Fail-Safe- und Soll-Trajektorie in einem gemeinsamen Planungsproblem zu lösen. Bisher wurden in dieser sowie anderen Publikationen [Alt16, Pek18] zunächst die Soll-Trajektorie und erst anschließend die Fail-Safe-Trajektorie geplant. Dadurch kann es in kritischen Situationen zu einer zu optimistischen Soll-Trajektorie kommen, für die keine beweisbar sichere Fail-Safe-Trajektorie mehr gefunden werden kann. In solchen Fällen muss die Arbitration deswegen auf die Alternativoptionen zurückgreifen. Integriert man hingegen beide Trajektorien in ein Planungsproblem, würden nur Soll-Trajektorien geplant, die eine sichere Fail-Safe-Trajektorie ermöglichen und somit sicher sind.

Im aktuellen Ansatz plant jeder Verhaltensbaustein eine Fail-Safe-Trajektorie. Deren Sicherheit überprüft der übergeordnete Arbitrator mit dem $\mathcal{V}_{\text{sicher}}$ Verifikator, indem sie mit den Worst-Case-Belegungen verglichen wird. Hierbei kann die Kombination aller möglichen Worst-Case-Ereignisse innerhalb einer Prädiktion zu übervorsichtigem Verhalten führen und somit die Effizienz und Akzeptanz des Systems schmälern. Alternativ könnte die Verifikation verschiedene einzelne Worst-Case-Szenarien abbilden (Gegenverkehr schert aus, Kind rennt auf die Straße, vorausfahrendes Fahrzeug leitet Notbremsung ein, etc.) und von den Verhaltensbausteinen für jede dieser Möglichkeiten jeweils eine Fail-Safe-Trajektorie fordern. Somit wäre das Verhalten weniger konservativ und könnte gleichzeitig mögliche Regelverstöße (engl. „incompliant behavior“) mit berücksichtigen. Allerdings wäre das Verhalten dann ggf. nicht mehr sicher, wenn mehrere dieser Risikofaktoren gleichzeitig eintreten (z. B. schert der Gegenverkehr aus, weil ein Kind auf die Straße rennt).

In der Belegungsprädiktion werden zudem bisher nur vorfahrtsberechtigte Verkehrsteilnehmer berücksichtigt, wobei dem Ego-Fahrzeug voraussichtlich auch bei Kollisionen mit nicht vorfahrtsberechtigten Verkehrsteilnehmern eine Teilschuld zugeschrieben würde, wenn es nicht adäquat auf diese reagiert. Daher ist eine noch offene Fragestellung, wie Regelverstöße anderer Verkehrsteilnehmer in der Verifikation berücksichtigt werden können. In einem ersten Schritt kann für nicht-vorfahrtsberechtigte Verkehrsteilnehmer zumindest auch eine Belegung mit Best-Case-Annahme berücksichtigt werden.

Eine weitere interessante Forschungsfrage ist, wie die Verifikationslogik mit den Rückfallebenen genutzt werden kann, um – hinsichtlich einer angestrebten Fahrkompetenz – bestmöglich zwischen verschiedenen Verhaltensplanern zu wählen. Für ein interaktives Reißverschlussverfahren könnte bspw. mit höchster Priorität ein Verfahren des Maschinellen Lernens eingesetzt werden. Liefert dieses eine ungültige oder zu riskante Trajektorie, kann auf eine Trajektorienoptimierung mittels Sequentieller Quadratischer Programmierung gewechselt werden. Konvergiert diese nicht oder zu langsam, würde schließlich ein IDM-basierter Ansatz eine akzeptable Trajektorie liefern. Eine parallele Auswertung der verschiedenen Optionen würde dabei die Echtzeitfähigkeit sicherstellen.

Zuletzt bietet die Verhaltensarbitration mit dem vorgestellten Sicherheitskonzept eine optimale Grundlage, um unsichere Planungsmethoden in der Verhaltensentscheidung abzusichern. Daher stellt der Einsatz von Methoden des Maschinellen Lernens zur Verhaltensplanung innerhalb des Arbitrationsverfahrens ein spannendes Forschungsfeld dar und sollte ausgiebig untersucht werden.

Literatur

[Aeb15] Aeberhard, M.; Rauch, S.; Bahram, M.; Tanzmeister, G.; Thomas, J.; Pilat, Y.; Homm, F.; Huber, W. und Kaempchen, N.: „Experience, Results and Lessons Learned from Automated Driving on Germany's Highways". In: *IEEE Intelligent Transportation Systems Magazine* 7.1 (20. Jan. 2015), S. 42–57. doi: 10.1109/MITS.2014.2360306 (siehe S. 3, 14).

[Alt16] Althoff, M. und Magdici, S.: „Set-Based Prediction of Traffic Participants on Arbitrary Road Networks". In: *IEEE Transactions on Intelligent Vehicles* 1.2 (Juni 2016), S. 187–202. doi: 10.1109/TIV.2016.2622920 (siehe S. 8, 43–46, 56, 140).

[Apt19] Aptiv Services US, LLC; AUDI AG; Bayrische Motoren Werke AG; Beijing Baidu Netcom Science Technology Co., Ltd; Continental Teves AG & Co oHG; Daimler AG; FCA US LLC; HERE Global B.V.; Infineon Technologies AG; Intel u. a.: Safety First for Automated Driving. Troy, MI, USA, 2. Juli 2019. url: https://www.mercedes-benz.com/en/innovation/safety-first-for-automated-driving/ (siehe S. 35, 38, 39).

[Ard10] Ardelt, M.; Waldmann, P.; Homm, F. und Kaempchen, N.: „Strategic Decision-Making Process in Advanced Driver Assistance Systems". In: *IFAC Proceedings Volumes*. 6th IFAC Symposium on Advances in Automotive Control 43.7 (1. Juli 2010), S. 566–571. doi: 10.3182/20100712-3-DE-2013.00006 (siehe S. 17).

[Ard11] Ardelt, M. und Waldmann, P.: „Hybrides Steuerungs- und Regelungskonzept für das hochautomatisierte Fahren auf Autobahnen“. In: *at – Automatisierungstechnik* 59.12 (1. Dez. 2011), S. 738–750. doi: 10.1524/auto.2011.0961 (siehe S. 6, 17, 18).

[Bac08] Bacha, A.; Bauman, C.; Faruque, R.; Fleming, M.; Terwelp, C.; Reinholtz, C.; Hong, D.; Wicks, A.; Alberi, T. und Anderson, D.: „Odin: Team Victortango's Entry in the Darpa Urban Challenge“. In: *Journal of Field Robotics* 25.8 (2008), S. 467–492. doi: 10.1002/rob.20248 (siehe S. 14).

[Bad21] Badue, C.; Guidolini, R.; Carneiro, R. V.; Azevedo, P.; Cardoso, V. B.; Forechi, A.; Jesus, L.; Berriel, R.; Paixão, T. M.; Mutz, F. u. a.: „Self-Driving Cars: A Survey“. In: *Expert Systems with Applications* 165 (1. März 2021), S. 113816. doi: 10.1016/j.eswa.2020.113816 (siehe S. 1).

[Bag12] Bagnell, J. A.; Cavalcanti, F.; Cui, L.; Galluzzo, T.; Hebert, M.; Kazemi, M.; Klingensmith, M.; Libby, J.; Liu, T. Y.; Pollard, N. u. a.: „An Integrated System for Autonomous Robotics Manipulation“. In: *International Conference on Intelligent Robots and Systems*. Vilamoura, Portugal: IEEE/RSJ, Okt. 2012, S. 2955–2962. doi: 10.1109/IROS.2012.6385888 (siehe S. 19, 21).

[Ban17] Banzhaf, H.; Nienhüser, D.; Knoop, S. und Zöllner, J. M.: „The Future of Parking: A Survey on Automated Valet Parking with an Outlook on High Density Parking“. In: *Intelligent Vehicles Symposium*. Redondo Beach, CA, USA: IEEE, Juni 2017, S. 1827–1834. doi: 10.1109/IVS.2017.7995971 (siehe S. 88).

[Ban18] Banzhaf, H.; Dolgov, M.; Stellet, J. und Zöllner, J. M.: „From Footprints to Beliefprints: Motion Planning under Uncertainty for Maneuvering Automated Vehicles in Dense Scenarios“. In: *International Conference on Intelligent Transportation Systems*. Maui, Hawaii, USA: IEEE, Nov. 2018, S. 1680–1687. doi: 10.1109/ITSC.2018.8569897 (siehe S. 4, 29, 56, 89).

[Ban19] Bansal, M.; Krizhevsky, A. und Ogale, A.: „ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst“. In: *Proceedings of Robotics: Science and Systems*. Freiburg, Deutschland, Juni 2019. doi: 10.15607/RSS.2019.XV.031 (siehe S. 3).

[Bar15] Bartels, A.; Rohlfs, M.; Hamel, S.; Saust, F. und Klauske, L. K.: „Querführungsassistenz“. In: *Handbuch Fahrerassistenzsysteme*. Hrsg. von Winner, H.; Hakuli, S.; Lotz, F. und Singer, C. ATZ/MTZ-Fachbuch. Wiesbaden, Deutschland: Springer Fachmedien, 2015, S. 937–957. doi: 10.1007/978-3-658-05734-3_49 (siehe S. 91).

[Ben14] Bengler, K.; Dietmayer, K.; Farber, B.; Maurer, M.; Stiller, C. und Winner, H.: „Three Decades of Driver Assistance Systems: Review and Future Perspectives“. In: *IEEE Intelligent Transportation Systems Magazine* 6.4 (27. Okt. 2014), S. 6–22. doi: 10.1109/MITS.2014.2336271 (siehe S. 1).

[Ben15] Bender, P.; Taş, Ö. Ş.; Ziegler, J. und Stiller, C.: „The Combinatorial Aspect of Motion Planning: Maneuver Variants in Structured Environments“. In: *Intelligent Vehicles Symposium*. Seoul, Korea: IEEE, 2015, S. 1386–1392. doi: 10.1109/IVS.2015.7225909 (siehe S. 28).

[Bou14] Bouraine, S.; Fraichard, T.; Azouaoui, O. und Salhi, H.: „Passively Safe Partial Motion Planning for Mobile Robots with Limited Field-of-Views in Unknown Dynamic Environments“. In: *International Conference on Robotics and Automation*. Hongkong, China: IEEE, Mai 2014, S. 3576–3582. doi: 10.1109/ICRA.2014.6907375 (siehe S. 42).

[Bou19] Bouton, M.; Nakhaei, A.; Fujimura, K. und Kochenderfer, M. J.: „Cooperation-Aware Reinforcement Learning for Merging in Dense Traffic“. In: *Intelligent Transportation Systems Conference*. Auckland, Neuseeland: IEEE, Okt. 2019, S. 3441–3447. doi: 10.1109/ITSC.2019.8916924 (siehe S. 102).

[Bre14] Brechtel, S.; Gindele, T. und Dillmann, R.: „Probabilistic Decision-Making under Uncertainty for Autonomous Driving Using Continuous POMDPs“. In: *Conference on Intelligent Transportation Systems*. Qingdao, China: IEEE, Okt. 2014, S. 392–399. doi: 10.1109/ITSC.2014.6957722 (siehe S. 42).

[Bro86] Brooks, R.: „A Robust Layered Control System for a Mobile Robot“. In: *IEEE Journal on Robotics and Automation* 2.1 (März 1986), S. 14–23. doi: 10.1109/JRA.1986.1087032 (siehe S. 7, 21).

[BSI20a] BSI Standards Limited: BSI Connected and Automated Vehicles – Vocabulary. BSI Flex 1890. London, Vereinigtes Königreich, Okt. 2020. url: https://www.bsigroup.com/globalassets/localfiles/en-gb/cav/bsi-flex-1890-v3-2020-10.pdf (besucht am 04. 05. 2022) (siehe S. 35).

[BSI20b] BSI Standards Limited: Operational Design Domain (ODD) Taxonomy for an Automated Driving System (ADS) – Specification. PAS 1883:2020. London, Vereinigtes Königreich, Aug. 2020. url: https://www.bsigroup.com/globalassets/localfiles/en-gb/cav/pas1883.pdf (besucht am 04. 05. 2022) (siehe S. 35, 36).

[Bun21] Bundesministerium der Justiz: Gesetz zur Änderung des Straßenverkehrsgesetzes und des Pflichtversicherungsgesetzes – Gesetz zum autonomen Fahren. Bundesgesetzblatt Jahrgang 2021 Teil I Nr. 48. Bonn, Deutschland, 27. Juli 2021 (siehe S. 37, 38).

[Bur17] Burger, C.; Orzechowski, P. F.; Taş, Ö. Ş. und Stiller, C.: „Rating Cooperative Driving: A Scheme for Behavior Assessment“. In: *Conference on Intelligent Transportation Systems*. Yokohama, Japan: IEEE, Okt. 2017, S. 1–6. doi: 10.1109/ITSC.2017.8317794 (siehe S. 54).

[Bur18] Burri, A.: TinyFSM. Version 0.3.2. Zürich, Schweiz: Digital Integrity GmbH, Okt. 2018. url: https://digint.ch/tinyfsm/ (besucht am 07. 05. 2022) (siehe S. 14).

[Car16] Carmo, M. P. do: Differential Geometry of Curves and Surfaces. Revised & updated second edition. Mineola, New York, USA: Dover Publications, 14. Dez. 2016. 510 S. (siehe S. 28).

[Cas21] Casas, S.; Sadat, A. und Urtasun, R.: „MP3: A Unified Model to Map, Perceive, Predict and Plan“. In: *Conference on Computer Vision and Pattern Recognition*. Nashville, TN, USA: IEEE/CVF, Juni 2021, S. 14398–14407. doi: 10.1109/CVPR46437.2021.01417 (siehe S. 3, 5).

[Che19] Chen, J.; Yuan, B. und Tomizuka, M.: „Model-Free Deep Reinforcement Learning for Urban Autonomous Driving“. In: *Intelligent Transportation Systems Conference*. Auckland, Neuseeland: IEEE, Okt. 2019, S. 2765–2771. doi: 10.1109/ITSC.2019.8917306 (siehe S. 5).

[Chu09] Chung, W.; Kim, S.; Choi, M.; Choi, J.; Kim, H.; Moon, C. b und Song, J. B.: „Safe Navigation of a Mobile Robot Considering Visibility of Environment“. In: *IEEE Transactions on Industrial Electronics* 56.10 (Okt. 2009), S. 3941–3950. doi: 10.1109/TIE.2009.2025293 (siehe S. 42, 43).

[Col17] Colledanchise, M. und Ögren, P.: „How Behavior Trees Modularize Hybrid Control Systems and Generalize Sequential Behavior Compositions, the Subsumption Architecture, and Decision Trees“. In: *IEEE Transactions on Robotics* 33.2 (Apr. 2017), S. 372–389. doi: 10.1109/TRO.2016.2633567 (siehe S. 21).

[Col18] Colledanchise, M. und Ögren, P.: Behavior Trees in Robotics and AI: An Introduction. Boca Raton, FL, USA: CRC Press, 20. Juli 2018. doi: 10.1201/9780429489105 (siehe S. 7, 19–21).

[Dah72] Dahl, O. J.; Dijkstra, E. W. und Hoare, C. A. R., Hrsg.: Structured Programming. London, Vereinigtes Königreich: Academic Press Ltd., 1972. 234 S. (siehe S. 17).

[Dan14] Danielis, P.; Skodzik, J.; Altmann, V.; Schweissguth, E. B.; Golatowski, F.; Timmermann, D. und Schacht, J.: „Survey on Real-Time Communication via Ethernet in Industrial Automation Environments“. In: *Emerging Technology and Factory*

Automation. Barcelona, Spanien: IEEE, Sep. 2014, S. 1–8. DOI: 10.1109/ETFA.2014.7005074 (siehe S. 72).

[Deb21] Debada, E.; Ung, A. und Gillet, D.: „Occlusion-Aware Motion Planning at Roundabouts". In: *IEEE Transactions on Intelligent Vehicles* 6.2 (Juni 2021), S. 276–287. DOI: 10.1109/TIV.2020.3019211 (siehe S. 3, 4).

[Dic15] Dickmann, J.; Appenrodt, N.; Klappstein, J.; Bloecher, H. L.; Muntzinger, M.; Sailer, A.; Hahn, M. und Brenk, C.: „Making Bertha See Even More: Radar Contribution". In: *IEEE Access* 3 (2015), S. 1233–1247. DOI: 10.1109/ACCESS.2015.2454533 (siehe S. 50).

[Dij68] Dijkstra, E. W.: „Letters to the Editor: Go to Statement Considered Harmful". In: *Communications of the ACM* 11.3 (1. März 1968), S. 147–148. DOI: 10.1145/362929.362947 (siehe S. 17).

[DIN06] DIN Deutsches Institut für Normung e.V.: Analysetechniken für die Funktionsfähigkeit von Systemen - Verfahren für die Fehlzustandsart- und -auswirkungsanalyse (FMEA). DIN EN 60812. Berlin, Deutschland: Beuth Verlag GmbH, Nov. 2006. DOI: 10.31030/9771315 (siehe S. 35, 163).

[DIN07] DIN Deutsches Institut für Normung e.V.: Fehlzustandsbaumanalyse. DIN EN 61025. Berlin, Deutschland: Beuth Verlag GmbH, Aug. 2007. DOI: 10.31030/9862300 (siehe S. 35, 163).

[Dol10] Dolgov, D.; Thrun, S.; Montemerlo, M. und Diebel, J.: „Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments". In: *The International Journal of Robotics Research* 29.5 (1. Apr. 2010), S. 485–501. DOI: 10.1177/0278364909359210 (siehe S. 88).

[Dre09] Drewes, J.: „Verkehrssicherheit Im Systemischen Kontext". Diss. Braunschweig, Deutschland: Technische Universität Braunschweig, 3. Sep. 2009. DOI: 10.24355/dbbs.084-201001071044-0 (siehe S. 34).

[Dri14] Dribbusch, H.; Kaun, L. und Stoll, E.: Lange Arbeitszeiten, bescheidener Verdienst: Berufskraftfahrer im Güterverkehr. Lohnspiegel Arbeitspapier 27. Düsseldorf, Deutschland: Hans-Böckler-Stiftung, Sep. 2014. url: https://www.lohnspiegel.de/berufskraftfahrer-13917.htm (besucht am 26. 03. 2022) (siehe S. 2).

[Dub13] Dubrova, E.: Fault-Tolerant Design. New York, NY, USA: Springer-Verlag, 2013. doi: 10.1007/978-1-4614-2113-9 (siehe S. 32, 63).

[Ech90] Echtle, K.: Fehlertoleranzverfahren. Studienreihe Informatik. Berlin, Deutschland: Springer-Verlag, 1990. doi: 10.1007/978-3-642-75765-5 (siehe S. 32, 34, 63, 74).

[Fra13] Franke, U.; Pfeiffer, D.; Rabe, C.; Knoeppel, C.; Enzweiler, M.; Stein, F. und Herrtwich, R. G.: „Making Bertha See“. In: *International Conference on Computer Vision Workshops*. Sydney, Australien: IEEE, Dez. 2013, S. 214–221. doi: 10.1109/ICCVW.2013.36 (siehe S. 50).

[Fuc18] Fuchs, S.: Härtetest für Fahrroboter – Das Testfeld Autonomes Fahren Baden-Württemberg wurde in Karlsruhe eingeweiht – KIT Campusreport. Unter Mitarb. von Zöllner, J. M. 22. Mai 2018. doi: 10.5445/DIVA/2018-334 (siehe S. 55).

[Gam21] Gammell, J. D. und Strub, M. P.: „Asymptotically Optimal Sampling-Based Motion Planning Methods“. In: *Annual Review of Control, Robotics, and Autonomous Systems* 4.1 (2021), S. 295–318. doi: 10.1146/annurev-control-061920-093753 (siehe S. 14).

[Gon16] González, D.; Pérez, J.; Milanés, V. und Nashashibi, F.: „A Review of Motion Planning Techniques for Automated Vehicles“. In: *IEEE Transactions on Intelligent Transportation Systems* 17.4 (Apr. 2016), S. 1135–1145. doi: 10.1109/TITS.2015.2498841 (siehe S. 28).

[Göt22] Götzke, M.: „Das Ende der Truckerromantik – Ausbeutung auf der Autobahn“. In: *Deutschlandfunk. Das Wochenendjournal* (19. Feb. 2022). url: https://www.deutschlandfunk.de/das-ende-der-truckerromantik-ausbeutung-auf-der-autobahn-dlf-713fdec2-100.html (besucht am 26. 03. 2022) (siehe S. 2).

[Gre21a] Greis, F.: „Autonomes Fahren: Staupilot von Mercedes verzögert sich offenbar“. In: *Golem.de* (11. Aug. 2021). url: https://www.golem.de/news/autonomes-fahren-staupilot-von-mercedes-verzoegert-sich-offenbar-2111-160923.html (besucht am 03. 12. 2021) (siehe S. 91).

[Gre21b] Gresyk, A.: High-Performance Hierarchical Finite State Machine. Version 1.8.0. Jan. 2021. url: https://hfsm.dev (besucht am 18. 01. 2021) (siehe S. 14).

[Gut17] Gutjahr, B.; Gröll, L. und Werling, M.: „Lateral Vehicle Trajectory Optimization Using Constrained Linear Time-Varying MPC“. In: *IEEE Transactions on Intelligent Transportation Systems* 18.6 (Juni 2017), S. 1586–1595. doi: 10.1109/TITS.2016.2614705 (siehe S. 4, 29).

[Hoe17] Hoermann, S.; Kunz, F.; Nuss, D.; Reuter, S. und Dietmayer, K.: „Entering Crossroads with Blind Corners. A Safe Strategy for Autonomous Vehicles“. In: *Intelligent Vehicles Symposium*. Redondo Beach, CA, USA: IEEE, Juni 2017, S. 727–732. doi: 10.1109/IVS.2017.7995803 (siehe S. 3, 42).

[Hop07] Hopcroft, J. E.; Motwani, R. und Ullman, J. D.: Introduction to Automata Theory, Languages, and Computation. 3rd. Boston, MA, USA: Pearson Addison Wesley, 2007. 535 S. url: https://doi.org/10.1145/568438.568455 (siehe S. 14).

[Hub18a] Hubmann, C.; Schulz, J.; Becker, M.; Althoff, D. und Stiller, C.: „Automated Driving in Uncertain Environments: Planning With Interaction and Uncertain Maneuver Prediction“. In: *IEEE Transactions on Intelligent Vehicles* 3.1 (März 2018), S. 5–17. doi: 10.1109/TIV.2017.2788208 (siehe S. 5).

[Hub18b] Hubmann, C.; Schulz, J.; Xu, G.; Althoff, D. und Stiller, C.: „A Belief State Planner for Interactive Merge Maneuvers in Congested Traffic“. In: *International Conference on Intelligent Transportation Systems*. Maui, Hawaii, USA: IEEE, Nov. 2018, S. 1617–1624. doi: 10.1109/ITSC.2018.8569729 (siehe S. 56, 102).

[Hub19] Hubmann, C.; Quetschlich, N.; Schulz, J.; Bernhard, J.; Althoff, D. und Stiller, C.: „A POMDP Maneuver Planner For Occlusions in Urban Scenarios“. In: *Intelligent Vehicles Symposium*. Paris, Frankreich: IEEE, Juni 2019, S. 2172–2179. doi: 10.1109/IVS.2019.8814179 (siehe S. 3).

[Iov22] Iovino, M.; Scukins, E.; Styrud, J.; Ögren, P. und Smith, C.: „A Survey of Behavior Trees in Robotics and AI“. Version 2. In: *Robotics and Autonomous Systems* 154 (Aug. 2022), S. 104096. doi: 10.1016/j.robot.2022.104096. arXiv: 2005.05842 (siehe S. 19).

[Isa08] Isaac, M.; Mark, C.; Dan, H.; Frank-Robert, K.; Aaron, N.; Lupashin Sergei; Catlin Jason; Schimpf Brian; Moran Pete; Zych Noah u. a.: „Team Cornell's Skynet: Robust Perception and Planning in an Urban Environment“. In: *Journal of Field Robotics* 25.8 (23. Juli 2008), S. 493–527. doi: 10.1002/rob.20253 (siehe S. 42, 43).

[ISO18] ISO International Organization for Standardization: Road Vehicles — Functional Safety. ISO 26262. Genf, Schweiz, Dez. 2018. url: https://www.iso.org/standard/68383.html (besucht am 19. 04. 2021) (siehe S. 35).

[ISO19] ISO International Organization for Standardization: Road Vehicles — Safety of the Intended Functionality. ISO/PAS 21448:2019. Genf, Schweiz, Jan. 2019. url: https://www.iso.org/standard/70939.html (siehe S. 164).

[ISO21] ISO International Organization for Standardization: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. ISO/SAE PAS 22736:2021. Genf, Schweiz, Aug. 2021. url: https://www.iso.org/standard/73766.html (siehe S. 35).

[Kir21] Kiran, B. R.; Sobh, I.; Talpaert, V.; Mannion, P.; Sallab, A. A. A.; Yogamani, S. und Pérez, P.: „Deep Reinforcement Learning for Autonomous Driving: A Survey“. In: *IEEE Transactions on Intelligent Transportation Systems* Early Access (9. Feb. 2021), S. 1–18. doi: 10.1109/TITS.2021.3054625 (siehe S. 5, 28).

[Kor20] Koren, I. und Krishna, C. M.: Fault-Tolerant Systems. 2. Aufl. Burlington, MA, USA: Morgan Kaufmann, 1. Sep. 2020. 416 S. doi: 10.1016/C2018-0-02160-X (siehe S. 32, 63).

[Lap95] Laprie, J.-C.: „Dependable Computing: Concepts, Limits, Challenges“. In: *International Symposium on Fault-Tolerant Computing*. Pasadena, California, USA: IEEE, 1995, S. 42–54 (siehe S. 32, 63).

[Lau10] Lauer, M.; Hafner, R.; Lange, S. und Riedmiller, M.: „Cognitive Concepts in Autonomous Soccer Playing Robots“. In: *Cognitive Systems Research* 11.3 (2010), S. 287–309. doi: 10.1016/j.cogsys.2009.12.003 (siehe S. 7, 21).

[Lee17] Lee, M.; Jo, K. und Sunwoo, M.: „Collision Risk Assessment for Possible Collision Vehicle in Occluded Area Based on Precise Map“. In: *International Conference on Intelligent Transportation Systems*. Yokohama, Japan: IEEE, Okt. 2017, S. 1–6. doi: 10.1109/ITSC.2017.8317666 (siehe S. 42).

[Lef14] Lefèvre, S.; Vasquez, D. und Laugier, C.: „A Survey on Motion Prediction and Risk Assessment for Intelligent Vehicles“. In: *ROBOMECH Journal* 1.1 (23. Juli 2014), S. 1. doi: 10.1186/s40648-014-0001-z (siehe S. 41).

[Mac09] Macek, K.; Govea, D. A. V.; Fraichard, T. und Siegwart, R.: „Towards Safe Vehicle Navigation in Dynamic Urban Scenarios“. In: *Automatika* 50.3-4 (1. Nov. 2009), S. 184–194. url: https://hal.inria.fr/inria-00447452/document (besucht am 16. 04. 2018) (siehe S. 42).

[Meg05] Meghanathan, S. B. N.: „A Survey of Contemporary Real-time Operating Systems“. In: *Informatica* 29.2 (2005), S. 233–240. URL: http://informatica.si/index.php/informatica/article/view/36 (besucht am 05. 07. 2021) (siehe S. 72).

[Mer18] Mercedes-Benz Research & Development North America, Inc. und Robert Bosch LLC: Reinventing Safety: A Joint Approach to Automated Driving Systems. Sunnyvale, CA, USA, Dez. 2018. URL: https://web.archive.org/https://www.daimler.com/innovation/case/autonomous/reinventing-safety.html (besucht am 07. 01. 2023) (siehe S. 35, 54).

[Mer21] Mercedes-Benz Online-Magazin: Hochautomatisiertes Fahren mit dem DRIVE PILOT. 29. Juli 2021. URL: https://www.daimler.com/magazin/technologie-innovation/einfach-technik-drive-pilot.html (besucht am 03. 12. 2021) (siehe S. 91).

[Mic85] Michon, J. A.: „A Critical View of Driver Behavior Models: What Do We Know, What Should We Do?“ In: *Human Behavior and Traffic Safety*. Boston, MA, USA: Springer US, 1985, S. 485–524. DOI: 10.1007/978-1-4613-2173-6_19 (siehe S. 11, 12).

[Mir18] Mirchevska, B.; Pek, C.; Werling, M.; Althoff, M. und Boedecker, J.: „High-Level Decision Making for Safe and Reasonable Autonomous Lane Changing Using Reinforcement Learning“. In: *International Conference on Intelligent Transportation Systems*. Maui, Hawaii, USA: IEEE, Nov. 2018, S. 2156–2162. DOI: 10.1109/ITSC.2018.8569448 (siehe S. 3).

[Moh22] Mohan, A. und Vaishnav, P.: „Impact of Automation on Long Haul Trucking Operator-Hours in the United States“. In: *Humanities and Social Sciences Communications* 9.1 (1 15. März 2022), S. 1–10. DOI: 10.1057/s41599-022-01103-w (siehe S. 2).

[Mon08] Montemerlo, M.; Becker, J.; Bhat, S.; Dahlkamp, H.; Dolgov, D.; Ettinger, S.; Haehnel, D.; Hilden, T.; Hoffmann, G.; Huhnke, B. u. a.: „Junior: The Stanford Entry in the Urban Challenge“. In: *Journal of Field Robotics* 25.9 (2008), S. 569–597. DOI: 10.1002/rob.20258 (siehe S. 6, 14–16).

[Mon99] Montenegro, S.: Sichere Und Fehlertolerante Steuerungen: Entwicklung Sicherheitsrelevanter Systeme. München, Deutschland: Carl Hanser Verlag, 1999 (siehe S. 32, 63).

[Mor82] Moret, B. M. E.: „Decision Trees and Diagrams“. In: *ACM Computing Surveys* 14.4 (1. Dez. 1982), S. 593–623. doi: 10.1145/356893.356898 (siehe S. 17).

[Nau18] Naumann, M.; Poggenhans, F.; Lauer, M. und Stiller, C.: „CoInCar-Sim: An Open-Source Simulation Framework for Cooperatively Interacting Automobiles“. In: *Intelligent Vehicles Symposium*. Changshu, Suzhou, China: IEEE, Juni 2018, S. 1–6. doi: 10.1109/IVS.2018.8500405 (siehe S. 113).

[NHT17] NHTSA National Highway Traffic Safety Administration: Automated Driving Systems 2.0: A Vision for Safety. Washington, DC, USA, Sep. 2017. url: https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/documents/13069a-ads2.0_090617_v9a_tag.pdf (besucht am 12. 01. 2021) (siehe S. 35).

[Obj17] Object Management Group, Inc: Unified Modeling Language. Standard 2.5.1. Milford, MA, USA, Dez. 2017, S. 796. url: https://www.omg.org/spec/UML/2.5.1 (besucht am 04. 05. 2022) (siehe S. 14).

[Ögr12] Ögren, P.: „Increasing Modularity of UAV Control Systems Using Computer Game Behavior Trees“. In: *Guidance, Navigation, and Control Conference*. Minneapolis, MN, USA: AIAA, Aug. 2012. doi: 10.2514/6.2012-4458 (siehe S. 7, 19).

[Ols16] Olsson, M.: „Behavior Trees for Decision-Making in Autonomous Driving“. Masterarbeit. Stockholm, Schweden: KTH, School of Computer Science and Communication, 2016. url: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-183060 (siehe S. 19).

[Orz18] Orzechowski, P. F.; Meyer, A. und Lauer, M.: „Tackling Occlusions & Limited Sensor Range with Set-based Safety Verification“. In: *International Conference on Intelligent Transportation*

Systems. Maui, HI, USA: IEEE, Nov. 2018, S. 1729–1736. DOI: 10.1109/ITSC.2018.8569332 (siehe S. 43).

[Orz20] Orzechowski, P. F.; Burger, C. und Lauer, M.: „Decision-Making for Automated Vehicles Using a Hierarchical Behavior-Based Arbitration Scheme". In: *Intelligent Vehicles Symposium*. Las Vegas, NV, USA: IEEE, Okt. 2020, S. 767–774. DOI: 10.1109/IV47402.2020.9304723 (siehe S. 7, 53, 61, 111, 114, 115, 137).

[Orz23] Orzechowski, P.; Fischer, J. und Stiller, C.: „Verhaltensentscheidung für Automatisches Fahren". In: *Handbuch Fahrerassistenzsysteme*. Hrsg. von Winner, H. Wiesbaden, Deutschland: Springer Fachmedien Wiesbaden, voraussichtliches Erscheinungsdatum 2023 (siehe S. 14).

[Pad16] Paden, B.; Čáp, M.; Yong, S. Z.; Yershov, D. und Frazzoli, E.: „A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles". In: *IEEE Transactions on Intelligent Vehicles* 1.1 (März 2016), S. 33–55. DOI: 10.1109/TIV.2016.2578706 (siehe S. 28).

[Pek18] Pek, C. und Althoff, M.: „Computationally Efficient Fail-safe Trajectory Planning for Self-driving Vehicles Using Convex Optimization". In: *International Conference on Intelligent Transportation Systems*. Maui, Hawaii, USA: IEEE, Nov. 2018, S. 1447–1454. DOI: 10.1109/ITSC.2018.8569425 (siehe S. 59, 91, 140).

[Pet13] Petrich, D.; Dang, T.; Kasper, D.; Breuel, G. und Stiller, C.: „Map-Based Long Term Motion Prediction for Vehicles in Traffic Environments". In: *Conference on Intelligent Transportation Systems*. Den Haag, Die Niederlande: IEEE, Okt. 2013, S. 2166–2172. DOI: 10.1109/ITSC.2013.6728549 (siehe S. 51).

[Pog18] Poggenhans, F.; Pauls, J.-H.; Janosovits, J.; Orf, S.; Naumann, M.; Kuhnt, F. und Mayr, M.: „Lanelet2: A High-Definition Map Framework for the Future of Automated Driving". In: *International Conference on Intelligent Transportation Systems*. Maui, Hawaii, USA: IEEE, Nov. 2018, S. 1672–1679. DOI: 10.1109/ITSC.2018.8569929 (siehe S. 50).

[Qui09] Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R. und Ng, A.: „ROS: An Open-Source Robot Operating System“. In: *ICRA Workshop on Open Source Software*. Bd. 3. Kobe, Japan, 2009. URL: http://www.robotics.stanford.edu/~ang/papers/icraoss09-ROS.pdf (siehe S. 164).

[Qui90] Quinlan, J. R.: „Decision Trees and Decision-Making“. In: *IEEE Transactions on Systems, Man, and Cybernetics* 20.2 (März 1990), S. 339–346. DOI: 10.1109/21.52545 (siehe S. 17).

[Rao92] Rao, A. S. und Georgeff, M. P.: „An Abstract Architecture for Rational Agents“. In: *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*. KR'92. San Mateo, CA, USA: Morgan Kaufmann Publishers Inc., 25. Okt. 1992, S. 439–449 (siehe S. 21).

[Ras20] Rasouli, A. und Tsotsos, J. K.: „Autonomous Vehicles That Interact With Pedestrians: A Survey of Theory and Practice“. In: *IEEE Transactions on Intelligent Transportation Systems* 21.3 (März 2020), S. 900–918. DOI: 10.1109/TITS.2019.2901817 (siehe S. 54).

[Ree90] Reeds, J. und Shepp, L.: „Optimal Paths for a Car That Goes Both Forwards and Backwards“. In: *Pacific Journal of Mathematics* 145.2 (1. Okt. 1990), S. 367–393. DOI: 10.2140/pjm.1990.145.367 (siehe S. 88).

[Rei10] Reif, K.: Fahrstabilisierungssysteme und Fahrerassistenzsysteme. Hrsg. von Reif, K. Wiesbaden, Deutschland: Vieweg+Teubner, 2010. DOI: 10.1007/978-3-8348-9717-6 (siehe S. 63).

[Ric19] Richter, S.; Wirges, S.; Königshof, H. und Stiller, C.: „Fusion of Range Measurements and Semantic Estimates in an Evidential Framework“. In: *tm - Technisches Messen* 86.s1 (1. Sep. 2019), S. 102–106. DOI: 10.1515/teme-2019-0052 (siehe S. 50).

[Ros16] Ross, H.-L.: Functional Safety for Road Vehicles: New Challenges and Solutions for E-mobility and Automated Driving. Cham, Schweiz: Springer International Publishing, 2016. DOI: 10.1007/978-3-319-33361-8 (siehe S. 35).

[Sad04] Sadou, M.; Polotski, V. und Cohen, P.: „Occlusions in Obstacle Detection for Safe Navigation". In: *Intelligent Vehicles Symposium*. Parma, Italien: IEEE, Juni 2004, S. 716–721. DOI: 10.1109/IVS.2004.1336472 (siehe S. 42, 43).

[SAE20] SAE International: AVSC Best Practice for Describing an Operational Design Domain: Conceptual Framework and Lexicon. AVSC00002202004. Warrendale, PA, USA, 15. Apr. 2020. URL: https://www.sae.org/standards/content/avsc00002202004/ (besucht am 26. 02. 2021) (siehe S. 35, 36).

[SAE21] SAE International: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. J3016. Warrendale, PA, USA, Apr. 2021, S. 41. URL: https://www.sae.org/standards/content/j3016_202104/ (besucht am 17. 05. 2021) (siehe S. 1, 35, 36, 163).

[Sal20] Salscheider, N.: „Simultaneous Object Detection and Semantic Segmentation". In: 9th International Conference on Pattern Recognition Applications and Methods. Valletta, Malta, 6. Okt. 2020, S. 555–561 (siehe S. 50).

[Sch14] Schäling, B.: The Boost C++ Libraries. 2nd Edition. Laguna Hills, CA, USA: XML Press, 22. Sep. 2014. 570 S. (siehe S. 14).

[Sch16] Schwesinger, U.; Bürki, M.; Timpner, J.; Rottmann, S.; Wolf, L.; Paz, L. M.; Grimmett, H.; Posner, I.; Newman, P.; Häne, C. u. a.: „Automated Valet Parking and Charging for E-Mobility". In: *Intelligent Vehicles Symposium*. Göteborg, Schweden: IEEE, Juni 2016, S. 157–164. DOI: 10.1109/IVS.2016.7535380 (siehe S. 88, 89).

[Sch18] Schwarting, W.; Alonso-Mora, J. und Rus, D.: „Planning and Decision-Making for Autonomous Vehicles“. In: *Annual Review of Control, Robotics, and Autonomous Systems* 1.1 (2018), S. 187–210. doi: 10.1146/annurev-control-060117-105157 (siehe S. 13, 14, 28).

[Sch20] Schwall, M.; Daniel, T.; Victor, T.; Favaro, F. und Hohnhold, H.: Waymo Public Road Safety Performance Data. 30. Okt. 2020. arXiv: 2011.00038. url: http://arxiv.org/abs/2011.00038 (besucht am 12. 01. 2021) (siehe S. 35).

[Sha17] Shalev-Shwartz, S.; Shammah, S. und Shashua, A.: On a Formal Model of Safe and Scalable Self-driving Cars. 21. Aug. 2017. arXiv: 1708.06374. url: https://arxiv.org/abs/1708.06374 (besucht am 12. 04. 2018) (siehe S. 8).

[Sic16] Siciliano, B. und Khatib, O., Hrsg.: Springer Handbook of Robotics. 2. Aufl. Springer Handbooks. Berlin, Deutschland: Springer-Verlag, 2016. doi: 10.1007/978-3-319-32552-1 (siehe S. 14).

[Son17] Sons, M.; Lauer, M.; Keller, C. G. und Stiller, C.: „Mapping and Localization Using Surround View“. In: *Intelligent Vehicles Symposium*. Redondo Beach, CA, USA: IEEE, Juni 2017, S. 1158–1163. doi: 10.1109/IVS.2017.7995869 (siehe S. 50).

[Ste20] Steyer, S.; Lenk, C.; Kellner, D.; Tanzmeister, G. und Wollherr, D.: „Grid-Based Object Tracking With Nonlinear Dynamic State and Shape Estimation“. In: *IEEE Transactions on Intelligent Transportation Systems* 21.7 (Juli 2020), S. 2874–2893. doi: 10.1109/TITS.2019.2921248 (siehe S. 50).

[Ste85] Stefik, M. und Bobrow, D. G.: „Object-Oriented Programming: Themes and Variations“. In: *AI Magazine* 6.4 (4 15. Dez. 1985), S. 40–40. doi: 10.1609/aimag.v6i4.508 (siehe S. 21).

[Taş17] Taş, Ö. Ş.; Hörmann, S.; Schäufele, B. und Kuhnt, F.: „Automated Vehicle System Architecture with Performance Assessment“. In: *International Conference on Intelligent Transportation Systems*. Yokohama, Japan: IEEE, Okt. 2017, S. 1–8. doi: 10.1109/ITSC.2017.8317862 (siehe S. 27, 37).

[Taş18a] Taş, Ö. Ş. und Stiller, C.: „Limited Visibility and Uncertainty Aware Motion Planning for Automated Driving“. In: *Intelligent Vehicles Symposium*. Changshu, China: IEEE, Juni 2018, S. 1171–1178. doi: 10.1109/IVS.2018.8500369 (siehe S. 42).

[Taş18b] Taş, Ö. Ş.; Salscheider, N. O.; Poggenhans, F.; Wirges, S.; Bandera, C.; Zofka, M. R.; Strauss, T.; Zöllner, J. M. und Stiller, C.: „Making Bertha Cooperate–Team AnnieWAY's Entry to the 2016 Grand Cooperative Driving Challenge“. In: *IEEE Transactions on Intelligent Transportation Systems* 19.4 (Apr. 2018), S. 1262–1276. doi: 10.1109/TITS.2017.2749974 (siehe S. 111).

[The20a] The MathWorks, Inc: MATLAB Stateflow. Natick, MA, USA, 2020. url: https://www.mathworks.com/products/stateflow.html (besucht am 07. 05. 2022) (siehe S. 14).

[The20b] The Qt Company Ltd.: The State Machine Framework. Version 5.15. Espoo, Finnland, Nov. 2020. url: https://doc.qt.io/qt-5/statemachine-api.html (besucht am 07. 05. 2022) (siehe S. 14).

[Tre00] Treiber, M.; Hennecke, A. und Helbing, D.: „Congested Traffic States in Empirical Observations and Microscopic Simulations“. In: *Physical Review E* 62.2 (1. Aug. 2000), S. 1805–1824. doi: 10.1103/PhysRevE.62.1805 (siehe S. 30).

[Ulb15] Ulbrich, S.; Menzel, T.; Reschka, A.; Schuldt, F. und Maurer, M.: „Definition Der Begriffe Szene, Situation Und Szenario Für Das Automatisierte Fahren“. In: *10. Workshop Fahrerassistenzsysteme FAS*. Walting, Deutschland, 2015, S. 105 (siehe S. 50).

[Völ18] Völklein, M.: „Busfahrer verzweifelt gesucht“. In: *Süddeutsche.de* (26. Feb. 2018). URL: https://www.sueddeutsche.de/karriere/busfahrer-mangel-bitte-einsteigen-1.3878529 (besucht am 26. 03. 2022) (siehe S. 2).

[Vos16] Vossen, G. und Witt, K.-U.: Grundkurs Theoretische Informatik: Eine anwendungsbezogene Einführung - Für Studierende in allen Informatik-Studiengängen. 6. Aufl. Wiesbaden, Deutschland: Springer Vieweg, 2016. DOI: 10.1007/978-3-8348-2202-4 (siehe S. 14).

[Voß21] Voßwinkel, R.; Gerwien, M.; Jungmann, A. und Schrödel, F.: „Intelligent Decision-Making and Motion Planning for Automated Vehicles“. In: *Autonomous Driving and Advanced Driver-Assistance Systems (ADAS)*. Boca Raton, FL, USA: CRC Press, 2021 (siehe S. 14).

[Wag06] Wagner, F.; Schmuki, R.; Wagner, T. und Wolstenholme, P.: Modeling Software with Finite State Machines: A Practical Approach. Boca Raton, FL, USA: CRC Press, 15. Mai 2006. 391 S. (siehe S. 14).

[Way20a] Waymo LLC: Waymo Safety Report. Mountain View, CA, USA, Sep. 2020. URL: https://waymo.com/safety/safety-report (besucht am 12. 01. 2021) (siehe S. 35, 37, 68, 85, 167).

[Way20b] Waymo LLC: Waymo's Safety Methodologies and Safety Readiness Determinations. Mountain View, CA, USA, Okt. 2020. URL: https://waymo.com/safety/methodologies (besucht am 12. 01. 2021) (siehe S. 35, 39, 40).

[Wer10] Werling, M.; Ziegler, J.; Kammel, S. und Thrun, S.: „Optimal Trajectory Generation for Dynamic Street Scenarios in a Frenét Frame“. In: *International Conference on Robotics and Automation*. Anchorage, AK, USA: IEEE, Mai 2010, S. 987–993. DOI: 10.1109/ROBOT.2010.5509799 (siehe S. 28, 29, 56).

[Wer17] Werling, M.: Optimale aktive Fahreingriffe, für Sicherheits- und Komfortsysteme in Fahrzeugen. Berlin, Deutschland: De Gruyter Oldenbourg, 2017. doi: 10.1515/9783110531923 (siehe S. 4, 27, 28, 30).

[Win15] Winner, H. und Schopper, M.: „Adaptive Cruise Control“. In: *Handbuch Fahrerassistenzsysteme*. Hrsg. von Winner, H.; Hakuli, S.; Lotz, F. und Singer, C. ATZ/MTZ-Fachbuch. Wiesbaden, Deutschland: Springer Fachmedien, 2015, S. 851–891. doi: 10.1007/978-3-658-05734-3_46 (siehe S. 91).

[Wör05] Wörn, H.: Echtzeitsysteme: Grundlagen, Funktionsweisen, Anwendungen. eXamen.press. Berlin, Deutschland: Springer-Verlag, 2005. doi: 10.1007/b139050 (siehe S. 65, 72).

[Yur20] Yurtsever, E.; Lambert, J.; Carballo, A. und Takeda, K.: „A Survey of Autonomous Driving: Common Practices and Emerging Technologies“. In: *IEEE Access* 8 (2020), S. 58443–58469. doi: 10.1109/ACCESS.2020.2983149 (siehe S. 14).

[Zen20] Zeng, W.; Wang, S.; Liao, R.; Chen, Y.; Yang, B. und Urtasun, R.: „DSDNet: Deep Structured Self-Driving Network“. In: *Computer Vision – ECCV 2020*. Hrsg. von Vedaldi, A.; Bischof, H.; Brox, T. und Frahm, J.-M. Bd. 12366. Lecture Notes in Computer Science. Cham, Schweiz: Springer International Publishing, 2020, S. 156–172 (siehe S. 56).

[Zha16] Zhan, W.; Liu, C.; Chan, C. Y. und Tomizuka, M.: „A Non-Conservatively Defensive Strategy for Urban Autonomous Driving“. In: *International Conference on Intelligent Transportation Systems*. Rio de Janeiro, Brasilien: IEEE, Nov. 2016, S. 459–464. doi: 10.1109/ITSC.2016.7795595 (siehe S. 42).

[Zha17] Zhang, L.; Merrifield, R.; Deguet, A. und Yang, G.-Z.: „Powering the World's Robots—10 Years of ROS“. In: *Science Robotics* 2.11 (25. Okt. 2017), S. 1–5. doi: 10.1126/scirobotics.aar1868 (siehe S. 72).

[Zie14a] Ziegler, J.; Bender, P.; Dang, T. und Stiller, C.: „Trajectory Planning for Bertha – A Local, Continuous Method“. In: *Intelligent Vehicles Symposium*. Dearborn, MI, USA: IEEE, Juni 2014, S. 450–457. doi: 10.1109/IVS.2014.6856581 (siehe S. 4, 14, 29, 31, 51, 56).

[Zie14b] Ziegler, J.; Bender, P.; Schreiber, M.; Lategahn, H.; Strauss, T.; Stiller, C.; Dang, T.; Franke, U.; Appenrodt, N.; Keller, C. G. u. a.: „Making Bertha Drive – An Autonomous Journey on a Historic Route“. In: *IEEE Intelligent Transportation Systems Magazine* 6.2 (23. Apr. 2014), S. 8–20. doi: 10.1109/MITS.2014.2306552 (siehe S. 6, 14).

Abkürzungsverzeichnis

ACC	Adaptive Cruise Control [SAE21]
BDI	Belief-Desire-Intention
EKF	Elektrokleinstfahrzeug
FMEA	Fehlzustandsart- und -auswirkungsanalyse [DIN06]
FTA	Fehlzustandsbaumanalyse [DIN07]
GPS	Globales Positionsbestimmungssystem (engl. „Global Positioning System“)
HMI	Mensch-Maschine-Schnittstelle (engl. „Human Machine Interface“)
IDM	Intelligent Driver Model
KFZ	Kraftfahrzeug
KIT	Karlsruher Institut für Technologie
LKW	Lastkraftwagen
MRC	Risikominimaler Zustand (engl. „Minimal Risk Condition“)
MRM	Risikominimierendes Manöver (engl. „Minimal Risk Maneuver“)
NHTSA	National Highway Traffic Safety Administration
ODD	engl. „Operational Design Domain“ [SAE21]
PKW	Personenkraftwagen

POMDP	Partiell beobachtbares Markow-Entscheidungsproblem (engl. „Partially Observable Markov Decision Process“)
PRM*	Probabilistische Straßenkarten (engl. „Probabilistic Roadmaps“)
RL	Bestärkendes Lernen (engl. „Reinforcement Learning“)
ROS	Robot Operating System [Qui09]
RRT*	Rapidly-exploring Random Trees
SOTIF	engl. „Safety of the Intended Functionality“ [ISO19]
V2F	Kommunikation zur Flottenverwaltung (engl. „Vehicle to Fleet Management Communication“)
V2I	Fahrzeug-Infrastruktur-Kommunikation (engl. „Vehicle to Infrastructure Communication“)
V2V	Fahrzeug-Fahrzeug-Kommunikation (engl. „Vehicle to Vehicle Communication“)
V2X	Fahrzeug-zu-X-Kommunikation, Sammelbegriff für V2V, V2I und V2F
VRU	vulnerabler Verkehrsteilnehmer (engl. „Vulnerable Road User“)

A Verhaltenskompetenzen

Folgende grundlegende Verhaltenskompetenzen wurden von der NHTSA und Waymo zusammengetragen [Way20a]:

1. Detect and Respond to Speed Limit Changes and Speed Advisories
2. Perform High-Speed Merge (e.g., Freeway)
3. Perform Low-Speed Merge
4. Move Out of the Travel Lane and Park (e.g., to the Shoulder for Minimal Risk)
5. Detect and Respond to Encroaching Oncoming Vehicles
6. Detect Passing and No Passing Zones and Perform Passing Maneuvers
7. Perform Car Following (Including Stop and Go)
8. Detect and Respond to Stopped Vehicles
9. Detect and Respond to Lane Changes
10. Detect and Respond to Static Obstacles in the Path of the Vehicle
11. Detect Traffic Signals and Stop/Yield Signs
12. Respond to Traffic Signals and Stop/Yield Signs
13. Navigate Intersections and Perform Turns
14. Navigate Roundabouts
15. Navigate a Parking Lot and Locate Spaces
16. Detect and Respond to Access Restrictions (One-Way, No Turn, Ramps, etc.)
17. Detect and Respond to Work Zones and People Directing Traffic in Unplanned or Planned Events
18. Make Appropriate Right-of-Way Decisions
19. Follow Local and State Driving Laws

20 Follow Police/First Responder Controlling Traffic (Overriding or Acting as Traffic Control Device)
21 Follow Construction Zone Workers Controlling Traffic Patterns (Slow/Stop Sign Holders)
22 Respond to Citizens Directing Traffic After a Crash
23 Detect and Respond to Temporary Traffic Control Devices
24 Detect and Respond to Emergency Vehicles
25 Yield for Law Enforcement, EMT, Fire, and Other Emergency Vehicles at Intersections, Junctions, and Other Traffic Controlled Situations
26 Yield to Pedestrians and Bicyclists at Intersections and Crosswalks
27 Provide Safe Distance From Vehicles, Pedestrians, Bicyclists on Side of the Road
28 Detect/Respond to Detours and/or Other Temporary Changes in Traffic Patterns
29 Moving to a Minimal Risk Condition When Exiting the Travel Lane is Not Possible
30 Perform Lane Changes
31 Detect and Respond to Lead Vehicle
32 Detect and Respond to a Merging Vehicle
33 Detect and Respond to Pedestrians in Road (Not Walking Through Intersection or Crosswalk)
34 Provide Safe Distance from Bicyclists Traveling on Road (With or Without Bike Lane)
35 Detect and Respond to Animals
36 Detect and Respond to Motorcyclists
37 Detect and Respond to School Buses
38 Navigate Around Unexpected Road Closures (e.g. Lane, Intersection, etc.)
39 Navigate Railroad Crossings
40 Make Appropriate Reversing Maneuvers
41 Detect and Respond to Vehicle Control Loss (e.g. reduced road friction)

42 Detect and Respond to Conditions Involving Vehicle, System, or Component-Level Failures or Faults (e.g. power failure, sensing failure, sensing obstruction, computing failure, fault handling or response)

43 Detect and Respond to Unanticipated Weather or Lighting Conditions Outside of Vehicle's Capability (e.g. rainstorm)

44 Detect and Respond to Unanticipated Lighting Conditions (e.g. power outages)

45 Detect and Respond to Non-Collision Safety Situations (e.g. vehicle doors ajar)

46 Detect and Respond to Faded or Missing Roadway Markings or Signage

47 Detect and Respond to Vehicles Parking in the Roadway