CARL VON OSSIETZKY universität OLDENBURG

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik

Statistisches Model Checking mittels geführter Simulation

im Kontext modellbasierter Entwicklung sicherheitskritischer Fahrerassistenzsysteme

Dissertation zur Erlangung des Grades eines

Doktors der Ingenieurwissenschaften

vorgelegt von

Dipl.-Inform. Stefan Puch

Gutachter:

Prof. Dr. Martin Fränzle
Prof. Dr.-Ing. Axel Hahn

Tag der Disputation:

19. Juni 2019

Kurzfassung

Das Statistische Bundesamt nennt für das Jahr 2017 2,6 Millionen polizeilich registrierte Unfälle und 3.180 Verkehrstote. Damit sterben durchschnittlich nahezu 9 Menschen täglich im deutschen Straßenverkehr, wobei in 34% der Fälle „nicht angepasste Geschwindigkeit“ und damit menschliches Fehlverhalten ursächlich ist.

Der derzeitige Ansatz der Automobilindustrie, um die Unfallrate weiter zu reduzieren, liegt in der Entwicklung von vernetzten Fahrerassistenzsystemen zum hochautomatisierten Fahren, bei denen der Fahrer nicht „in the Loop“ gehalten werden muss. Die Automobilindustrie sucht in diesem Zusammenhang nach verbesserten bzw. neuen Entwicklungs- und Testmethoden, mit denen früh im Entwicklungsprozess Prototypen simulativ analysiert und evaluiert werden können. Die simulative Analyse eines Unfalls mit Personenschaden, bei denen Probanden in einem Fahrsimulator die Funktionsweise eines Fahrerassistenzsystems erproben, ist allerdings ineffektiv, da diese Situation aus Sicht eines Autofahrers einerseits ein extrem seltenes sicherheitskritisches Ereignis darstellt und andererseits nicht mehrfach wiederholt werden kann (z. B. aufgrund des Überraschungseffekts). Computerausführbare Fahrermodelle, welche an die Stelle eines Probanden treten, bieten einen alternativen Ansatz.

Die vorliegende Arbeit führt in diesem Kontext eine Methodik ein, welche mit Statistischem Model Checking im Rahmen von geführter Simulation seltene Ereignisse erfassen kann. Dazu wird zunächst ein Kosimulationsframework auf Basis der High Level Architecture entwickelt, welches eine Kosimulation bestehend aus Fahrermodell, Fahrsimulationssoftware und Fahrerassistenzsystem ermöglicht. Anschließend wird der auf Adaptive Importance Sampling basierende Threshold Uncertainty Tree Search (TUTS) Algorithmus beschrieben, welcher mit Hilfe einer Kritikalitätsfunktion – abgeleitet aus Expertenwissen – eine Simulation führen kann. Seine Funktionsweise wird anhand eines Fahrer-Fahrzeug-Assistenzsystem-Szenarios demonstriert, in dem seltene Ereignisse mit einer Wahrscheinlichkeit deutlich kleiner als 10^{-9} erfasst werden. Nach einer mathematischen Rückrechnung der geführten Simulationsergebnisse erfolgt eine quantitative Abschätzung mittels 99%igem Konfidenzniveaus, mit welcher Sicherheit das vorliegende Ergebnis zutreffend ist. Zum Abschluss wird eine Übertragbarkeit der Methodik auf weitere Domänen anhand eines Benchmarks dargelegt. Der Benchmark, bei dem die Wahrscheinlichkeit zu schätzen ist, mit der ein zufällig springender Ball ein winziges Loch trifft, zeigt, dass der TUTS-Algorithmus im direkten Vergleich zu einer Implementierung der Cross-Entropy Methode zumindest bei weniger als 2.000 Simulationsläufen genauer ist.

Abstract

For 2017, the German Federal Statistical Office announced 2.6 million accidents were registered by the police and 3,180 people lost their lives. On average, almost 9 people die every day in German road traffic, whereby in 34% of these cases „unadjusted speed“ and thus human error was the main cause.

The current approach of the automotive industry to further reduce the accident rate is to develop interconnected driver assistance systems for highly automated driving, where the driver does not always has to be „in the loop“. In this context, the automotive industry is looking for improved or new development and test methods which allow to simulate and analyse prototypes as early as possible during the development process. The analysis through simulation of an accident resulting in injury, where subjects can test the functionality of a driver assistance system in a driving simulator is however not effective, since from the driver's point of view such a situation represents an extremely rare safety-critical event and thus cannot be repeated several times (e. g. for reasons of surprise). Computer-executable driver models which take the place of a subject offer an alternative approach.

In this context, the present thesis introduces a methodology which can capture rare events using Statistical Model Checking within the scope of guided simulation. At first a co-simulation framework based on the High Level Architecture will be developed which enables a co-simulation consisting of driver model, driving simulation software and driver assistance system. Subsequently, the Threshold Uncertainty Tree Search (TUTS) algorithm, which is based on Adaptive Importance Sampling, will be described. It can guide a simulation with the help of a criticality function – derived from expert knowledge – into rare situations. Its functionality is demonstrated using a driver-vehicle assistance system scenario in which rare events are recorded with a probability significantly less than 10^{-9}. After a mathematical recalculation of the simulation results, a quantitative estimation is made using a 99% confidence level in order to obtain an estimate if the existing results are applicable. Finally, the transferability of the methodology to other domains by means of a benchmark is presented. For the benchmark, it has to be estimated by which probability a randomly bouncing ball hits a tiny hole. It turns out the TUTS algorithm is more accurate in direct comparison to an implementation of the cross-entropy method at least with less than 2,000 simulation runs.

Danksagung

Geschafft!!! Der Doktortitel ist in Deutschland der höchster akademische Grad, welcher mit einer erfolgreichen Promotion verliehen wird und die Fähigkeit zum selbstständigen wissenschaftlichen Arbeiten belegt.

Glücklicherweise konnte ich bei der Verfassung dieser Dissertation auf die Hilfe und Unterstützung von zahlreichen Personen zurückgreifen, denen ich an dieser Stelle explizit „Danke“ sagen möchte.

Ohne meinen Doktorvater Prof. Dr. Martin Fränzle würde diese Arbeit nicht existieren. Als ich direkt im Anschluss an meine Diplomarbeit herzlichst in seine Arbeitsgruppe aufgenommen wurde, hatte ich große Zweifel, ob ich überhaupt zu eigenständigem wissenschaftlichen Arbeiten geeignet bin. Doch während der folgenden Jahre wurde ich stetig von ihm mit Ratschlägen und Ideen ermutigt, den eingeschlagenen Weg fortzusetzen. Auf teils geniale Weise wurden mir mögliche Auswege aus gefühlten Sackgassen aufgezeigt, ohne jemals eine Entscheidung vorwegzunehmen. Auch wenn diese Situationen niemals einfach waren, bin ich für die Erfahrungen, welche mir durch diese großen Herausforderungen zuteilwurden, im Nachhinein besonders dankbar!

Weiterhin möchte ich Prof. Dr.-Ing. Axel Hahn danken, der mir eine zweite professorale Meinung unter einem ingenieurmäßigen Blickwinkel offeriert hat, sowie Prof. Dr. Sebastian Lehnhoff und Dr. Ingo Stierand als weiteren Mitgliedern meiner Prüfungskommission.

Ich möchte an dieser Stelle all meinen Kollegen, die ich gar nicht alle einzeln aufzählen kann, in der Arbeitsgruppe Hybride Systeme, der Universität und dem OFFIS e.V. für die vielen Stunden wissenschaftlicher Diskussion und den Austausch bereichernder sozialer Aspekte danken. Hinsichtlich dieser Arbeit möchte ich allerdings für die besonders gute wissenschaftliche Zusammenarbeit bei meinen Publikationen und die hervorragende Vertretung meiner selbst bei der Vorstellung und Präsentation der erzielten Ergebnisse auf Konferenzen Jan-Patrick Osterloh, Dr. Bertram Wortelen, Dr. Sebastian Gerwinn und Paul Kröger namentlich erwähnen. Auch Günter Ehmen und Björn Koopmann, welche mir insbesondere in der letzten Phase meiner Arbeit den Rücken von der Projektarbeit freigehalten haben, gebührt ein aufrichtiger Dank. Nicht zu vergessen sind in diesem Zusammenhang auch Lars Weber sowie mein Bruder Martin für ihr wertvolles Lektorat meiner Prototypen.

Erwähnen möchte ich an dieser Stelle auch meine engsten Freunde, die mir geholfen haben, zumindest zeitweise die Gedanken des Arbeitsalltages zu vergessen, indem sie unermüdlich darauf hingewiesen haben, dass es ein Leben abseits der wissenschaftlichen Fragestellungen gibt.

Abschließend ist allerdings von ganzem Herzen meiner Familie zu danken: Zunächst meinen Eltern, die ihrem Sohn, was immer auch anstand, mit Rat und Tat zur Seite gestanden haben. Meiner Ehefrau Antje, die sich in liebevoller und verlässlicher Weise um den noch jungen Familiennachwuchs gekümmert hat, dafür sorgte, dass der Kühlschrank zu Hause niemals leer wurde und dennoch den einen oder anderen wohl verdienten Feierabend alleine auf dem Sofa verbringen musste, anstatt in trauter Zweisamkeit. Gemäß dem Motto „das Beste am Schluss“ dürfen meine beiden Töchter Finja und Lina nicht fehlen. Auch wenn sie in der heißen Phase des Schreibens öfters ohne ihren Papa ins Bett gehen mussten, waren sie in Gedanken stets bei ihm. Seid versichert: „Papa hat Euch immer lieb, komme was wolle!“

Stefan Puch
Oldenburg, im Oktober 2019

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Kontext und Motivation

„Das Auto ist des Deutschen liebstes Kind“ heißt es im deutschen Volksmund. Und genau wie verantwortungsvolle Eltern ihren leiblichen Sprössling vor Unfällen jeglicher Art zu schützen versuchen, ist die vergleichbare Aufgabe für den materiellen Nachwuchs mit viel Mühe verbunden. Das Statistische Bundesamt nennt für 2017 20.928 polizeilich registrierte Unfälle mit Personenschaden allein auf deutschen Autobahnen, wobei 409 Menschen ums Leben kamen. Bei einem Blick auf den gesamten deutschen Straßenverkehr steigt die Zahl der polizeilich aufgenommenen Unfälle auf 2,6 Millionen, die Anzahl der Getöteten liegt bei 3.180 und 390.312 Personen wurden leicht bis schwer verletzt [Sta17b, S. 44 - 54]. Zwar liegt die Anzahl der getöteten Personen deutlich unter den Werten von 1950 bis 2012, dennoch sterben durchschnittlich nahezu 9 Menschen täglich im deutschen Straßenverkehr und obwohl eine Abwägung von Menschenleben gegen einen Wert ethisch schwierig ist, beziffert die Bundesanstalt für Straßenwesen den volkswirtschaftlichen Schaden mit etwa einer Million Euro pro Unfalltotem [BKW10].

Bei einer Analyse der Ursachen nennt das Statistische Bundesamt „nicht angepasste Geschwindigkeit“ als häufigsten Grund für tödliche Verkehrsunfälle: „Damit starb mehr als jeder Dritte (34%) aller im Straßenverkehr Getöteten bei Unfällen aufgrund nicht angepasster Geschwindigkeit.“ [Sta17a, S. 14], auf Autobahnen war es beinahe jeder Zweite [Sta17a, S. 16]. Was die konkreten Zahlen und Fakten beim ersten Blick vermeintlich schwer erkennen lassen, ist der Umstand, dass nicht etwa technisches Versagen oder allgemeine Ursachen (z. B. Straßenverhältnisse, Witterungseinflüsse) die Hauptursache sind, sondern menschliches Fehlverhalten. Wird an dieser Stelle einmal von vorsätzlichem Fehlverhalten abgesehen, können unzählige Ablenkungen dazu führen, dass Geschwindigkeitsvorgaben übersehen werden oder die Geschwindigkeit für die gegebene Straßen- bzw. Wettersituation nicht angemessen angepasst wird. Denn der Fahrer eines Kraftfahrzeuges befindet sich in einer hochkomplexen dynamischen Umgebung, bei der viele unterschiedliche Aufgaben Aufmerksamkeit verlangen. In einem modernen Fahrzeug strömen nicht nur die Reize der aktuellen

Verkehrssituation, sondern auch zahlreiche Stimuli von Komfort- und Infotainmentsystemen auf den Fahrzeugführer ein, welche während der eigentlichen Fahraufgabe gefiltert und bewältigt werden müssen.

Der derzeitige Ansatz der Automobilindustrie, durch Entlastung bzw. Unterstützung des Fahrers die Unfallrate und damit direkt auch die Anzahl der Verkehrstoten trotz jährlich sinkender Werte auch zukünftig zu reduzieren, liegt in der Entwicklung einer stetig zunehmenden und besser vernetzten Anzahl von Fahrerassistenzsystemen (FAS). Diese Fahrerassistenzsysteme sollen den Fahrzeugführer nicht nur während der routinemäßigen Fahraufgaben wie beispielsweise Längs- und Querführung des Fahrzeugs unterstützen (z. B. Abstandsregeltempomat, Spurwechselassistent), sondern auch in Extremsituationen zuverlässig helfen (z. B. Notbremsassistent).

Studien, wie die von Hoffman [Hof12] oder Hummel [Hum+11], zur allgemeinen Wirksamkeit von Fahrerassistenzsystemen belegen, „dass moderne FAS in der Lage sind, das Schaden- bzw. Unfallgeschehen (Unfälle mit Personenschaden und einem Schadenaufwand von 15.000 Euro und mehr) positiv zu beeinflussen“ [Hum+11, S. 5]. Eine Gemeinsamkeit der dazu untersuchten FAS ist jedoch die Tatsache, dass der Fahrer eines Fahrzeugs während des Betriebs des FAS stets im Loop gehalten werden muss. Das bedeutet, dass das FAS einer fortwährenden Überwachung (Monitoring) seitens des Benutzers bedarf, für welche Aufmerksamkeit erforderlich ist. Da der Fahrer aber gleichzeitig auch das unzuverlässigste Glied im Regelkreis bestehend aus Umwelt, Fahrzeug, FAS und Fahrzeugführer darstellt [Sta17a, S. 11], ist für die Entwicklung neuer FAS eine der größten Herausforderungen der Schritt hin zu höheren Automatisierungsgraden (hoch- bzw. vollautomatisiert), für die der Fahrer nicht mehr im Loop gehalten werden muss. Die damit einhergehenden rechtlichen Voraussetzungen müssen jedoch erst noch geschaffen werden (Haftungsrecht, Zulassungsrecht, etc.), ebenso wie die Realisierung neuer Entwicklungs- und Testabläufe, welche den extrem hohen Sicherheitsanforderungen dieser Systeme gerecht werden [BB12].

Die derzeit etablierten Entwicklungs- und Testabläufe bei Fahrerassistenzsystemen, welche den Faktor Mensch direkt mit einbeziehen, setzen auf Simulatorstudien mit Versuchspersonen [ZBS16; Wan+18]. In zahlreichen Versuchsfahrten im Fahrsimulator werden dabei Prototypen von Fahrerassistenzsystemen evaluiert und validiert, bevor zur Absicherung der Ergebnisse Fahrten in Versuchsfahrzeugen im Feld durchgeführt werden. Das Forschungsprojekt Ko-HAF beispielsweise nennt nach Ende der Projektlaufzeit 33 empirische Studien mit 1.723 Teilnehmern, welche in über 1.750 Stunden durchgeführt wurden [Ben18]. Der benötigte Zeitaufwand ist einerseits enorm, andererseits ist die Anzahl von 14 Probanden, welche jeweils mit 6 kritischen Fahrsituationen konfrontiert wurden, sehr gering [Wan+18].

Mit Hilfe von computerausführbaren Fahrermodellen lässt sich der notwendige Zeitaufwand teilweise reduzieren und die Anzahl an Versuchsfahrten erhöhen. Fahrermodelle lassen sich ebenso wie Versuchspersonen in Simulatorstudien einsetzen. Sie simulieren dabei sowohl

interne Prozesse eines Menschen hinsichtlich der Informationswahrnehmung, -verarbeitung und -speicherung als auch externe Handlungen, wie zum Beispiel das Lenkverhalten eines Autofahrers beim Führen eines Fahrzeugs. Außerdem lassen sich mit dem Einsatz von Fahrermodellen Simulatorstudien ergänzen und automatisieren, denn ein Fahrermodell braucht im Gegensatz zu Versuchsprobanden weder eine Eingewöhnungsphase noch eine Erholungspause.

Ein Problem beim Einsatz von Fahrermodellen ist aber, wie bei jedem anderen Modell auch, ihr Gültigkeitsbereich. Modelle stellen immer eine Abstraktion des realen Weltbildes dar und die ohnehin sehr komplexen und teils schwer beobachtbaren mentalen Prozesse des menschlichen Verhaltens sind immer noch in weiten Teilen unerforscht. Aufgrund dieser Einschränkung sind Simulationen mit kognitiven Fahrermodellen derzeit auch nicht als Ersatz, sondern lediglich als Ergänzung bei der Assistenzsystementwicklung in Betracht zu ziehen. Doch selbst unter der Prämisse der Verfügbarkeit adäquater ausführbarer, kognitiver Fahrermodelle, die stellvertretend für einen menschlichen Fahrer im Rahmen von Simulationen eingesetzt werden können, eröffnet sich bei genauerer Betrachtung ein derzeit noch weitestgehend ungelöstes Problemfeld:

Trotz der durchschnittlich 9 tödlich verunglückten Menschen täglich kommt die Mehrheit der deutschen Straßenverkehrsteilnehmer unbeschadet durch den Straßenverkehr. So war nach dem Statistischen Bundesamt im Durchschnitt der Jahre 2005 bis 2009 ein Pkw-Fahrer im Durchschnitt alle 1,46 Millionen Kilometer in einen Unfall mit Personenschaden involviert [Vor10]. Bei einer durchschnittlichen jährlichen Kilometerleistung von 18.693 für eine Einzelperson ist so ein Unfall demnach ein extrem seltenes sicherheitskritisches Ereignis [AXA09]. Für einen Assistenzsystementwickler, der den Prototyp seines neuen Fahrerassistenzsystems zur weiteren Reduzierung der Unfallrate mit Hilfe einer Fahrsimulationssoftware und einem ausführbaren Fahrermodell simulationsbasiert evaluieren möchte, bedeuten diese Zahlen hingegen, dass er bei einem rein Monte-Carlo basierten Simulationasansatz[1] sein Assistenzsystem mit sehr hoher Wahrscheinlichkeit niemals in Aktion sehen wird. Wünschenswert wäre an dieser Stelle eine Technik, die es dem Entwickler erlaubt, Situationen, für deren Einsatzzweck ein FAS entwickelt wurde, gezielt anzusteuern. Dabei sind vor allem Situationen von Interesse, die aufgrund ihrer Verteilung extrem selten zu beobachten sind. Das können beispielsweise strukturbedingte Randbereiche einzelner Modelle sein, aber auch Konstellationen, die erst durch die Kombination heterogener Systemklassen – wie Fahrsimulationssoftware, kognitives Fahrermodell und automatenbasiertes Fahrerassistenzsystem – entstehen.

[1] Für eine Einführung siehe [WH16].

1.2 Wissenschaftlicher Beitrag und methodisches Vorgehen

Motiviert durch den Bedarf in der Automobilindustrie nach speziellen Entwicklungs- und Testabläufen im Bereich der Fahrerassistenzsystementwicklung [BB12] soll in dieser Arbeit untersucht werden, inwieweit sich Ansätze aus dem Bereich der Rare-event Simulation [JS06; ZBC12] oder der Statistischen Modellprüfung [You+06; LDB10] in diesen Bereich übertragen lassen. Stellvertretend für die Entwicklungs- und Testabläufe wird in dieser Arbeit ein simulationsbasierter Ansatz als Ausgangspunkt gewählt. Ziel ist es, mit einer deutlich reduzierten Anzahl an Simulationsläufen im Gegensatz zu einer reinen Monte-Carlo-Simulation extrem seltene Ereignisse simulieren zu können. Dabei sollen die originalen Modellwahrscheinlichkeiten im Ergebnis erhalten bleiben. Ferner soll eine Abschätzung gegeben werden, mit welcher Sicherheit das vorliegende Ergebnis zutreffend ist. Ausgangsbasis ist eine Kosimulation bestehend aus heterogenen domänentypischen Simulatoren im eingeschränkten Kontext der Fahrerassistenzsystementwicklung. Die kognitive Architektur CASCaS [Lüd+09] stellt dazu ein Fahrermodell zur Verfügung und Fahrsimulationssoftware wie SILAB [STS11; WIV19] ermöglichen die Simulation von Verkehrsszenarien einschließlich Fahrzeugdynamik und Umgebungsverkehr. Eine einfache Integration prototypischer Fahrerassistenzsysteme ist entweder über die APIs der Fahrsimulationssoftware oder als eigenständige Applikation möglich.

Der wissenschaftliche Mehrwert dieser Arbeit leitet sich direkt aus der oben genannten praktischen Relevanz ab. Zum einen aus der Einbeziehung eines kognitiven Fahrermodells im Rahmen der Fahrerassistenzsystementwicklung und zum anderen mit der Möglichkeit, mittels geführter Simulation Techniken aus dem Bereich der Statistischen Modellprüfung in diesem Kontext nutzbar zu machen.

Diese Dissertation beinhaltet nur wenige Inhalte, welche nicht schon in vorhergehenden Arbeiten wie Konferenzbänden, Workshops oder auf Postern vom Autor zusammen mit anderen Wissenschaftlern veröffentlicht wurden. Damit einher gehen Anregungen und Ideen, welche in zahlreichen Diskussionen entstanden sind, sowie Bewertungen in Form von Interpretationen der Ergebnisse. Eingeflossen sind damit direkt bzw. indirekt auch Arbeiten von anderen Wissenschaftlern im Rahmen von Softwaretools durch Implementierungen, Benchmarks und Dokumentationen im Quellcode. Da es im Rahmen von Forschungsprojekten gängige Praxis ist, schon während der Projektlaufzeit Konzepte und (Zwischen-)Ergebnisse zu veröffentlichen, besteht diese Arbeit zumeist aus Inhalten, die bereits vorab (in der Regel nach einem Review-Prozess) veröffentlicht wurden, wenn auch nicht in dem vollen Detailgrad, wie es im Rahmen dieser Arbeit möglich ist. Auf diesen Umstand soll an dieser Stelle noch einmal deutlichst hingewiesen werden. Die einzelnen Veröffentlichungen sind sorgfältig zu Beginn der jeweiligen Kapitel referenziert worden, zum Zwecke einer besseren Lesbarkeit wurde allerdings darauf verzichtet, alle Auszüge in Anführungszeichen kenntlich zu machen.

1.2.1 Forschungsfrage

Bevor es im folgenden Kapitel mit den Grundlagen weitergeht, ist es sinnvoll, zunächst eine Definition der Forschungsfrage einzuführen:

> Lassen sich Methoden aus dem Bereich des Statistischen Model Checking (SMC) im Rahmen von geführter Simulation nutzbar machen, um die modellbasierte Entwicklung von sicherheitskritischen Fahrerassistenzsystemen, hinsichtlich ihrer angedachten Funktionsweise, in einem möglichst realistischen Anwendungsszenario zu unterstützen?

Die Vision hinter der Fragestellung ist es, einem Entwicklungsingenieur für sicherheitskritische Fahrerassistenzsysteme eine Entwicklungs- und Testmethodik zur Verfügung zu stellen. Diese soll es in einem modellbasierten Entwicklungsprozess ermöglichen, extrem seltene Risikosituationen zunächst sichtbar zu machen und anschließend ihre Auftretenswahrscheinlichkeit möglichst exakt abzuschätzen.

Für die Beantwortung der Forschungsfrage wurden vier relevante Teilfragen identifiziert, die im Verlauf der Arbeit nacheinander adressiert werden und in ihrer Zusammensetzung am Ende als Einheit zu betrachten sind.

1. Welche technischen Voraussetzungen müssen geschaffen werden, um Entwicklungswerkzeuge, Tools und Fahrsimulationssoftware, die unter anderem im Rahmen von modellbasierter Entwicklung zu Einsatz kommen, zu einer Kosimulation zusammenzuschließen?
2. Kann Statistisches Model Checking im Rahmen einer Kosimulation in einem möglichst realistischen Anwendungsszenario nutzbar gemacht werden, um seltene Ereignisse, wie sie auch im Kontext sicherheitskritischer Fahrerassistenzsysteme vorzufinden sind, simulativ zu erfassen?
3. Wie lassen sich die erreichten Ergebnisse auch im Falle sehr seltener Ereignisse hinreichend exakt quantitativ bewerten? Als hinreichend exakt gilt dabei eine Schätzgenauigkeit in der Größenordnung der domänentypischen quantitativen Sicherheitsziele.
4. Lässt sich die Methodik auch auf andere Domänen übertragen?

1.2.2 Beitrag zum Stand der Technik

In den originalen Publikationen, deren Inhalte Grundlage dieser Dissertation sind, hat der Autor zusammen mit den jeweiligen Koautoren die folgenden Beiträge zum Stand der Technik geleistet. Zunächst wurde ein Framework entwickelt und implementiert, mit dem es mög-

lich ist, im Rahmen von Rapid-Prototyping unterschiedliche Simulatoren zu einer Kosimulation zusammenzuschließen. Der IEEE 1516 Standard [IEE10] stellt diesbezüglich mit der High Level Architecture (HLA) ein wohl etabliertes Konzept zur Kopplung von verteilten Simulatoren bereit, eignet sich aufgrund seines Schnittstellenumfangs und seiner zahlreichen Funktionalitäten allerdings nur schlecht für Rapid-Prototyping. Das vorgeschlagene Framework, welches den Schnittstellenumfang stark reduziert und sowohl eine Open Source- als auch eine kommerzielle Runtime-Infrastructure (RTI) unterstützt, wurde mit einer Virtual-Human-In-The-Loop-Simulation validiert und hinsichtlich Effizienz begutachtet [Puc+12a]. Mit einem Kosimulationssetup, bestehend aus einem ausführbaren Menschmodell, einer Fahrsimulationssoftware und unterschiedlichen Monitoring-Tool, wurde die Fragestellung evaluiert, wie der Effekt von Fahrerassistenzsystemen analysiert und vorhergesagt werden kann [Frä+11; Frä+10]. Durch den technischen Zusammenschluss mehrerer Simulatoren zu einer Kosimulation entsteht allerdings ein verhältnismäßig großer Zustandsraum, der nicht in seiner Gesamtheit untersucht werden kann. Dem Bedarf, auch in einem solch komplexen Simulationsaufbau Analysen durchführen und statistische Aussagen zu speziellen Ereignissen treffen zu können, wurde Rechnung getragen, indem eine Methodik entwickelt wurde, welche es erlaubt, eine Kosimulation aus heterogenen domänentypischen Simulatoren in eine zuvor definierte Richtung zu führen. Für das Konzept einer „geführten Simulation“ wurde ein Algorithmus entwickelt, der es ermöglicht, extrem seltene Ereignisse innerhalb einer Kosimulation sichtbar zu machen [Puc+12b; Puc+13]. Diese Ausgangsbasis ermöglichte es im Folgenden, nicht nur qualitative Aussagen (wie wahrscheinlich ist das Auftreten seltener Ereignisse), sondern auch mit statistischen Methoden quantitative Bewertungen (mit welcher Konfidenz sind die erzielten Ergebnisse zu bewerten) abzuleiten [PFG18].

1.3 Aufbau der Arbeit

Dieses erste Kapitel widmet sich der Motivation und Einführung in das Thema der Arbeit. Im zweiten Kapitel werden die im weiteren Verlauf benötigten Grundlagen beschrieben und Probleme aufgezeigt, die im Fortgang der Arbeit adressiert werden. Darin eingeschlossen ist die Darstellung etablierter methodischer Ansätze zur Simulationsoptimierung sowie ein Vergleich mit bestehenden Arbeiten in einem verwandten bzw. vergleichbaren Kontext. Den Kern der Arbeit bilden die Kapitel drei und vier. Sie beschreiben zuerst die Entwicklung eines Frameworks, in dem heterogene Systemklassen zu einer Kosimulation zusammengeschlossen werden können. Anschließend wird ein Algorithmus für eine geführte Simulation beschrieben, der mit einer Instanz des Frameworks – bestehend aus Fahrsimulationssoftware, kognitivem Fahrermodell und Fahrerassistenzsystem – in einem sicherheitskritischen Fahrszenario seltene Ereignisse mit einer deutlich reduzierten Anzahl an Simulationsläufen aufzeigen kann, als es im Vergleich mit naiver Monte-Carlo-Simulation möglich ist. Eine nachfolgende statistische Analyse der Simulationsergebnisse ermöglicht eine quantitative

Bewertung, welche in Verbindung beispielsweise mit der Zulassung eines Fahrerassistenzsystems notwendig ist. Im Abschluss des vierten Kapitels werden die Übertragbarkeit der entwickelten Methodik auf andere Domänen diskutiert und Grenzen des Ansatzes aufgezeigt. Kapitel fünf fasst die Arbeit noch einmal zusammen und gibt einen Ausblick auf mögliche zukünftige Arbeiten.

Kapitel 2

Grundlagen und Problembeschreibung

In diesem Kapitel werden Methoden und Techniken eingeführt, welche für die Beantwortung der Forschungsfrage relevant sind oder in ihrem Zusammenhang stehen. Zu Beginn wird beschrieben, warum während eines Entwicklungsprozesses Modelle eingesetzt werden und es sinnvoll ist, eine Simulation der Eigenschaften nicht autark, sondern in Kombination mit anderen Modellen in einer Kosimulation durchzuführen. Daran anschließend wird die Problematik aufgezeigt, welche bei dem Vorhaben entsteht, innerhalb einer Kosimulation mit einem Szenario aus der Praxis, seltene Ereignisse simulativ zu erfassen. Dazu wird zunächst erklärt, was unter dem Begriff „selten“ zu verstehen ist und wie Statistik eine Abschätzung der zu einer Erfassung notwendigen Simulationszeit ermöglicht. Mit der Erkenntnis, dass naive Monte-Carlo-Simulation für seltene Ereignisse zu zeitaufwendig ist, wird eine Auswahl etablierter Optimierungsverfahren zur Reduktion der notwendigen Simulationszeit diskutiert, welche zudem eine anschließende statistische Bewertung der erreichten Ergebnisse ermöglichen. Zusammenfassend werden die im Kapitel identifizierten Anforderungen resümiert und eine Abgrenzung zu verwandten Arbeiten dargelegt.

2.1 Modellbasierte Entwicklung und Simulation

Modellbasierte Entwicklung ist in den vergangenen Jahren zum Quasistandard bei der Entwicklung neuer Systeme geworden. Modelle sind vereinfachte Abbilder der Wirklichkeit (z. B. kleinerer Maßstab) und haben den großen Vorteil, je nach Abstraktionsgrad schon zu einem sehr frühen Zeitpunkt des Entwicklungsprozesses zur Verfügung zu stehen. Sie erlauben ein visuelles Feedback, können auf unbedachte Probleme aufmerksam machen und einen guten Überblick über das spätere Gesamtsystem ermöglichen. Schon vor Jahrtausenden entwarfen Wissenschaftler wie Aristoteles und Ptolemäus für große komplexe Systeme zunächst Modelle [Cel61, S. 83], um zu analysieren, evaluieren oder visualisieren, wie

Partielle Ausschnitte dieses Kapitels wurden in [Puc+12a; Puc+12b; PFG18] veröffentlicht.

sich erdachte Konzepte verhalten und erklären ließen. Spätestens mit dem Einzug der Digitalisierung in den Massenmarkt wurden modellbasierte Verfahren auch im Rahmen von CAD-Programmen in fast allen Brachen eingesetzt, da sie einen kostengünstigen und zeiteffizienten Entwurf von Prototypen erlauben. Bei der Entwicklung von Fahrerassistenzsystemen ist ein modellbasiertes Vorgehen inzwischen ebenso gängige Praxis [Ehm05; KEB07; EB07]. Technische Systeme werden dabei zunächst in modellbasierten Entwicklungswerkzeugen wie beispielsweise MATLAB / Simulink, SCADE, AutoCAD, etc. hinsichtlich ihres physischen Verhaltens nachmodelliert und anschließend im Rahmen von Simulationen, automatisierten Tests bzw. Modelcheckern auf korrekte Funktionalität hin überprüft. Bei den am Computer ausführbaren Modellen werden dabei die in der Realität zu erwartenden Eingaben an die Eingangsschnittstellen der Modelle gelegt und nach Ausführung des Modells lassen sich die Ausgaben an den entsprechenden Ausgängen auswerten. Je nach Präzisionsgrad der verwendeten Modelle können zusätzlich alle Funktionsketten, welche an der Verarbeitung der Eingangssignale bis zur Ausgabe beteiligt sind, detailliert analysiert werden. Die sogenannte „Simulation“ [Meh94] eines Modells wird nach dieser Vorgehensweise etwas aufwändiger, wenn auch die Eingaben von anderen Modellen bereitgestellt werden sollen und analog die produzierten Ausgaben an nachgelagerte Modelle weitergereicht oder rückgekoppelt werden sollen. Solange die Grenzen eines Modellierungswerkzeugs nicht verlassen werden und die einzelnen Teilmodelle über hinreichend ähnliche Simulationszeitmodelle [Puc07] verfügen, wird lediglich der Handhabungsaufwand für den Entwickler entsprechend größer. Werden für die Bereitstellung von Eingaben bzw. das Weiterreichen von Ausgaben andere Modellklassen benötigt, die sich z. B. aufgrund unterschiedlicher interner Zeitrepräsentationen nicht innerhalb eines Modellierungswerkzeugs abbilden lassen, steigert sich der Komplexitätsgrad nicht nur hinsichtlich des Modellierungsaufwands, sondern auch der für eine spätere Kosimulation der einzelnen Teilmodelle notwendige Entwicklungsaufwand hinsichtlich geeigneter Schnittstellen für den Austausch von Daten oder Nachrichten.

2.2 Kosimulation heterogener Systeme

Unter einer Kosimulation wird im Kontext dieser Arbeit die gleichzeitige Simulation unterschiedlicher Modelle bezeichnet, die während ihrer Ausführung an den modelleigenen Schnittstellen Daten bzw. Nachrichten austauschen und hinsichtlich ihres internen Simulationszeitmodells rückgekoppelt sind. Eine Schwierigkeit beim Aufbau einer Kosimulation besteht nicht nur in der Definition von passenden Datentypen zum Austausch von Informationen, sondern auch in der Definition eines geeigneten, möglicherweise abstrakten Zeitmodells, so dass sich die unterschiedlichen Simulatoren bei ihrer Ausführung nicht gegenseitig beeinträchtigen.

Der unvermeidliche Extra-Aufwand bei der Entwicklung einer Kosimulation im Gegensatz zu einer monolithischen Simulation relativiert sich schnell, wenn der Blick auf die Vorteile gerichtet wird. Dazu ist es notwendig, noch einmal einen Schritt zurückzugehen und die Entwicklung eines ausführbaren Modells zu betrachten.

Für den Kontext dieser Arbeit wird zunächst definiert, dass ein Assistenzsystementwickler ein ausführbares Modell seines Systems, welches für sicherheitskritische Situationen gedacht ist (z. B. ein Notbremsassistent), im Rahmen einer Kosimulation analysieren und evaluieren möchte. Der Entwickler nutzt dazu das Entwicklungs- oder Modellierungswerkzeug, in dem er durch jahrelange Erfahrung die meiste Expertise vorzuweisen hat. Im Kontext der Automobilindustrie werden dazu beispielsweise Tools wie MATLAB / Simulink benutzt, welche eine grafische Oberfläche für den Entwickler bieten, um physikalische, mechanische oder elektronische Prozesse in einem ausführbaren Modell nachzubilden. Soll die korrekte Funktionalität des Systems nach der Modellierung aber über den reinen funktionalen Ablauf hinaus an einem realistischen Fahrzeugmodell mit entsprechender Dynamik getestet werden, oder wird ein System entwickelt, welches die Interaktion mit einem Fahrzeugführer erfordert, ist eine Rahmenbedingung gegeben, die sich innerhalb eines Entwicklungswerkzeugs meinst nicht oder nur mit sehr begrenzten Möglichkeiten umsetzen lässt.

Für die computergestützte Nachbildung einer Fahrer-Fahrzeug-Interaktion gibt es wiederum ausführbare Fahrzeug- und Fahrermodelle am Markt, zumeist sind diese aber in einem anderen, nicht unbedingt direkt kompatiblen Entwicklungswerkzeug modelliert worden. Das ist einerseits darin begründet, dass beispielsweise im Rahmen von simulierbaren Fahrermodellen andere Systemklassen zum Einsatz kommen und andererseits das Domänenwissen des jeweiligen Entwicklers eine völlig andere Herangehensweise notwendig macht. Menschmodelle, die kognitive Prozesse nachbilden, können zum Beispiel in Werkzeugen wie ACT-R [TA09] oder CASCaS [WBL13; WLB13] entwickelt werden. Der Entwickler eines Fahrermodells versucht dabei Handlungsabläufe oder Gedächtnisabläufe nachzubilden, die im Rahmen von Simulatorstudien mit Probanden zunächst aufgezeichnet und im Anschluss analysiert und statistisch evaluiert werden. Die Nachbildung erfolgt dann in regelbasierten oder prozeduralen Beschreibungssprachen aber auch funktionale Abbildungen in der Programmiersprache LISP sind in der Praxis zu finden [ACT19]. Der Fokus bei der Modellbildung liegt in diesem Bereich zunächst darin, im Mittel eine passende Nachbildung der in der Studie erhobenen Daten zu erreichen. Nicht selten sind dabei Experten aus der Domäne der Psychologie beteiligt, welche nicht zwingend einen direkten Bezug zur (logisch orientierten) Informatik mit sich bringen, sondern zumeist Hintergrundwissen aus der empirischen Verhaltensforschung.

Im Hinblick auf eine gemeinsame Simulation von Modellen verschiedener Systemklassen – hier modellbasiertes Entwicklungswerkzeug und prozedurale Beschreibungssprache – müssen nicht nur unterschiedliche Domänen vereint werden, sondern auch passende Schnittstellen für eine Interaktionsmöglicheit existieren, die Zugriff auf notwendige, gegebenenfalls in-

terne, Komponenten ermöglichen. Letzteres bieten Hersteller von Spezialsoftware aber nicht unbedingt vollständig oder im gewünschten Format an, da das technische Know-How vor Konkurrenz geschützt werden soll, oder weil der ursprünglich intendierte Anwendungskontext ein ganz anderer ist. So lassen sich zum Beispiel Realzeitfahrsimulatoren nicht an beliebigen Stellen in einem Szenario instanziieren, weil die integrierten Fahrdynamikmodelle dazu nicht in der Lage sind [WIV19; STS11].

Festgehalten werden soll an dieser Stelle, dass jedes Entwicklungs- oder Simulationswerkzeug zumeist einen speziellen Hintergrund in einer Domäne hat, um spezifische Aspekte zu modellieren und zugehörige Fragestellungen zu beantworten. Die Unterschiede reichen dabei von inkompatiblen Abstraktionsgraden über möglicherweise divergente Simulationszeitmodelle bis hin zu unterschiedlichen Programmiersprachen, in denen das ausführbare Modell verfügbar ist.

Deshalb gibt es Methoden und Standards, welche versuchen, die genannten Probleme angemessen zu lösen. Die High Level Architektur (HLA), verfügbar als Standard IEEE-1516 [IEE10], definiert ein Framework, welches sich eignet, eine Kosimulation heterogener Systeme aufzusetzen und zu simulieren, ohne dabei die Eigenheiten einzelner an einer Simulation beteiligter Komponenten zu beschränken. Die Entstehung der HLA geht auf eine Entwicklung des American Department of Defense im Jahr 1996 zurück. Sie sollte der Wiederverwendbarkeit bestehender Simulatoren Rechnung tragen und eine möglichst große Flexibilität hinsichtlich der nutzbaren Simulations- und Datenmodelle gewährleisten. Der Standard definiert zu diesem Zweck Rahmenbedingungen und Regeln, an die sich ein Simulator zwecks Kompatibilität halten muss, liefert selbst aber keine Implementierung einer notwendigen zentralen Steuerungskomponente, der sogenannten Runtime Infrastuktur (RTI), mit. Lediglich die Schnittstellen, welche eine RTI zur Verfügung stellen muss, werden spezifiziert. Die HLA wird im späteren Verlauf der Arbeit noch einmal in einem eigenen Abschnitt im Detail beschrieben.

Eine weitere Möglichkeit, verschiedene ausführbare Modelle in eine Kosimulation zu integrieren, ist das im Jahr 2010 in der ersten Version veröffentlichte Functional Mock-up Interface (FMI) für den Modellaustausch bzw. für Kosimulation. Initiiert wurde die Entwicklung von der Daimler AG mit dem Ziel, den Modellaustausch zwischen Automobilherstellern (OEMs) und den Zulieferern (Supplier) zu verbessern. Inzwischen beteiligen sich ca. 16 Firmen und Forschungsunternehmen an der aktiven Weiterentwicklung, eine Version 2.0 wurde im Jahr 2014 veröffentlicht und integriert die beiden Ansätze der ersten Version [Mod14].

FMI hat wie HLA das grundsätzliche Ziel, Modelle und Werkzeuge während der Entwicklung wiederverwendbar zu machen. Dennoch haben beide Standards völlig unterschiedliche Ansätze. Die High Level Architecture definiert Services wie Ressourcen-, Zeit-, Daten- und Besitzmanagement, während FMI auf einem niedrigeren Detailgrad arbeitet. So gibt es

bei FMI ausschließlich Methoden für den Austausch einfacher Datentypen und den lokalen Simulationszeitfortschritt. An die Stelle einer HLA Runtime Infrastuktur tritt bei FMI eine Master Functional Mock-up Unit (FMU) und statt einer abstrakten Zeitverwaltung werden ausschließlich diskrete Kommunikations- und Synchronisationspunkte verwendet. Trotz grundsätzlicher Unterschiede stellt [Yil+14] eine Methode vor, HLA-kompatible Simulationsteilnehmer mittels FMI zu entwickeln.

Um auf das fortlaufende Eingangsbeispiel einer Kosimulation für einen Assistenzsystementwickler zurückzukommen, wurden zwei technische Möglichkeiten skizziert, um ein modelliertes Fahrerassistenzsystem, im Folgenden auch als „System unter Test“ bezeichnet, mit anderen Modellen gemeinsam simulativ zu evaluieren. Da zahlreiche Assistenzsysteme in heutigen Fahrzeugen mit einem Fahrzeugführer interagieren, sei es weil Eingaben zwingend erforderlich sind oder Informationen zwecks Transparenz kommuniziert werden sollen, und in 88% der Verkehrsunfälle in 2017 menschliches Fehlverhalten vom Fahrzeugführer zu den Unfallursachen gehörten [Sta17a, S. 11], soll in einer geeigneten Art und Weise die Interaktion mit einem Fahrzeugführer einbezogen werden. Eine Möglichkeit der simulativen Darstellung eines Fahrzeugs samt Umwelt bietet der Einbezug von Fahrsimulationssoftware. Letztere enthält neben einer Umweltsimulation (Straßen, Landschaft, ggf. Umgebungsverkehr und Lichtsignalanlagen) auch eine Fahrdynamik für das Fahrzeug (auch als Egofahrzeug bezeichnet), welches vom Autofahrer gesteuert werden muss. Im weiteren Verlauf der Arbeit bedeutet eine Kosimulation heterogener Systeme folglich eine gemeinsame Simulation eines Fahrerassistenzsystems mit einem Fahrermodell und einer Fahrsimulationssoftware. Da zu Beginn der Forschungsaktivitäten für die vorliegende Arbeit noch keine Standardisierung von FMI existierte, wurde die HLA genommen.

2.3 Simulation seltener Ereignisse

Nachdem die mögliche Kopplung unterschiedlicher Modelle zu einer Kosimulation beschrieben wurde, folgt im nächsten Schritt eine Betrachtung zur Laufzeit. Dazu ist zunächst ein Blick auf Zufallsentscheidungen notwendig, im Folgenden auch als probabilistische Elemente bezeichnet. Viele Assistenzsysteme basieren auf deterministischen endlichen Automaten, insbesondere, wenn sie mit Modellierungswerkzeugen wie MATLAB / Simulink / Stateflow, Statemate, SCADE, etc. entwickelt wurden. Das bedeutet, dass sie bei gleicher Belegung der Eingangsvariablen ein identisches Ergebnis am Ausgang erzeugen. Ist das Spektrum der möglichen Eingangsbelegungen allerdings so groß, dass nicht alle Kombinationen ausprobiert werden können, besteht ein alternativer Ansatz darin, die zu erwartenden Eingangsgrößen aus der Umwelt des Modells geeignet nachzuempfinden. Etablierte Methoden wie Äquivalenzklassenbildung oder systematische Test sollen an dieser Stelle aber nicht diskutiert werden. Der Fokus soll hier vielmehr auf die zufällige Auswahl von Variablenbe-

legungen gelegt werden. Zufällige Belegungen sind von einem Zufallsgenerator abhängig, welcher während der Ausführung eines Modells gemäß einer Verteilung (Gleichverteilung, Normalverteilung, etc.) eine Belegung bestimmt. Zufallsentscheidungen sind also zum einen wünschenswert und notwendig, um beispielsweise einen unendlichen Eingabebereich systematisch testen zu können, zum anderen bringen Zufallsentscheidungen aber Probleme mit sich. So steigt der Aufwand zur Reproduktion von Ergebnissen (auch der Zufallsgenerator muss identisch nachgestellt werden) und bei unpassender Wahl eines Zufallsgenerators können viele Eingaben ungewollt mehrfach bzw. mit einer höheren Wahrscheinlichkeit erzeugt werden, so wie bei zwei Würfeln die Augenzahl 7 durch diverse Kombinationen realisierbar ist. Um die Menge an Zufallsentscheidungen innerhalb eines Kosimulationsaufbaus bei der späteren Evaluation bewältigen zu können, werden die Modellklassen für die weitere Betrachtung wie folgt eingeschränkt: Ein Assistenzsystem ist als deterministischer endlicher Automat realisiert und auch bei einer Fahrsimulationssoftware, welche primär für die Durchführung von reproduzierbaren Nutzerstudien entwickelt ist [WIV19], wird von deterministischem Verhalten ausgegangen. Beide Modelle sind damit frei von zufallsbedingten Wahrscheinlichkeiten.

Im Gegensatz dazu enthält das Modell eines Autofahrers, welches vielfach aus empirischen Beobachtungen von Autofahrern in Nutzerstudien extrapoliert wurde, auf Modellebene diverse Zufallsentscheidungen. Ein Autofahrermodell wird bei der Entwicklung darauf ausgelegt, im Mittel das Verhalten zu reproduzieren, was zuvor beobachtet wurde. Im Umkehrschluss bedeutet diese Auslegung allerdings, dass nicht beobachtetes Verhalten nur ungenau nachempfunden ist und damit nur zu einer gewissen Wahrscheinlichkeit zutreffend ist. Bei einer tiefer gehenden Analyse der stochastischen (zufallsbedingten) Prozesse in einem Fahrermodell wie z. B. CASCaS [WBL13; WLB13] lässt sich zudem feststellen, dass einige Prozesse wie z. B. Gedächtnisprozesse oder Handlungsausführungen eine sehr kleine Wahrscheinlichkeit haben, wohingegen die übrigen im relativen Verhältnis sehr hohe Wahrscheinlichkeiten aufweisen. Bei einer (Ko)Simulation des Fahrermodells werden folglich Simulationsläufe, bei denen ausschließlich die Prozesse mit sehr geringen Wahrscheinlichkeit auftreten, äußerst selten sein.

Unter dem Begriff „selten“ nennt Beck in [BZ15] zum Beispiel eine Wahrscheinlichkeit kleiner als 10^{-3}. Aufgrund der Domäneneigenschaften wird für den Rahmen dieser Arbeit allerdings eine Größenordnung um die 10^{-6} (ein Millionstel) als seltenes Ereignis (Rare Event) festgelegt. Dieser Wert lässt sich konservativ aus Unfallzahlen des Statistischen Bundesamtes herleiten. Die Polizei registrierte auf deutschen Straßen im Jahre 2017 insgesamt 2.643.098 (ca. 2,6 Millionen) Verkehrsunfälle bei 784 Milliarden zurückgelegten Kilometern [Sta17a]. Das entspricht etwa einem Unfall je 300.000 km gefahrene Strecke oder einer Wahrscheinlichkeit von $3,3 \cdot 10^{-6}$ pro gefahrenem Kilometer, um in einen Unfall verwickelt zu werden. Bei einer analogen Rechnung, eingeschränkt auf den Bereich von Autobahnen, wo insgesamt mehr Kilometer pro Unfall zurückgelegt werden (ca. 31,4% der zurückgeleg-

ten Kilometer und ca. 26,8% der Verkehrsunfälle, wenn die innerörtlichen abgezogen werden) und Unfälle mit Personenschaden (6,9%), sinkt die Wahrscheinlichkeit auf $2,01 \cdot 10^{-7}$ pro gefahrenem Kilometer, in einen derartigen Unfall verwickelt zu werden. Die unter sozioökonomischen Aspekten nach wie vor viel zu hohe Anzahl an Verkehrsunfällen wandelt sich, beim Wechsel der Betrachtungsweise von der Gesamtheit hin zum individuellen Einzelfall, zu einem extrem seltenen Ereignis.

Was lässt sich daraus folgern, wenn derart seltene Ereignisse in einem Kosimulationsaufbau bestehend aus Fahrerassistenzsystem, Fahrsimulationssoftware und Fahrermodell simulativ analysiert werden sollen? Bei klassischer Monte-Carlo-Simulation, auch als *Sampling* bezeichnet, wird der Simulationsaufbau gestartet, das Endergebnis nach einem Simulationslauf notiert und anschließend der nächste Simulationslauf durchgeführt. Jeder Simulationslauf entspricht einem Zufallsexperiment, bei dem die probabilistischen Elemente in den verwendeten Modellen mit Hilfe eines Zufallsgenerators belegt werden. Dieses stochastische Verfahren eignet sich zum Beispiel dazu, Probleme, die sich nicht analytisch lösen lassen, mit Hilfe des Gesetzes der großen Zahlen [Hen13] aus der Wahrscheinlichkeitstheorie numerisch zu lösen. Die Simulation seltener Ereignisse ist mit der Variante hingegen nur dann sinnvoll, wenn keine Methode zur Simulationsoptimierung anwendbar ist oder die Simulationslaufzeit eine untergeordnete Rolle spielt. So würde beispielsweise die Simulation eines Unfalls mit Personenschaden auf einer Autobahn, bei dem die Funktionsweise eines neuartigen Assistenzsystems zum verbesserten Schutz des Fahrzeugführers analysiert werden soll, mit den im letzten Abschnitt genannten Daten und einer angenommenen Richtgeschwindigkeit von 130 km/h in der Größenordnung von 2.300 Stunden bzw. 96 Tagen liegen. Einige Methoden, die sich für eine Optimierung der Simulationslaufzeit eignen, werden in Kapitel 2.5 auf Seite 20 genauer betrachtet.

2.4 Statistisches Model Checking

Für die Überprüfung eines Modells eines Assistenzsystems auf seine korrekte Funktionalität wird in der Informatik der Begriff „Model Checking“ (Modellprüfung) verwendet. Gemeint ist damit die Verifikation, ob das Modell eines Systems seiner vor der Entwicklung gesetzten Spezifikation entspricht [ISO15]. Diese Überprüfung soll zum einen vollständig und zum anderen automatisch erfolgen. Am Ende des Überprüfungsprozesses kann ein Zertifikat mit dem entsprechenden Ergebnis ausstellt werden. In einem Fehlerfall besteht die Möglichkeit, den genauen Hergang zu reproduzieren. Endet der Prozess ohne Beanstandung, ist die korrekte Funktionalität des Systems gemäß seiner Spezifikation sichergestellt. Problematisch ist in diesem Verfahren der Aspekt der Vollständigkeit, wenn der zu analysierende Zustandsraum für eine automatische Überprüfung zu groß wird. Bei dem Versuch der vollständigen

Analyse eines Fahrerassistenzsystems in Kombination mit einem stochastischen Fahrermodell würde das Model Checking in simulativer Form mittels einer Monte-Carlo-Simulation aufgrund des unendlich großen Zustandsraumes des Fahrermodells, verbunden mit teils sehr kleinen Wahrscheinlichkeiten z. B. für die Nachbildung von Gedächtnisprozesse, nicht in akzeptabler Zeit terminieren.

Um dennoch nach endlicher Zeit einer Überprüfung ein Zertifikat ausstellen zu können, sind Methoden der Statistik hilfreich. „Statistik [...] als Wissenschaftliche Disziplin ist [...] eine der Möglichkeiten, eine systematische Verbindung zwischen Erfahrung (Empirie) und Theorie herzustellen“ [Rin08, S. 1]. Sie beschreibt, wie quantitative Informationen (Daten) analysiert und interpretiert werden können. Bei der Simulation eines stochastischen Modells in einem unendlichen Zustandsraum erweitert jeder Simulationslauf die Empirie, und der Einsatz von Statistik erlaubt z. B. die Beantwortung der Fragestellung, wann genügend Simulationsläufe durchgeführt wurden, um eine Theorie anzunehmen bzw. zurückzuweisen. Die Kombination von Model Checking und Statistik (Statistisches Model Checking) als Technik bedeutet daher simulatives Model Checking (Monte-Carlo-Simulation) kombiniert mit statistischer Analyse, um stochastische Systeme zu verifizieren.

Die Idee des Statistischen Model Checkings geht auf Håkan Younes und Reid Simmons zurück [YS02; You+06; YS06]. In einem einfachen stochastischen System lässt sich aus Stichproben wie folgt generalisieren:
Gegeben sei ein stochastischer Prozess P sowie eine Zufallsvariable $x \in [0,1]$, welche in P zufällig belegt werden kann. Der tatsächliche Erwartungswert E_x der Variablen x im Prozess P lässt sich dann folgenderweise abschätzen: Zuerst wird eine Menge von $n \in N$ zufälligen Stichproben (beispielsweise Simulationsläufe) generiert, wobei für die Variable x jeweils ein Wert x_i, $i \in \{1,\ldots,n\}$ entsteht. Anschließend wird der geschätzte Erwartungswert $\tilde{E}_x$ über den Mittelwert aller Belegungen von x_i berechnet:

$$\tilde{E}_x = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.1}$$

Gemäß dem Gesetz der großen Zahlen [Hen13] ist bei einer ausreichend großen Stichprobe n der abgeschätzte Erwartungswert $\tilde{E}_x$ nicht weit vom tatsächlichen Erwartungswert E_x entfernt, denn für eine erhebliche Über- bzw. Unterabschätzung des tatsächlichen Erwartungswerts E_x müssten die Belegungen x_i einen sich aufsummierenden einseitigen Fehler aufweisen, was auffällig wäre.

Eine Abschätzung, wie groß der Unterschied zwischen dem tatsächlichen- und dem abgeschätzten Erwartungswert ist, kann über Hoeffdings Ungleichung [Hoe63] gegeben werden.

Sie beschreibt die Wahrscheinlichkeit, dass die Summe von unabhängigen, zufälligen Variablen mehr als eine Konstante von ihrem Mittelwert abweicht (vergleiche Gleichung 2.2).

$$\forall t > 0 : \; Pr(\tilde{E}_x - E_x \geq t) \leq e^{-2nt^2}, \qquad Pr(\tilde{E}_x - E_x \leq -t) \leq e^{-2nt^2} \tag{2.2}$$

Die Abschätzung ist symmetrisch, so dass sich eine Abschätzung sowohl gegen eine Konstante t als auch gegen ihr Inverses $-t$ ermöglichen lässt. Um Hoeffdings Ungleichung auf obiges Beispiel anwenden zu können, seien an dieser Stelle zum einen noch die Unabhängigkeit der einzelnen Eigenschaften des Prozesses P gefordert (wie bei Wahrscheinlichkeitsverteilungen, einem Zustandsraum, etc.), sowie die stochastische Unabhängigkeit der einzelnen Samples im Sinne von richtiger Zufälligkeit.

Auf die Praxis bezogen sieht eine Abschätzung nach Hoeffding dann wie folgt aus: Gegeben sei eine Eigenschaft, deren Erwartungswert kleiner oder gleich einem Schwellwert Teta ist.

$$E_x \leq \theta, \text{ mit } \theta \in]0,1[\tag{2.3}$$

Für eine beliebige Eigenschaft sei ferner eine Variable x gegeben, die den Wert 1 genau dann annimmt, wenn ein Simulationslauf einen kritischen Zustand erreicht, andernfalls den Wert 0 (siehe Gleichung 2.4).

$$x = \begin{cases} 1 & \text{ein Simulationslauf erreicht einen kritischen Zustand,} \\ 0 & \text{sonst.} \end{cases} \tag{2.4}$$

$E_x \leq \theta$ bedeutet umgangssprachlich also, dass das Risiko eines Fehlverhaltens (Erwartungswert) höchstens Teta ist. Diese Aussage soll nun mit rein zufälligen Simulationsläufen (sampling) in drei Schritten verifiziert werden.

1. Es wird ein kleiner „don't care"-Bereich t hinzugefügt, so das jedes Ergebnis akzeptiert wird, wenn es von Teta um die Konstante t nach oben bzw. unten abweicht.

$$E_x \in [\theta - t, \theta + t] \tag{2.5}$$

2. Anschließend werden n rein zufällige Simulationsläufe generiert und der Erwartungswert berechnet.

$$\tilde{E}_x \leq \theta \;\; \Leftrightarrow \;\; \tilde{E}_x - \theta \leq 0 \;\; \Leftrightarrow \;\; \varepsilon \leq 0 \quad \text{mit } \varepsilon \stackrel{\text{def}}{=} \tilde{E}_x - \theta \tag{2.6}$$

3. Zuletzt kann überprüft werden, ob $\varepsilon \leq 0$.

Die Ungleichung von Hoeffding beschreibt, wie das Ergebnis zu interpretieren ist:

- Wenn $\varepsilon \leq 0$ lässt sich sagen, dass sich die Eigenschaft erfüllt hat.

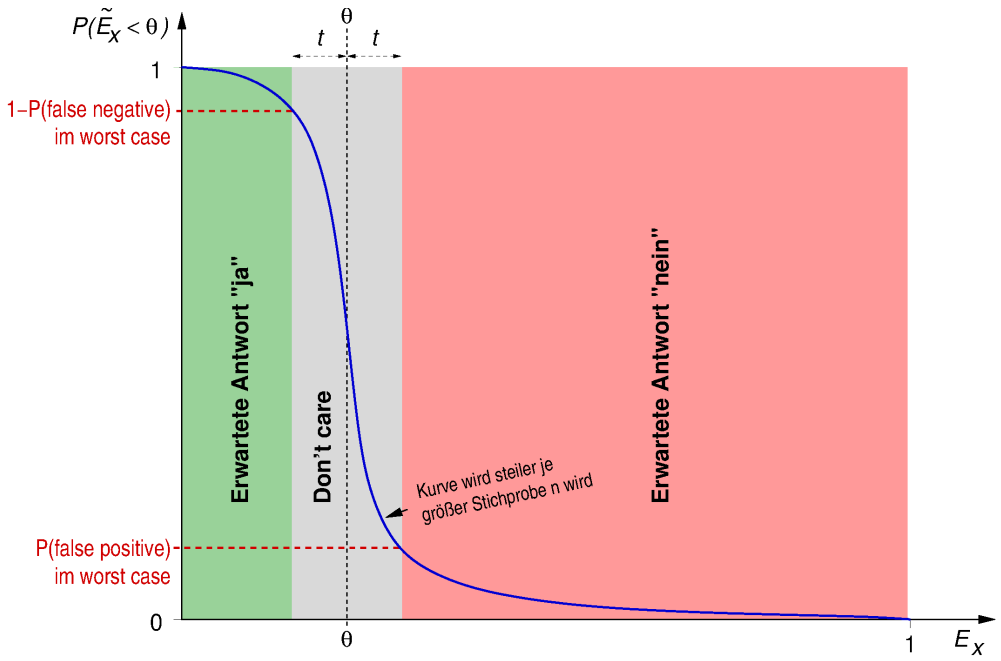

Abbildung 2.1: Grafische Interpretation von Hoeffdings Ungleichung. (Original: Martin Fränzle)

- Wenn $\varepsilon > 0$ lässt sich sagen, dass die Eigenschaft widerlegt ist.
- Jedes Ergebnis ist mit einer Konfidenz von $1 - e^{-2nt^2}$ zutreffend.
- Die Irrtumswahrscheinlichkeit (false positive / false negative) liegt maximal bei e^{-2nt^2}.

Eine grafische Interpretation kann der Abbildung 2.1 entnommen werden. Auf der X-Achse ist der tatsächliche Erwartungswert E_x aufgetragen, dessen realer Wert mittels Simulation abgeschätzt werden soll. Um den Schwellwert θ ist mit der Konstanten t ein kleiner Unsicherheitsbereich (don't care-Bereich) eingezeichnet, in welchem jedes Ergebnis der Simulation akzeptiert wird. Bei einem Wert oberhalb von θ, welcher außerhalb des Unsicherheitsbereichs liegt, wird das Ergebnis zurückgewiesen, analog wird ein Wert unterhalb akzeptiert. Auf der Y-Achse ist die Wahrscheinlichkeit aufgetragen, dass der geschätzte Erwartungswert kleiner des Schwellwerts θ ist. Je weiter der reale Erwartungswert E_x gegen 1 läuft, desto unwahrscheinlicher ist es, dass eine simulative Schätzung einen Wert kleiner θ liefert. Auf der anderen Seite steigt die Wahrscheinlichkeit, dass die Abschätzung einen Wert kleiner θ liefert, je weiter der reale Erwartungswert gegen 0 konvergiert. Besonders gekennzeichnet sind die Wahrscheinlichkeiten, dass ein falsches Ergebnis akzeptiert wird (false positive) bzw. ein korrektes Ergebnis zurückgewiesen wird (false negative). Die Steigung der Kurve, insbesondere im Bereich des Intervalls $\theta \pm t$, wird mit zunehmender Stichprobengröße größer, so dass die Wahrscheinlichkeit, ein falsches Zertifikat auszustellen (false positve / false

negative) sinkt. Allgemein lässt sich festhalten, dass die Wahrscheinlichkeit, ein falsches Zertifikat auszustellen gegen Null sinkt, wenn die Stichprobengröße gegen unendlich steigt oder t gegen das Maximum von θ und $1-\theta$ konvergiert (siehe Gleichung 2.7).

$$P(\text{falsches Zertifikat}) \to 0 \Leftrightarrow n \to \infty \text{ oder } t \to \max(\theta, 1-\theta) \tag{2.7}$$

Die notwendige Strichprobengröße n hängt direkt von der erwünschten Konfidenz ab, mit welcher der abgeschätzte Erwartungswert abgesichert werden soll. Mit einem Unsicherheitswert $t > 0$ als don't care-Bereich und einer Konfidenz $c < 1$ kann erneut Hoeffdings Ungleichung benutzt werden, wie folgendes Beispiel zeigt.

Nach Gleichung 2.2 lässt sich eine Konfidenz von 99% mit $1 - e^{-2nt^2}$ abschätzen. Bei einem Erwartungswert θ für ein seltenes Ereignis von 10^{-6} (vergleiche Abschnitt 2.3) sei als Unsicherheitsbereich t exemplarisch eine Größenordnung kleiner $(t = 10^{-7})$ gewählt. Damit ergibt sich folgende Rechnung für die Stichprobengröße n.

$$\begin{aligned}
& 99\% = 1 - e^{-2nt^2} \\
\Leftrightarrow \quad & 0{,}99 = 1 - e^{-2n\cdot 10^{-7\cdot 2}} \\
\Leftrightarrow \quad & 0{,}01 = e^{-2n\cdot 10^{-14}} \\
\Leftrightarrow \quad & \ln(0{,}01) = -2n \cdot 10^{-14} \\
\Leftrightarrow \quad & -1/2 \cdot 10^{14} \cdot \ln(0{,}01) = n \\
\Leftrightarrow \quad & \sim 2{,}3 \cdot 10^{14} = n
\end{aligned} \tag{2.8}$$

Für eine Konfidenz von 99% eines seltenen Ereignisses in der Größenordnung von 10^{-6} bedarf es also einer Stichprobengröße von ungefähr 230 Billionen. Da jede Stichprobe stellvertretend für einen Simulationslauf in dem zuvor beschriebenen Kosimulationssetup steht, lässt sich mit Kenntnis der Stichprobengröße die dazu notwendige Simulationsdauer abschätzen. Unter der Annahme, dass für jeden Simulationslauf lediglich 5 Sekunden Simulationszeit zu veranschlagen wären, (es sei an dieser Stelle noch einmal in Erinnerung gerufen, dass ein Realzeitsimulator Teil des Simulationssettings ist) summiert sich die Gesamtlaufzeit bereits auf 36 Millionen Jahre. Als ein Werkzeug zur Überprüfung eines sich in der Entwicklung befindlichen Fahrerassistenzsystems wäre der Aufbau damit nicht geeignet. Selbst unter hypothetischer Betrachtung einer Parallelisierung auf 10 Millionen Rechenkerne,[1] wäre eine Simulationszeit von 36 Jahren keine praxistaugliche Größenordnung.

Ein anderer Aspekt, der im Rahmen dieser Rechnung bisher unbeachtet geblieben ist, betrifft die angenommene Konfidenz von 99% im Zusammenhang mit seltenen Ereignissen. Bei der

[1] Der chinesische Großrechner Sunway TaihuLight hat aktuell 10.649.600 Kerne.

Wahrscheinlichkeitsschätzung per Simulation wird zunächst als Erwartungswert das Stichprobenmittel einschließlich seiner Varianz berechnet. Im Anschluss lässt sich das Konfidenzintervall bestimmen, wobei je nach Größe des zu schätzenden Werts θ mal die absolute halbe Breite und mal die relative halbe Breite des Konfidenzintervalls das geeignetere Gütekriterium für die Schätzung ist. Für eine genauere Differenzierung auch hinsichtlich unterschiedlicher Stichprobengrößen n sei an dieser Stelle auf die Dissertation von Sandmann verwiesen [San04]. Gängige Konfidenzniveaus liegen bei 90%, 95% bzw. 99%, wobei zu einer besseren Bewertung der Güte des Schätzers bei sehr kleinen Schätzwerten zusätzlich der relative Fehler (Variationskoeffizient) hinzugenommen wird. So ist ein Konfidenzintervall mit 10^{-3} im Allgemeinen zwar recht klein, für Schätzwerte im Bereich 10^{-6} oder 10^{-9} sind dort aber Werte enthalten, die um Größenordnungen abweichen. Für eine tabellarische Übersicht der benötigten Stichprobengrößen bei den verschiedenen Konfidenzniveaus einschließlich maximalem relativen Fehler sei ebenfalls auf die Arbeit von Sandmann verwiesen.

Offen bleibt an dieser Stelle die Frage, ein wie großer Restfehler bei sehr kleinen Schätzgrößen tolerierbar ist, gerade wenn es um den Nachweis von Sicherheitseigenschaften geht. Das Restrisiko einer Katastrophe von 10^{-9} erscheint auf den ersten Blick vernachlässigbar zu sein, bei einer Irrtumswahrscheinlichkeit von 1% (bei 99% Konfidenz) stellt sich der Sachverhalt jedoch anders dar.

Als Fazit lässt sich festhalten, dass für die Simulation seltener Ereignisse mit einem eingangs beschriebenen komplexen Systemaufbau, reine Monte-Carlo-Simulation ungeeignet ist. Es bedarf anderer Verfahren, um die Simulationszeit deutlich zu verkürzen und effizienter zu gestalten. Bei der Auswahl statistischer Analysemethoden muss insbesondere bei sehr kleinen Wahrscheinlichkeiten im Zusammenhang mit seltenen Ereignissen darauf geachtet werden, dass allgemeine Qualitätsaussagen nicht $1:1$ übertragbar sind und dass sich die Methode auf Modelle stützt, welche nur eine – möglicherweise fehlerbehaftete – Abstraktion ihrer physischen Umwelt repräsentieren.

2.5 Simulationsoptimierung

Der Einsatz von reiner Monte-Carlo-Simulation im Rahmen von Statistischem Model Checking kann, wie in Abschnitt Abschnitt 2.4 beschrieben, gerade bei seltenen Ereignissen Wochen bzw. mehrere Jahre dauern. Um einem Assistenzsystementwickler eine Methode für die alltägliche Entwicklungsarbeit zur Verfügung zu stellen, bedarf es daher nicht nur geeigneter, möglichst effizienter statistischer Methoden zum Entwurf und zur Auswertung von Simulationen, sondern auch Verfahren zu deren beschleunigter Ausführung. Dabei ist unter beschleunigter Simulation nicht das erhöhte Verhältnis von Simulationszeit zur Realzeit gemeint, sondern dass der Zeitaufwand zur Bestimmung einer statistischen Schätzung auf eine

gewünschte Genauigkeit (z. B. vorgegebenes Konfidenzintervall) reduziert wird. Da Parallelisierung (mehr Simulationen in gleicher Zeit), wie in Abschnitt 2.4 beispielhaft beschrieben, keine hinreichende Verbesserung mit sich bringt, bleibt nur die Möglichkeit, mit weniger Simulationen für verlässliche statistische Aussagen auszukommen. Simulationsbeschleunigung ist dann oftmals gleichbedeutend mit Varianzreduktion der verwendeten Schätzer, welche in der Simulationsliteratur zwar ausführlich beschrieben werden, allerdings meist in einem speziellen Kontext, der sich nicht direkt auf seltene Ereignisse übertragen lässt (vgl. Literaturverweise in [San04, S. 5]).

Einige Ansätze versuchen Techniken aus dem Bereich Counterexample-Guided Abstraction Refinement (CEGAR) im Rahmen von Simulationen nutzbar zu machen (siehe Unterabschnitt 2.5.1). Zwei etablierte Varianzreduktionsverfahren werden mit Importance Sampling in Unterabschnitt 2.5.2 und Importance Splitting in Abschnitt Unterabschnitt 2.5.3 genauer vorgestellt. Während das erste Verfahren versucht, den Zustandsraum mittels Abstraktion zu verkleinern und damit die zur vollständigen Simulation notwendige Zeit zu reduzieren, haben die beiden letzteren Verfahren das gemeinsame Ziel, die seltenen Ereignisse öfter hervorzurufen. Unterschiedlich sind jedoch einerseits die Voraussetzungen der Anwendbarkeit (z. B. Anforderungen an die verwendeten Simulatoren) und andererseits die Art und Weise, wie seltene Ereignisse öfter sichtbar gemacht werden sollen. So wird bei Importance Sampling das Wahrscheinlichkeitsmaß der Zufallsentscheidungen verändert, während bei Importance Splitting an festgelegten Punkten (den sogenannten Splitting Points) „aussichtslos erscheinende“ Simulationsläufe terminiert werden und im Gegenzug „vielversprechende“ Simulationsläufe aufgespalten auf mehrere Läufe fortgesetzt werden. Ausschlaggebend für den Erfolg beider Verfahren ist dabei die geschickte Art der Veränderung des Wahrscheinlichkeitsmaßes bzw. die Aufteilung der Simulationsläufe, welche zwar oftmals Domänen- / Expertenwissen erfordert, dafür aber im Idealfall Simulationsbeschleunigungen um mehrere Größenordnungen ermöglicht. Zitat nach Dr. W. Sandmann [San04]: „*Insbesondere beim Importance Sampling ist prinzipiell eine unbeschränkte Varianzreduktion möglich.*“

2.5.1 Counterexample-Guided Abstraction Refinement

Counterexample-Guided Abstraction Refinement (CEGAR) ist ein Ansatz, um dem Problem der Zustandsraumexplosion beim Model Checking entgegenzuwirken. In einem iterativen Prozess wird dabei zunächst ein abstraktes Modell des zu verifizierenden Modells erstellt. Die mit der Abstraktion einhergehende Vereinfachung soll anschließend eine einfachere Verifikation ermöglichen. Wird bei der Überprüfung ein Fehlerfall gefunden, muss zunächst überprüft werden, ob dieser auch im ursprünglichen Modell enthalten ist oder durch die Abstraktion „eingeschleppt“ wurde. Im letzteren Fall wird von einem unechten Fehler (*spurious counterexample*) gesprochen. Beim Auftreten eines solchen Fehlerfalls wird die anfängliche

Abstraktion unter Berücksichtigung dieses unerwünschten Falls verfeinert und der Verifikationsprozess kann von vorne starten.

Das CEGAR Verfahren hat seinen Ursprung im Symbolischen Model Checking [Cla+00; Cla+03], das Prinzip wird aber auch im simulativen Kontext unter der Bezeichnung *Abstraction-Guided Simulation* angewendet. So stellt Shyam in [SB06] eine hybride Verifikationssoftware mit dem Namen GUIDO vor, um Schaltkreise zu verifizieren. Sie nutzt eine Distanzfunktion, welche mittels exakter Erreichbarkeitsanalyse auf einem abstrakten Design Kosten anhand der kürzesten Entfernung zum Ziel berechnet, um damit die Zufallssimulation schrittweise in die gewünschte Richtung zu führen. Ein Logiksimulator übernimmt dabei die Aufgabe der Designabstraktion. In [DH07] wird die Führungsstrategie von GUIDO (heuristische Suche) verbessert, damit die Simulation weniger oft in lokalen Optima oder Sackgassen (*dead ends*) stecken bleibt. Die Problematik von Sackgassen entspricht dabei dem Problem der unechten Fehler. So kann es zwar in einem abstrakten Schaltkreisdesign eine Lösung geben, welche aber bei einer konkreten Instanziierung nicht umsetzbar ist. In [ZLL10] wird das Verfahren der abstraktionsgeführten Simulation mit einem Markov Model [Gag17] kombiniert, um das Problem der Sackgassen zu vermeiden und Testfälle für extrem schwer erreichbare Zustände in Mikroprozessoren zu generieren.

Für die Übertragbarkeit des Ansatzes auf andere Problemklassen – wie im Kontext dieser Arbeit – lässt sich festhalten, dass konzeptbedingt für ein komplexes gegebenes Ausgangsmodell zunächst eine (möglichst) gute Abstraktion gewonnen werden muss. Dazu werden einfachere Systeme wie beispielsweise Kripke-Strukturen verwendet. Wie die Literaturreferenzen zeigen, ist abstraktionsgeführte Simulation im Kontext von Schaltkreis- und Hardwaredesignverifikation ein durchaus erfolgreicher Ansatz, im Zusammenhang mit Kosimulationen heterogener Systemklassen konnten vom Autor aber keine Referenzen gefunden werden.

2.5.2 Importance Sampling

Das Importance Sampling (IS) ist eine Möglichkeit zur Varianzreduktion von Monte-Carlo-Simulation (MCS) [RT09]. Vereinfacht formuliert versucht es, die wichtigen Stichproben zu bevorzugen, weshalb auch vom Verfahren der wesentlichen Stichproben gesprochen wird. Die Methode basiert darauf, die zugrundeliegenden Wahrscheinlichkeiten von Zufallsereignissen so zu verändern, dass die gewünschten (zu untersuchenden) Ereignisse öfter in Erscheinung treten. Die bei dem Prozess systematisch verfälschten Ergebnisse lassen sich im Nachhinein mit einem Korrekturfaktor, dem sogenannten Likelihood-Quotienten (*likelihood ratio* oder dem *importance factor*), zurückrechnen (vergleiche Gleichung 2.9). Notwendig ist dazu lediglich die Berechnung und Notation des entsprechenden Verstärkungs- bzw. Abschwächungsfaktors (Gewichtungsfaktors) während der Laufzeit einer Simulation.

Wird das Verfahren mit Lernmethoden kombiniert, so dass eine iterative Anpassung des Gewichtungsfaktors möglich ist, wird auch von Adaptive Importance Sampling (AIS) [Buc88] gesprochen. Es gibt allerdings keine allgemeine Form des IS, welche eine Varianzreduktion garantiert, sondern lediglich Strategien, die oft auf Expertenwissen basieren und für spezielle Modellklassen passend sind, da sie die entsprechenden Modellparameter in ihrer Wahrscheinlichkeit verändern (*bias correction*).

Formal bedeutet IS, dass die einzelnen Stichproben nicht gemäß ihrer natürlichen, sondern nach einer vorgeschlagenen Verteilung (*proposal distribution*) gezogen werden. Der Erwartungswert $\tilde{E}[g]$ einer Zufallsvariable g wird mit IS wie folgt geschätzt (vergleiche [Buc04]):

$$\tilde{E}[g] = \frac{1}{N} \sum_{i=1}^{N} g(X_i) \, \frac{p(X_i)}{q(X_i)} \quad , \tag{2.9}$$

wobei $p(x)/q(x)$ der Likelihood-Quotient ist, mit $p(x)$ als Originalwahrscheinlichkeit der Variablen x und $q(x)$ die Wahrscheinlichkeit mit der für die Variable vorgeschlagenen Verteilung. Wie anspruchsvoll das Finden einer guten Wahrscheinlichkeitsdichtefunktion (*probability density function*) ist, zeigen die Arbeiten von Gietelink et al. [GDV05; GDV06]. Die Autoren präsentieren unterschiedliche Varianten von AIS, um am Beispiel eines einfachen Abstandregeltempomaten ein Fahrerassistenzsystem zu validieren. Alle Varianten haben gemeinsam, dass sie anfangs eine Anzahl von n Stichproben (als Batch) ziehen, bevor sie überhaupt eine Anpassung ihrer vorgeschlagenen Verteilung durchführen können. Da ihre Indikatorfunktion (*indicator function*) zum Anzeigen, ob ein seltenes, im vorliegenden Falle auch kritisches Ereignis, erreicht wurde, nur ein binäres Ergebnis (erfolgreich / nicht erfolgreich) liefert, können sie im worst-case[2] auch nach einem Batchlauf keine Anpassung an ihrer Wahrscheinlichkeitsdichtefunktion durchführen. Ungeachtet dessen zeigen die Autoren, dass AIS die Effizienz ihrer Simulation ungefähr um den Faktor zehn verbessert hat.

Eine theoretische Basis zum Bestimmen einer optimalen vorgeschlagenen Verteilung bietet die Kreuzentropie-Methode [Rub99]. Sie nutzt ein Maß für den relativen Abstand bzw. die relative Abweichung zwischen zwei Wahrscheinlichkeitsverteilungen, welches von Kullback und Leibler [KL51] definiert wurde (bekannt als *Kullback-Leibler divergence*), um adaptiv eine optimale Wahrscheinlichkeitsdichtefunktion zu bestimmen. Vereinfacht dargestellt werden bei der Methode in zwei abwechselnden Schritten erst eine Menge zufälliger Stichproben nach einem zuvor spezifizierten Verfahren und gesetzten Parametern bestimmt (z. B. Simulationsläufe mit geratenen Parametern) und anschließend der Kullback-Leibler-Abstand minimiert. Im nächsten Durchlauf können dann Stichproben mit besser passenden Parametern gezogen werden. Bei der praktischen Umsetzung hängt die Wirksamkeit eines solchen Ansatzes allerdings sehr stark von gut gewählten Parametern der Wahrscheinlichkeitsdichtefunktion ab, sowie bei einer algorithmischen Umsetzung innerhalb einer Simulation von

[2] Im worst-case wird kein seltenes Ereignis gefunden.

Batchgrößen oder initial passend gewählten Parametern, um im Kontext seltener Ereignisse diese überhaupt sichtbar zu machen (siehe Indikatorfunktion im vorhergehenden Absatz) [PFG18].

Zusammenfassend lässt sich festhalten, dass Importance Sampling eine etablierte Methode zur Varianzreduktion bei Monte-Carlo-Simulation ist. Im Kontext der Simulation seltener Ereignisse kommt sie in unterschiedlichen Systemklassen zur Anwendung [ZBC12; Jeg+16], wobei sich die Benchmarks auf überschaubare Matlab / Simulink Beispiele oder Automatenmodelle beziehen. Auch in der aktuellen Publikation von Salins [SS17], welche IS im Kontext von linearen stochastischen partiellen Differentialgleichungen beschreibt, liegt der Fokus auf einer theoretischen beweisgetriebenen Herleitung mit einer abschließenden numerischen Simulation. Keiner der Ansätze verwendet einen Kosimulationsaufbau mit unterschiedlichen Domänensimulatoren. Eine generelle Schwierigkeit beim IS ist das Finden einer geeigneten Wahrscheinlichkeitsdichtefunktion, was ein gewisses Verständnis der zugrundeliegenden Modelle voraussetzt und damit implizit Expertenwissen. Durch die methodische Notwendigkeit der Modifikation der Wahrscheinlichkeiten in den simulierten Modellen existiert zusätzlich ein gewisses Problem bei wahrscheinlichkeitsbehafteten Blackbox-Modellen, welche ihre Zufallsentscheidungen nicht nach außen transparent machen. Die Integration von rein deterministischen Blackbox-Modellen in eine Kosimulation mit zufallsbedingten Entscheidungen stellt hingegen kein Problem dar.

2.5.3 Importance Splitting

Importance Splitting (ISp) [JLS13] ist neben IS eine weitere etablierte Methode zur Varianzreduktion von MCS. Bei diesem Verfahren entscheidet ein Auswahlmechanismus (z. B. eine Abstandsfunktion), welche Simulationsläufe wahrscheinlich zu einem seltenen Ereignis führen. Die Grundidee besteht darin, wie in Abbildung 2.2 dargestellt, die absoluten Simulationspfade vom Start bis zu einem seltenen Ereignis in kürzere Teilpfade zu zerlegen. Dabei werden aussichtsreiche Teilpfade, die mutmaßlich zu einem seltenen Ereignis führen, begünstigt, indem sie vervielfältigt werden, wohingegen aussichtslose Teilpfade nicht weiter verfolgt und gegebenenfalls terminiert werden. Der gesamte Simulationsraum wird typischerweise in Stufen/Level oder Ebenen unterteilt, wie es beispielsweise bei Wetter- bzw. Reliefkarten üblich ist. Dabei liegt der Startpunkt auf der niedrigsten Stufe, das seltene Ereignis auf der höchsten Stufe. An dem Punkt, wo ein Simulationsprozess eine höhere Stufe erreicht, wird er auf mehrere Prozesse aufgeteilt[3]. Aus den sogenannten *Splitting Points* wurde auch der Name der Methode abgeleitet. Alle Simulationsprozesse, die nach einer gewissen Zeit keine höhere Stufe erreicht haben, können das seltene Ereignis der Idee nach nicht mehr erreichen. Auf diese Art und Weise entsteht ein künstlicher Drift in Richtung eines seltenen

[3] im Sinne von geklont

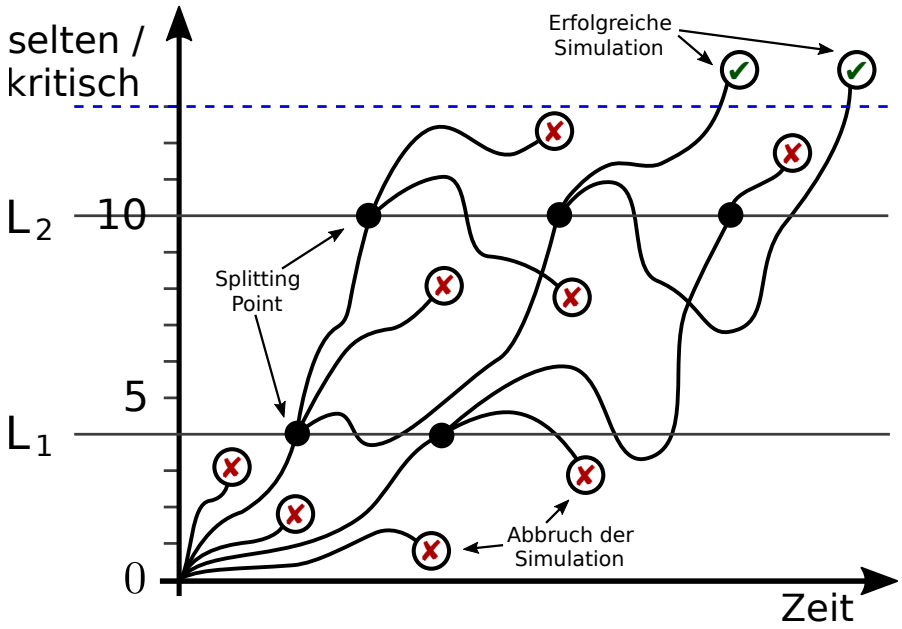

Abbildung 2.2: Prinzip des Importance Splitting, angelehnt an [BDH19].

Ereignisses. Das verfälschte Ergebnis kann am Ende durch Multiplikation des Beitrags jeder Trajektorie mit dem entsprechenden Gewicht wiederhergestellt werden [RT09].

Eine Variante des ISp ist die RESTART Methode (REpetitive Simulation Trials After Reaching Thresholds), welche zum ersten Mal 1991 von Manuel und José Villén-Altamirano in [VV91] vorgestellt wurde. Im Wesentlichen wird dabei jeder Simulationspfad, der einen höheren Level erreicht mit einem festen Faktor aufgesplittet und der originale Pfad wird als solcher gekennzeichnet. Wenn anschließend ein kopierter Pfad wieder einen tieferen Level erreicht, wird er verworfen. Nur der originale darf in solch einem Fall weiterlaufen. Das Verfahren ist rekursiv und mit einer Tiefensuche implementiert [LDT06]. In den folgenden Jahren wurde das Verfahren sukzessive verbessert und sowohl im Kontext von Statistischem Modell Checking (SMC) in [JLS13] angewendet, als auch zur Simulation seltener Ereignisse, bei der die Funktion zum Führen der Simulation automatisch aus der zugrundeliegenden Modellbeschreibung mittels eines Graphen gewonnen wird [BDH15]. Die Möglichkeit einer automatischen Generierung darf allerdings nicht über den Umstand hinwegtäuschen, dass die zugrundeliegende Modellklasse eine entsprechende graphische Repräsentation ermöglichen muss. Im Rahmen einer Kosimulation unterschiedlicher Modellklassen, welche gegebenenfalls Nebenläufigkeiten und als Blackbox einzubeziehende Fahrsimulationssoftware enthalten kann, dürfte ein derartiger Automatismus kaum realisierbar sein.

Die nach wie vor größte Schwierigkeit beim Importance Splitting besteht also darin, eine geeignete Funktion (*importance function*) zu finden, welche die Level bestimmt, an denen

das Splitting stattfindet. Der Umstand lässt sich mit dem Bestimmen eines geeigneten Importance Faktors beim Importance Sampling vergleichen (vergleiche Unterabschnitt 2.5.2). So versucht die Arbeit von Gollücke [Gol16] beispielsweise, mit Hilfe von Distanzmaßen die Nähe eines Systemzustands zu einem seltenen Ereignis zu bestimmen, um damit Kosimulationen in risikoreiche Situationen zu führen. Grundlage der sogenannten Risikodistanzfunktionen ist aber sowohl hinsichtlich der Struktur der Funktionen als auch der verwendeten Subdistanzfunktionen ein manueller Prozess eines Systemexperten. So wird zur Bestimmung der Struktur einer Risikodistanzfunktion eine manuelle Verhaltens- und Gefahrenbeschreibung in einem Prozessmodell genutzt, aus der in einem Zwischenschritt zunächst Fehlerbäume und anschließend die Struktur selbst generiert werden. Auch die zusätzlich berücksichtigten Subdistanzfunktionen gehen auf eine Spezifikation eines Experten durch einen Ursachen- und Relevanzraum zurück. Die in diesem Prozess gewonnene Risikodistanzfunktion liefert letztlich einen kontinuierlichen Wert zwischen 0 und 1 zurück, welcher über einen Ähnlichkeitsvergleich in einer Situationsdatenbank diskretisiert wird und über einen Abbruch bzw. eine Fortführung einer Simulation entscheidet. Die diskreten *Splitting Points* des Verfahrens, welche aus der kontinuierlichen Risikodistanzfunktion abgeleitet werden, müssen ebenfalls aus Expertenwissen gewonnen werden. Dieser Umstand ist jedoch eine generelle Einschränkung von Importance Splitting im direkten Vergleich zu Importance Sampling. Es besteht die Notwendigkeit der Zerlegung des Zustandsraumes in Unterräume, so dass die Wahrscheinlichkeit einen höheren Level zu erreichen auf der einen Seite nicht zu hoch und auf der anderen Seite nicht unmöglich ist. Auf der technischen Seite muss bei dem Vorteil der Unterstützung von Blackbox-Modellen auch einschränkend darauf hingewiesen werden, dass simulierte Prozesse an einer höheren Stufe auch geklont werden müssen, was nicht generell bei Simulatoren zu jedem Simulationszeitpunkt umsetzbar ist.

2.6 Zusammenfassung

Im Hinblick auf die Beantwortung der Forschungsfrage wurde zunächst Simulation im Kontext modellbasierter Entwicklung eingeführt. Um einem Assistenzsystementwickler das Testen seines Systems mit anderen ausführbaren Modellen an den Ein- bzw. Ausgabeschnittstellen zu ermöglichen, wurden mit der HLA und FMI zwei Techniken vorgestellt, heterogene Systemklassen in einer Kosimulation – bestehend aus Fahrsimulationssoftware, Fahrermodell und Fahrerassistenzsystem – zusammenzuschalten. Für eine praktische Realisierung wurde die HLA ausgewählt.

Da Unfälle mit Personenschaden – gemäß aktueller Statistik in Deutschland – betrachtet auf eine Einzelperson ein seltenes Ereignis darstellen, wurde die Problematik aufgezeigt, seltene Ereignisse simulativ im Rahmen einer Kosimulation sichtbar zu machen. Die in einem derartigen Simulationsaufbau entstehende Größe des Zustandsraumes schließt allerdings zeit- und

ressourcen-bedingt Methoden für eine vollständige Überprüfung aus, so dass ein Rückgriff auf eine statistische Analyse notwendig ist.

Um mit statistischem Model Checking die für eine aussagekräftige Statistik notwendige Anzahl an seltenen Ereignissen zu generieren, wurden drei Methoden zur Optimierung von Simulationszeit vorgestellt. Gegen die Nutzung der Methode Counterexample-Guided Abstraction Refinement in einer Kosimulation heterogener Systemklassen spricht das Problem der Gewinnung eines geeigneten und ausführbaren Modells, welches von den kosimulierten Modellen gemeinsam abstrahiert. Der CEGAR Ansatz wird daher im weiteren Verlauf der Arbeit nicht weiter betrachtet.

Die verbleibenden Methoden des Importance Sampling und Importance Splitting eignen sich konzeptbedingt beide für einen Einsatz innerhalb einer Kosimulation, haben allerdings ihre jeweiligen methodischen bzw. technischen Vor- und Nachteile. Die Entscheidung, für den weiteren Verlauf der Arbeit den Fokus auf IS statt auf ISp zu legen, leitet sich aus den Ausführungen in Abschnitt 2.2 auf Seite 10 ab. Unter der gesetzten Annahme, dass ein Entwickler von Fahrerassistenzsystemen im Laufe seiner Karriere aufgrund der stetig zunehmenden Expertise eine Präferenz für ein Werkzeug hat, wird der Wunsch nach einer Verifikation seines Systems innerhalb der gleichen Entwicklungsumgebung abgeleitet. Bei einer entsprechenden Übertragung dieser Sichtweise auf die Entwickler von Fahrzeugdynamikmodellen, ausführbaren Menschmodellen, Umwelt- und Umgebungsmodellen müsste bei einer Simulationszeitoptimierung auf der Basis von auf ISp für alle Modelle eine Möglichkeit gefunden werden, sie an den der Methode inhärenten Splitting Points nicht nur neu zu instanziieren, sondern auch zu vervielfältigen. Unter der Vernachlässigung von ressourcentechnischen Aspekten (im Sinne von Computerhardware und Softwarelizenzen), bleibt immer noch die softwaretechnische Anforderung an die einzelnen Modellklassen bestehen. Da insbesondere die derzeit am Markt verfügbare Fahrsimulationssoftware wie [STS11; WIV19; VIR19] nicht auf derartige Voraussetzungen ausgelegt ist, wird im weiteren Verlauf der Arbeit die Methode des Importance Sampling verwendet. IS erlaubt durch mathematische Rückrechnung (siehe Gleichung 2.9) die simulative Schätzung der Wahrscheinlichkeit seltener Ereignisse, so dass im Anschluss an die Simulation eine quantitative Aussage – zum Beispiel durch eine Konfidenz – gegeben werden kann. Einem Fahrerassistenzsystementwickler ermöglicht die Anwendung dieser Methodik beispielsweise den Nachweis einer Sicherheitseigenschaft für ein (sicherheitskritisches) Fahrerassistenzsystem.

2.7 Verwandte Arbeiten

Bevor es in den folgenden Kapiteln um die Entwicklung eines eigenen Ansatzes und eine experimentelle Evaluation geht, soll noch einmal ein vergleichender Überblick zu verwandten Arbeiten gegeben werden. Dieser andere Blickwinkel soll im Anschluss hilfreich sein, den

Ansatz der vorliegenden Arbeit besser einzuordnen und zu verstehen. Das Ziel der Arbeit ist eine Methodik, welche den Entwicklungsprozess von sicherheitskritischen Fahrerassistenzsystemen verbessern kann. Die möglichen technischen und methodischen Voraussetzungen wurden in den vorhergehenden Abschnitten beschrieben. Für einen direkten Vergleich ist zu berücksichtigen, dass jede Technik bzw. Methodik für sich ebenfalls Gegenstand fortwährender Forschung ist, so dass nicht alle aktuellen Arbeiten für eine Abgrenzung berücksichtigt werden konnten.

Der Fokus soll daher im Folgenden auf eine geführte Kosimulation gelegt werden, mit der seltene Ereignisse sowohl in qualitativer als auch in quantitativer Hinsicht untersucht werden können. Bedingt durch die Größe des Zustandsraumes bei einer Kosimulation muss auch die Möglichkeit zur Abstraktion berücksichtigt werden und wenn es um die Realisierung der Führung einer Kosimulation geht, müssen auch die dahinterliegenden (ggf. technischen) Voraussetzungen einbezogen werden. In den meisten Fällen wird auf das Wissen von (Domänen-)Experten zurückgegriffen, welches z. B. in Form von Abhängigkeitsgraphen, Fehlerbäumen, Zustandsrelationen, etc. in eine Simulation integriert wird.

Tabelle 2.1: Vergleichbare oder verwandte Arbeiten, die ähnliche Methoden oder Techniken wie die vorliegende Arbeit einsetzen.

Autor	Abstraktion	geführte		kein vordef.	Seltene Ereignisse	
		Kosim.	Sim.	Fehlerbaum	qualitativ	quantitativ
S. Shyam [SB06]	✓	✗	✓	✓	✓	✗
F. De Paula [DH07]	✓	✗	✓	✓	✓	✗
T. Zhang [ZLL10]	✓	✗	✓	✓	✓	✗
O. Gietelink [GDV05]	✗	✗	✓	✓	✓	✓
Y. Hu [Hu05]	✓	✗	✓	✗	✓	✓
P. Zuliani [ZBC12]	✓	✗	✓	✓	✓	✓
V. Gollücke [Gol16]	✓	✓	✓	✗	✓	✗

Allen in Betracht gezogenen Arbeiten in Tabelle 2.1 ist gemein, dass sie sich im simulativen Kontext mit dem Finden von seltenen Ereignissen in qualitativer Hinsicht befassen und dabei bis auf eine Ausnahme von einer Abstraktionsmöglichkeit des Zustandsraumes Gebrauch

machen, um die Komplexität zu reduzieren. Die Arbeiten von [SB06; DH07; ZLL10], welche auf CEGAR (Unterabschnitt 2.5.1) zur Abstraktion setzen, finden sich alle im weiten Kontext von Hardwaredesign wieder. Dabei geht es rein um das Auffinden von bestimmten Zuständen (seltenes Ereignis), eine quantitative Analyse ist dabei irrelevant. Der Simulationsablauf selbst besteht zwar aus einer Kombination mehrere Tools wie z. B. Simulator und Simulationscontroller, eine Kosimulation wie sie im Kontext dieser Arbeit definiert wird (vergleiche Abschnitt 2.2), hat dabei aber keine Relevanz. Auch die Arbeiten von Gietelink [GDV05; GDV06], welche seltene Ereignisse bei der Nutzung eines Abstandsregeltempomaten (Adaptive Cruise Control kurz: ACC) untersuchen, nutzen keinen Kosimulationsaufbau, sondern ein sehr einfaches Funktionsmodell eines ACCs (vergleiche Seite 23).

Etwas ausführlicher sollen an dieser Stelle noch einmal die Arbeiten von [Hu05; ZBC12; Gol16] hinsichtlich gemeinsamer Ansätze und Methoden in chronologischer Reihenfolge betrachtet werden. Das Ziel von Hu in seiner Dissertation war die Entwicklung eines integrierten Frameworks für die allgemeine Dynamische Probabilistische Risikoanalyse (DPRA) einschließlich einer effizienten modellbasierten Simulationsumgebung zur quantitativen Risikobewertung komplexer Systeme aus Hardware, Software und menschlichen Elementen [Hu05]. Sein Fokus liegt wie bei der vorliegenden Arbeit darauf, die Sicherheit (im Sinne von *safety*) in technischen Systemen zu verbessern. In einer komplexen Simulation, die den Start eines Space Shuttle in der Startphase nachbildet, werden Hardwarefehler, Fehlfunktionen in der Software sowie menschliches Versagen nachgebildet. Die Steuerung der Simulation in Richtung seltener Ereignisse übernimmt in seinem SIMPRA (für Simulation-based probabilistic risk assessment) genannten Framework ein Planer, welcher auf Fehlerbäume zurückgreifen kann, die auf Expertenwissen basieren. In einer in weiten Teilen deterministischen Simulation der physischen Gegebenheiten des Systems unter Test werden sowohl Hardwarefehler als auch ungewisse Handlungen menschlicher Operateure als probabilistische Elemente abgebildet. Als Methode zur Optimierung der Simulationszeit hat auch Hu sich entschieden, Importance Sampling zu verwenden. Er nutzt dabei das Wissen aus, dass die Wahrscheinlichkeit einen Systemfehler zu erreichen deutlich steigt, wenn das System in Richtung von Komponentenfehlern gebracht wird. Dieser Ansatz funktioniert solange ausreichend gut, wie Systemfehler als eine Kette von Komponentenfehlern interpretiert werden können. Der Aufbau von SIMPRA besteht aus einer Bibliothek, welche es erlaubt, in MATLAB / Simulink Modelle für eine dynamische probabilistische Risikoanalyse zu entwickeln. Hinzu kommen ein Planer und Scheduler zur Simulationssteuerung, welche zwar in Java implementiert wurden, über eine Importfunktion aber direkt in MATLAB genutzt werden können, so dass beim Framework SIMPRA nicht von einem klassischen Kosimulationsaufbau gesprochen werden kann, sondern technisch gesehen eine monolithische Anwendung vorliegt. Eine schnelle und einfache Integration anderer Modellklassen, wie sie beispielsweise im Kontext von Fahrermodellierung zum Einsatz kommt (siehe [WBL13; WLB13]) ist damit nicht möglich. Diese Eigenschaft soll im Rahmen der vorliegenden Arbeit verbessert werden, ebenso wie die Funktionsweise des der Simulationssteuerung zugrundeliegenden Planers. Er

enthält auf der Basis von Ingenieurwissen erstellte Listen mit Interessensszenarien über das zu simulierende System und dient damit als eine Art Karte für die Erkundung während der Simulationsausführung. Wünschenswert wäre hier ein Ansatz, der auch mit geringem Vorwissen auskommt und zum Beispiel aus dem Simulationsverlauf gewonnen werden kann.

Die Arbeit von Zuliani in [ZBC12] präsentiert einen Ansatz, welcher in diese Richtung geht. Sie verwendet im Rahmen von Importance Sampling die Kreuzentropie-Methode [Rub99] zur Generierung einer annähernd optimalen Wahrscheinlichkeitsdichtefunktion, um Statistisches Model Checking von stochastischen Hybriden Systemen zu ermöglichen. Als Anwendungsfall wird ein in Simulink-Stateflow modellierter fehlertoleranter Controller für die hydraulische Höhenrudersteuerung eines Luftfahrzeugs herangezogen. Dabei wurden in die hydraulischen Kreisläufe randomisiert Fehler induziert. Mittels Simulation sollte in diesem Aufbau die Wahrscheinlichkeit ermittelt werden, dass innerhalb eines Zeitintervalls von 25 Sekunden Steuerungseingaben mit der Dauer von einer Sekunde nicht ausgeführt werden (seltenes Ereignis). Zuliani konnte mittels der Kreuzentropie-Methode zeigen, dass mit zunehmender Stichprobengröße der relative Fehler zurückging und dass es bei einer realisierbaren Stichprobengröße von 10^4 möglich ist, Wahrscheinlichkeiten in der Größenordnung von 10^{-14} mit hinreichender Genauigkeit (relativer Fehler = 0,24) zu schätzen. Die Ergebnisse bestätigen aber auch, dass die Parametrisierung der Kreuzentropie-Methode zur Bestimmung einer annähernd optimalen vorgeschlagenen Verteilung (vgl. Unterabschnitt 2.5.2 auf Seite 23) Teil eines Evaluationsprozesses sind, welcher gegebenenfalls länger dauern kann, wenn kein Expertenwissen verfügbar ist. Für die Evaluation des Ansatzes nutzt Zuliani in seinem Anwendungsfall Bounded Linear Temporal Logic (BLTL) als Spezifikationssprache, so dass sich die Wahrscheinlichkeit eines seltenen Ereignisses mit einer BLTL-Formel spezifizieren lässt. Die boolesche Auswertung von Zuständen ist allerdings im Kontext sicherheitskritischer hybrider Systeme, wo u. a. kontinuierliche reellwertige Abstandsmaße eine wichtige Rolle spielen, nicht immer hinreichend. Gerade in Systemen, in denen Umgebungsrauschen (z. B. von Sensoren) oder numerische Fehler (durch eine simulationsbedingt notwendige Abstraktion) auftreten können, ist eine robuste Interpretation, welche quantitativ den Grad einer Erfüllung spezifizieren kann, wünschenswert. Die Handhabbarkeit der Komplexität eines solchen Setups im Kontext einer Kosimulation heterogener Systeme müsste allerdings erneut untersucht werden.

Die Dissertation von Gollücke [Gol16] befasst sich mit der „Bewertung von Simulationszuständen für eine gezielte Analyse risikoreicher Systeme“. Dazu beschreibt er eine Methodik, welche die Distanz zu einem Risiko innerhalb einer simulativen Analyse berechnen kann. Die Bewertung von Systemzuständen hinsichtlich ihrer Nähe zu einer risikoreichen Situation erfolgt mittels Distanzfunktionen. Diese werden aus einer von Domänenexperten gegebenen Verhaltensbeschreibung (in Form eines Fehlerbaumes) abgeleitet und können genutzt werden, um Kosimulationen in risikoreiche Situationen (seltene Ereignisse) zu führen. Für die Evaluation seiner Methodik nutzt Gollücke einen Kosimulationsaufbau in dem dazu

entwickelten Framework DistriCT, welches zur Kopplung von Simulatoren auf die High Level Architecture setzt (vgl. Abschnitt 3.2 auf Seite 42). Anhand eines maritimen Beispielszenarios, welches einen realen Unfall zwischen einem Fracht- und einem Containerschiff nachmodelliert, wird zunächst die Herleitung einer geeigneten (Risiko-)Distanzfunktion validiert. Anschließend wurde in einem zweiten Evaluationsexperiment die zuvor ermittelte Funktion zur Simulationsführung eingesetzt. Bei der zur Optimierung der Simulationszeit eingesetzten Technik wurde das in Unterabschnitt 2.5.3 auf Seite 24 beschriebene Importance Splitting verwendet. An dem Experiment waren zwei Simulatoren zur Steuerung jeweils eines Schiffes beteiligt, welche sich zwar auf einem entgegengesetzten Kurs befanden, aber mit einer Wahrscheinlichkeit von 99,9% einem direkten Kurs zu einem kollisionsfreien Zielwegpunkt folgten. Lediglich mit der Restwahrscheinlichkeit von 0,1% fanden Kursabweichungen zu einer Seite statt. Die qualitative Auswertung der Simulationsergebnisse zeigte, dass eine mittels Risikodistanzfunktion gesteuerte Simulation gegenüber einer naiven Simulation „einen erheblichen Vorteil gegenüber der Anwendung einer naiven Simulation bringen kann.“ [Gol16, S. 139]. Eine quantitative Bewertung der Simulationsergebnisse, d. h. mit welcher Wahrscheinlichkeit eine Schiffskollision zu beobachten ist und mit welcher Konfidenz die Ergebnisse zu interpretieren sind, leistet die Arbeit nicht.

Im weiteren Verlauf der Arbeit wird nun der eigene Ansatz vorgestellt, welcher die in Tabelle 2.1 genannten Eigenschaften vollumfänglich erfüllt.

Kapitel 3

Kosimulationsframework

Im letzten Kapitel wurden die Grundlagen und Methoden beschrieben, die für den eigenen Ansatz relevant sind. Anhand von verwandten Arbeiten wurden Defizite identifiziert, die in den folgenden Abschnitten mit einem eigenen Ansatz geschlossen werden. Dazu wird zunächst ein Kosimulationsframework entwickelt, welches in seiner Instanziierung genutzt werden kann, um im Kontext modellbasierter Entwicklung sicherheitskritischer Fahrerassistenzsysteme statistisches Model Checking auszuführen. Das Framework stellt die technische Grundlage für die methodische Umsetzung der geführten Simulation, um seltene Ereignisse evaluationsgetrieben (qualitativ) sichtbar zu machen und anschließend quantitativ zu bewerten. Ausgehend von einer allgemeinen Beschreibung konzeptueller Simulationsmodelle, die von dem Kosimulationsframework unterstützt werden müssen, schließt sich eine Erläuterung der Modelle an, welche für die spätere Evaluation einer geführten Simulation Anwendung finden. Insgesamt beantwortet das Kapitel die erste der vier Forschungsteilfragen.

3.1 Allgemeiner Aufbau

Bereits im Abschnitt 2.2 (Kosimulation heterogener Systeme) ist beschrieben worden, dass ein Assistenzsystementwickler sein System unter Test mit möglichst realistischen Ein- und Ausgaben an den Modellschnittstellen testen möchte. Je nach Einsatzzweck des Fahrerassistenzsystems (FAS) erfordert die Simulation bestimmter Fahrsituationen eine unterschiedlich detaillierte Abbildung der realen Welt in der virtuellen Simulationsumgebung. Dies geschieht typischerweise durch spezifische Modelle, die darauf ausgelegt sind, eine entsprechende Abbildung der Realität darzustellen. Alle Modelle zusammen laufen in einem sogenannten Simulationsframework. Abbildung 3.1 zeigt exemplarisch einen sehr detaillierten Aufbau eines Simulationsframeworks. Die Eingabedaten auf der linken Seite sind notwendig, um beispielsweise Szenarien zu spezifizieren oder Modelle zu parametrisieren. Auf der gegenüberliegenden Seite entstehen als Ergebnis einer Simulation die Ausgabedaten

Wesentliche Teile dieses Kapitels wurden in [Frä+11; Frä+10; Puc+12a; Pag+15] veröffentlicht.

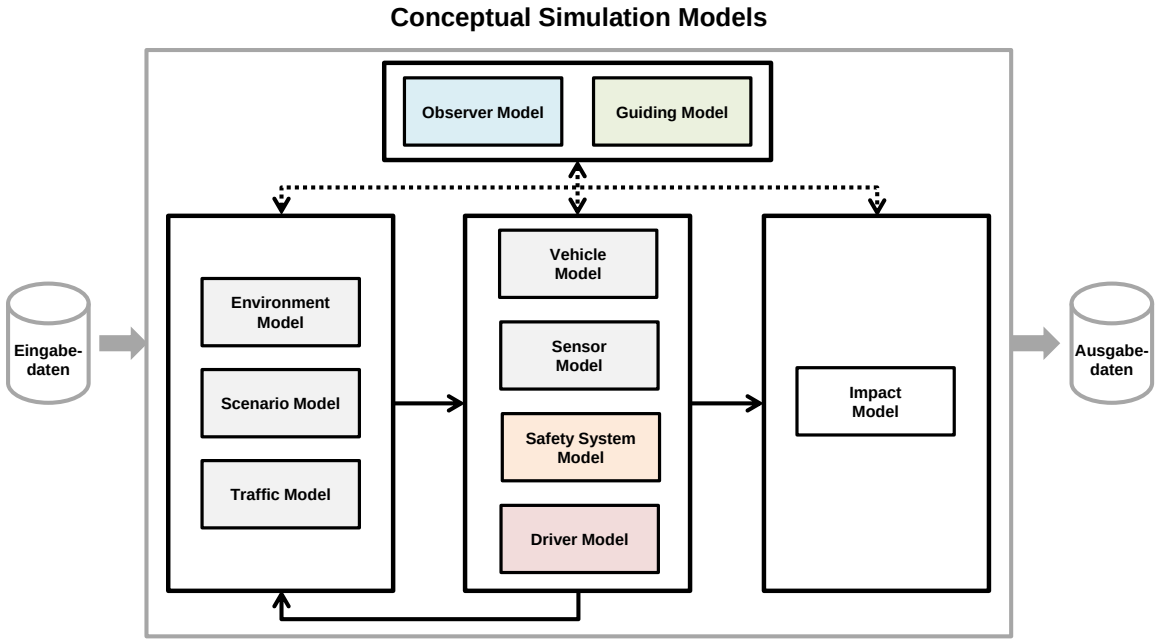

Abbildung 3.1: Exemplarischer Aufbau eines Simulationsframeworks angelehnt an [Pag+15].

wie beispielsweise Variablenbelegungen zur Laufzeit, Fehlermeldungen oder Rohdaten von Sensoren. Innerhalb des Simulationsframeworks befinden sich unterschiedliche Modelle, die je nach Anwendungskontext für eine simulative Darstellung der physischen Gegebenheiten stehen und beispielhaft nach ihren Abhängigkeiten von anderen Modellen gruppiert wurden. Die durchgezogenen Pfeile zwischen den einzelnen Gruppen veranschaulichen einen möglichen Informations- und Datenaustausch untereinander, die gestrichelten bidirektionalen Pfeile heben die Sonderstellung der beiden Modelle am oberen Rand der Grafik hervor.

Die Observer- und Guiding-Modelle nehmen eine zentrale Rolle in dem Framework ein. Sie dienen zur Simulationsüberwachung im Sinne von Monitoring (Observer Model) bzw. zur Simulationsteuerung (Guiding Model), wobei letzteres die – intelligente – Führung bzw. Beeinflussung während der Laufzeit bedeutet und nicht die Kontrolle einer Simulation hinsichtlich des Startens und Stoppens ihrer Ausführung. Das Guiding-Modell ist für die Methodik der geführten Simulation unverzichtbar.

Dem generellen Datenfluss von links nach rechts folgend beschreibt das Szenariomodell (Scenario Model) das zu analysierende Szenario und beinhaltet z. B. Vorgaben für Längs- und Quersteuerungen (z. B. Geschwindigkeit, Lenkung) der beteiligten Verkehrsteilnehmer. Es kann je Kontext mit einem separaten Verkehrsmodell (Traffic Model) verfeinert werden, welches die Eigenschaften und das Verhalten der anderen beteiligten Verkehrsteilnehmer beschreibt, oder mit einem Umgebungsmodell (Environment Model) ergänzt werden, welches

Eigenschaften wie Straßenmerkmale, Verkehrsregeln, (vorübergehende) statische Objekte, Beleuchtung, Wetter usw. beschreibt.

Im mittleren Teil von Abbildung 3.1 sind ein Fahrer- und Fahrzeugmodell (Driver and Vehicle Model) sowie ein Sensor- und Assistenzsystemmodell (Sensor and Safety System Model) dargestellt. Das Fahrermodell wird verwendet, um den Faktor Mensch in einer Simulation zu berücksichtigen, beispielsweise für eine Analyse, wie ein Autofahrer auf Reize reagiert, die aus der Umwelt (von anderen Verkehrsteilnehmern, Ampeln, Signalen) oder von innerhalb des eigenen Fahrzeugs zu ihm gelangen. Je nach Anwendungsfall oder zu untersuchender Fahraufgabe kann ein Fahrermodell zum Einsatz kommen, das nur die Fahrerreaktion wie Lenkung, Bremsen und Beschleunigen abbilden kann oder wie im Falle von kognitiven Fahrermodellen mit statistischen Verteilungen mentale Prozesse oder die Dauer von Hand- / Augenbewegung oder Informationsaufnahme nachbildet. Das Fahrzeugmodell beschreibt alle relevanten Parameter des verwendeten Fahrzeugs mit den erforderlichen Freiheitsgraden für Fahrwerk, Federung, Reifen, Lenkung und Fahrzeugabmessungen usw. zugehörig zum Fahrzeug. Aufgrund möglicher modellcharakteristischen Eigenheiten sind Funktionsmodelle als separate Modelle repräsentiert. Angefangen von einfachen Sensormodellen können Assistenzsystemmodelle eine Kombination aus Entscheidungsmodell (in Form eines Algorithmus), Betätigungsmodell (Bremsen, Beschleunigen, Lenken, etc.) sowie – falls erforderlich – Kommunikationsmodells (automatischer Notruf) sein.

Sollen bei Simulationen Unfälle und deren Folgen betrachtet werden, kann ein Wirkungsmodell (Impact Model) erforderlich sein, welches die Auswirkungen zwischen Teilnehmern (z. B. Autos, Fußgänger) beschreibt sowie zu erwartende Verletzungen oder Sachschäden.

Im Rahmen dieser Arbeit wird zur Reduzierung der Komplexität auf die Verwendung eines Wirkungsmodells verzichtet und eine commercial off-the-shelf-Fahrsimulationssoftware verwendet, welche die in Abbildung 3.1 grau hinterlegten Modelle direkt beinhaltet. Die übrigen farbig gekennzeichneten Modelle kommen als eigene Instanz entweder bei der späteren Evaluation des Simulationsframeworks oder der daran anschließenden geführten Simulation zum Einsatz. Die dazu verwendeten Softwarekomponenten werden im Folgenden in jeweils einem Unterabschnitt beschrieben.

3.1.1 Fahrsimulationssoftware

Eine Fahrsimulationssoftware bietet neben der Integration der im letzten Abschnitt genannten Teilmodelle zu einer Gesamtsimulation noch weitere Vorteile. Sie ermöglicht eine Kopplung mit weiteren ausführbaren Modellen im Sinne von Software-in-the-Loop oder die Integration von Hardwarekomponenten (Hardware-in-the-Loop), allerdings sind die verfügbaren Schnittstellen auf interne Daten und Variablen zumeist nicht standardisiert, so dass

Abbildung 3.2: Starre „Sitzkiste“ für Simulatorstudien in kleinem Rahmen.

die jeweilige API der Fahrsimulationssoftware entsprechend konfiguriert und eigenentwickelte Modelle adäquat angepasst werden müssen. Gleiches ist zutreffend, wenn statt klassischer Human-in-the-Loop (HITL)-Simulationen, bei der ein Proband das simulierte Fahrzeug (*EgoCar*) mittels Lenkrad und Gas- bzw. Bremspedal kontrolliert, Virtual-Human-in-the-Loop-Simulationen durchgeführt werden sollen, bei der ein Fahrermodell die Steuerung übernimmt. Letzteres hat den Vorteil, dass von speziellen Fahrsituationen zahlreiche Wiederholungen durchgeführt werden können, da anders als bei Nutzerstudien mit Probanden keine Trainingsphasen oder Erholungspausen notwendig sind.

Im Rahmen dieser Arbeit wurde zunächst die ST-Software [STS11] verwendet. Sie wurde für den Trainingszweck in einer Fahrschule konzipiert, um Fahranfängern einen ersten Einstieg in die komplexen Zusammenhänge des Autofahrens zu ermöglichen oder Berufskraftfahrer für spezielle Situationen im Rahmen von Weiterbildungen zu schulen. Da die verfügbare API zum Zugriff auf interne Daten allerdings stark eingeschränkt war, wurde im weiteren Verlauf auf die professionelle Fahrsimulationssoftware SILAB [WIV19] gewechselt, welche u. a. für den Einsatz in umfassenden verkehrspsychologischen Studien mit Probanden ausgelegt ist, und demzufolge eine umfassendere API bereitstellt. Auf diese Weise konnte die gleiche Software sowohl im Rahmen von Nutzerstudien in einer einfachen sogenannten „Sitzkiste“ verwendet werden, dargestellt in Abbildung 3.2, als auch im Anschluss mit einem auf Basis der ausgewerteten Experimentdaten entwickelten Fahrermodell [Wor14].

3.1.2 Fahrermodell

Heutzutage ist menschliches Versagen einer der Hauptfaktoren für Unfälle im Transportwesen. In der Luftfahrt werden 60 - 80% der kommerziellen Flugzeugunfälle [Boe02] und in der Automobilindustrie 88% der Autounfälle [Sta17a] durch menschliche Fehler verursacht. Um die Unfallraten weiter zu reduzieren, werden in Autos und Flugzeugen immer mehr Aufgaben automatisiert. Die zunehmende Automatisierung und der damit einhergehende Rollenwechsel des Bedieners von der aktiven Steuerung hin zur Überwachung und Kontrolle von Abläufen führt allerdings auch neue Risiken für menschliche Fehler ein [SW95]. Aus diesem Grund sind neue Methoden und Techniken erforderlich, um die Einflüsse solcher Systeme auf den Faktor Mensch zu analysieren. Typische Designfragen wie „Wie verändern sich die Aufgaben eines Autofahrers oder Piloten mit neuen Systemen?“, oder „Was passiert, wenn ein System ausfällt?“ müssen beantwortet werden. Aktuelle industrielle Praxis ist der Bau von physischen Mockups mit Prototypen (siehe Abbildung 3.2), um das System mit Menschen (Testfahrer oder Testpiloten) zu testen. Dieser Ansatz ist sehr teuer und zeitaufwendig, daher wird eine Alternative benötigt, die frühzeitig im Entwicklungsprozess einsetzbar ist, beispielsweise eine modellbasierte Mensch-Maschine-Schnittstelle.

Solch eine Schnittstelle stellen ausführbare Fahrermodelle[1] bereit, und je nach Anwendungskontext existieren diverse Möglichkeiten der Detailausprägung. Im einfachsten Fall stellt ein Fahrermodell wie ein Autofahrer im klassischen Sinne lediglich die Stellgrößen für Lenkrad, Gas- und Bremspedal bereit. Sollen aber z. B. Bewertungen der Nützlichkeit eines Fahrerassistenzsystems getroffen werden, für die eine Interaktion mit dem Fahrer erforderlich ist, sind detailliertere Fahrermodelle notwendig.

Für den weiteren Verlauf der Arbeit ist daher ein spezielles kognitives Fahrermodell wesentlicher Bestandteil, dessen Besonderheiten hinsichtlich Struktur und Funktionsweise an dieser Stelle etwas präziser erläutert werden. Wenn von einem kognitiven Fahrermodell gesprochen wird, muss rein begrifflich zwischen zwei Teilen unterschieden werden. Zum einen gibt es eine kognitive Architektur, welche eine Art Framework für einen kognitiven Modellierer zur Verfügung stellt. Zum anderen gibt es die Wissensbasis, welche Verhaltensbeschreibungen für die Simulation während der Laufzeit enthält. Diese muss beim Start einer Simulation in eine kognitive Architektur geladen werden. Eine detailliertere Beschreibung von allgemeinen Konzepten kognitiver Architekturen kann der Arbeit von Weber [Web17, S. 28 ff.] entnommen werden. Kurz zusammengefasst ist allen Architekturen gemein, dass ein sogenanntes Produktionssystem bei der Simulation von *zielgerichtetem*, *regelbasiertem* menschenähnlichem Verhalten, eine zentrale Rolle einnimmt. Mit Zielen (*goals*) werden Intentionen definiert, während Regeln (*rules*) aus einem bedingten Aktionsplan bestehen, welcher einer Reihe von Maßnahmen beinhaltet, die ausgeführt werden müssen, um ein Ziel zu erreichen.

[1] Im weiteren Verlauf wird nur die Domäne des Automobils betrachtet, eine Übertragbarkeit auf andere Domänen wie beispielsweise die Luftfahrt ist aber in weiten Teilen analog möglich.

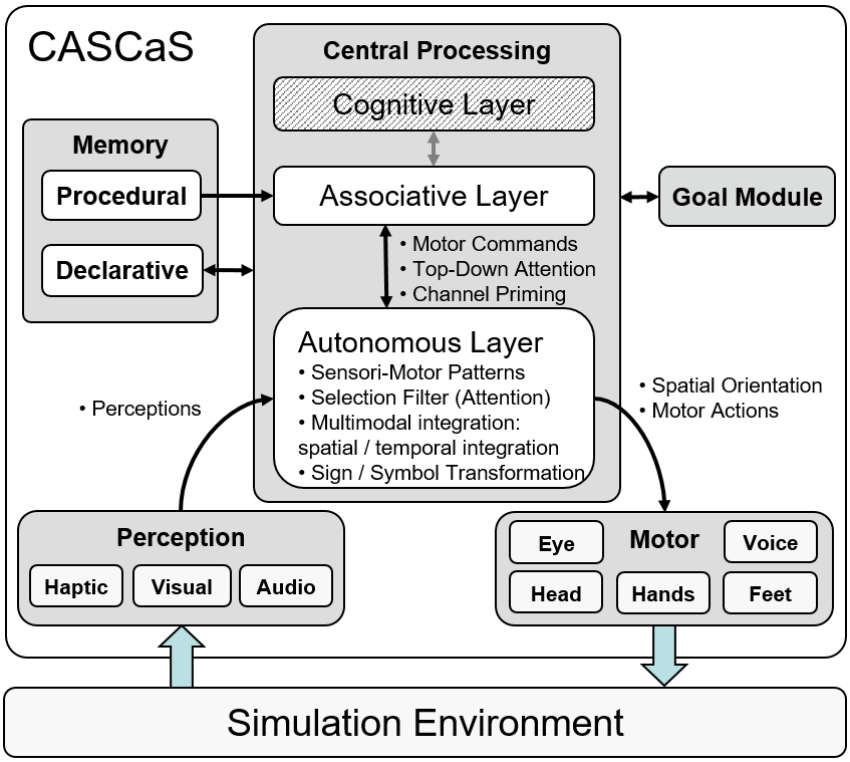

Abbildung 3.3: Komponentenübersicht der kognitiven Architektur CASCaS [Hum18].

Weitere Komponenten repräsentieren das menschliche Gedächtnis sowie den Austausch mit der Umwelt. Die Definition und Beschreibung der Ziele und Regeln selbst erfolgt außerhalb einer Architektur mittels einer Wissensbasis.

Eine komponentenbasierte Übersicht der kognitiven Architektur CASCaS (Cognitive Architecture for Safety Critical Task Simulation) ist in Abbildung 3.3 dargestellt. Sie bildet die Grundlage des im weiteren Verlauf benutzen kognitiven Fahrermodells. Wie der Arbeit von Wortelen in [Wor14] zu entnehmen ist, wurde CASCaS entworfen, um auf einfache und flexible Art kognitive Agenten für sicherheitskritische Umgebunden zu entwickeln. Damit bietet die Architektur einerseits ideale Voraussetzungen für ein Fahrermodell, welches im Rahmen einer Fahrerassistenzsystementwicklung eingesetzt werden soll, und zum anderen wurden bereits Fahrermodelle mit CASCaS modelliert und auch publiziert [Web+09; Wor14].

In CASCaS wurde der für kognitive Architekturen typische Kreislauf aus Wahrnehmung (*Perception*), Kognition und Aktion (*Motor*) umgesetzt, die Synchronisation zwecks Datenaustausch mit der Umwelt ist mit 50 ms getaktet. Über eine Schnittstelle werden Informationen aus der Simulationsumgebung (*Simulation Environment*) durch Wahrnehmungskomponenten z. B. aus dem visuellen oder auditiven Bereich aufgenommen und über die Wis-

sensverarbeitung (*Central Processing*) im Gedächtnis (*Memory*) abgelegt. In der Wissensverarbeitung können diese Informationen anschließend genutzt werden, um die im Zielmodul (*Goal Module*) enthaltenen Ziele mittels Regeln aus dem prozeduralen Gedächtnis zu erreichen. Die Ziele und Regeln müssen wie eingangs beschrieben beim Start einer Simulation aus einer Wissensbasis geladen werden. Die Wissensverarbeitung verändert dann zur Laufzeit die Inhalte des deklarativen Gedächtnisses bzw. die Reihenfolge oder Priorisierung der Ziele und initiiert Aktionen, welche von der Motorik beispielsweise in Form von Augen-, Hand- oder Kopfbewegungen ausgeführt und letztendlich zurück an die Simulationsumgebung propagiert werden. Für weitere technische Details wie Konfiguration und Parametrisierung von CASCaS, oder funktionale Modellaspekte der Wahrnehmungs- und Motorkomponenten, sei an dieser Stelle ebenfalls auf die Arbeit von Wortelen in [Wor14] verwiesen. Wichtig sei an dieser Stelle festzuhalten, dass ein ausführbares, kognitives Fahrermodell, welches mittels einer kognitiven Architektur instanziiert wurde, aus zahlreichen Detailmodellen besteht, die jeweils einzelne Komponenten der Architektur widerspiegeln. Das Fahrermodell instanziiert das in Abbildung 3.1 dargestellte *Driver Model*.

3.1.3 Fahrerassistenzsystem

Die Entwicklung eines Fahrerassistenzsystems im Automobilbereich erfordert bis zur Serienreife umfassende Tests und Bewertungen hinsichtlich seiner korrekten Funktionalität. Da sich die Forschungsfrage dieser Arbeit nur auf die Unterstützung des Entwicklungsprozesses eines Assistenzsystems bezieht (vergleiche Unterabschnitt 1.2.1), soll im Rahmen dieser Arbeit kein Fahrerassistenzsystem (FAS) entwickelt werden, sondern lediglich sichergestellt werden, dass ein ausführbares Modell eines FAS in einer Kosimulation analysiert und evaluiert werden kann. Aus Sicht eines Fahrerassistenzsystementwicklers ist das Fahrerassistenzsystem während der Simulationsphase das sogenannte System unter Test. Im Verlauf der Arbeit beschränkt sich die Unterstützung eines FAS zunächst auf die Integrationsmöglichkeit eines ausführbaren Modells in das Kosimulationsframework, welches in dem weit verbreiteten Modellierungswerkzeug MATLAB / Simulink / Stateflow konzipiert wurde. Stellvertretend für die Interaktionen eines FAS mit einem Nutzer wird eine separate Aufgabe im Fahrermodell benutzt. Die tatsächliche Umsetzung einer Bedienprozedur in CASCaS zur Interaktion mit einem konkreten FAS ist nicht Gegenstand dieser Arbeit. Das Fahrerassistenzsystem instanziiert das in Abbildung 3.1 dargestellte *Safety System Model*.

3.1.4 Onlinemonitoring

Zur automatischen Analyse von speziellen Ereignissen oder bestimmten Situationen während der Ausführung von Simulationen – dem sogenannten *Onlinemonitoring* – können sogenannte *Observer* eingesetzt werden (vergleiche *Observer Model* in Abbildung 3.1). Die

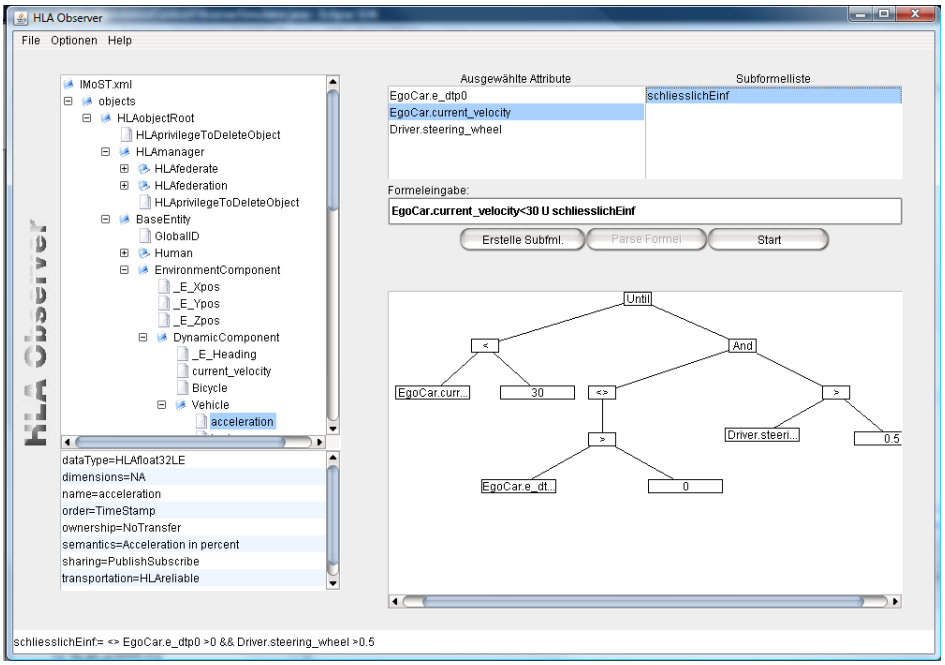

Abbildung 3.4: Screenshot des HLAObservers aus der Masterarbeit von Gezgin [Gez09].

Komplexität eines Observers kann je nach Anwendungsfall stark variieren. Beginnend mit der einfachen booleschen Überwachung einer Variablen hinsichtlich ihrer erfüllenden Belegung kann ein Observer darauf achten, ob bestimmte Sequenzen oder zeitlich bedingte Abhängigkeiten von Variablen eintreffen bzw. die Abwesenheit von Ereignissen nachverfolgen. So hat Gezgin im Rahmen seiner Masterarbeit [Gez09] das in Abbildung 3.4 dargestellte Tool entwickelt, welches es erlaubt, Formeln in der Sprache der Quantitative Linear Temporal Logic (QLTL) zu definieren und online zu überwachen. Eine quantitative Auswertung einer QLTL-Formel über einen Simulationslauf erlaubt neben der Aussage, ob es eine erfüllende Belegung gibt (oder nicht), auch zu welchem Grad (im Sinne einer Robustheit) die jeweilige Aussage zutreffend ist. Das Ergebnis eines Onlinemonitors kann für spätere Offline-Analysen von Simulationsläufen genutzt werden oder bei geeigneter Implementierung auch als zusätzlicher Input für nachfolgende Simulationsläufe dienen, beispielsweise im Rahmen von Batchsimulationen. Die in Abbildung 3.4 exemplarisch dargestellte Formel überwacht, ob das EgoCar bei der Auffahrt auf die Autobahn solange unterhalb einer Geschwindigkeit von 30 m/s bleibt (EgoCar.current_velocity < 30), bis schließlich ein Spurwechsel vom Beschleunigungsstreifen auf die rechte Fahrspur der Autobahn durchgeführt wurde ($\cup$ schliesslichEinf). Das $\cup$ entspricht dabei dem temporal-logischen *Until-Operator*.

3.1.5 Simulationsführung (Guiding)

Das Steuern oder Führen (*Guiding*) einer Simulation ermöglicht die Beeinflussung oder Anpassung von Parametern während der Ausführung bzw. zwischen einzelnen Simulationsläufen. Die Eigenschaft, welche mit Simulationsführung adressiert wird, geht dabei über das einfache Starten, Stoppen oder Pausieren einer Simulation hinaus und beinhaltet eine gewisse Intelligenz. Da für das intelligente Führen einer Simulation unterschiedlichste Informationen und Daten hilfreich sein können, steht das Führungsmodul (im weiteren Verlauf auch Simulationsguide genannt), ähnlich dem Modul zum Onlinemonitoring, in Abbildung 3.1 als *Guiding-Model* oberhalb der klassischen Modelle eines Simulationsframeworks an zentraler Stelle. Das Führen einer Simulation wird im Kontext dieser Arbeit eingesetzt, um bei der Simulation seltener Ereignisse die notwendige Simulationszeit zu reduzieren. Die Instanziierung dieses Modells in Form eines Algorithmus wird im späteren Verlauf der Arbeit in Abschnitt 4.2 beschrieben.

3.2 Technische Umsetzung einer Kosimulation

Die Kopplung verschiedener Simulatoren und der Aufbau einer verteilten Simulation ist eine komplexe und lizenztechnisch gesehen möglicherweise teure Aufgabe. Besonders in Forschungsprojekten, bei denen schnelle Ergebnisse erwünscht sind, ist diese Aufgabe oft zu zeitaufwendig. Die Verwendung von Normen und bestehenden Frameworks kann in diesem Kontext hilfreich sein. Der IEEE 1516 Standard, auch bekannt unter dem Namen „High Level Architecture (HLA)“, ist ein etabliertes Konzept für verteilte Simulationen, allerdings ist HLA nicht besonders gut für Rapid Prototyping geeignet. In den folgenden Abschnitten wird daher beschrieben, wie HLA für eine Virtual-Human-In-The-Loop Simulation genutzt wird und wie ein selbst entwickeltes Framework dabei hilfreich ist, um eine schnelle Anbindung neuer Simulatoren zu ermöglichen. Eine abschließende Bewertung des Frameworks zeigt, dass eine Open-Source Runtime Infrastuktur (RTI) im Kontext der Forschung gut mit einer kommerziellen Implementierung konkurrieren kann.

Um die in Unterabschnitt 3.1.2 beschriebene kognitive Architektur CASCaS schnell mit unterschiedlichen Umgebungssimulationen wie X-Plane (als Flugsimulator [Lam]) oder SILAB (als Fahrsimulator [WIV19]) zu koppeln ist viel Entwicklungszeit notwendig, da sich ein Entwickler jedes Mal um den Datenaustausch und die Zeitsynchronisation für jeden Simulator kümmern muss. Eine Neuimplementierung ist kontraproduktiv und die Verwendung eines bestehenden Frameworks oder Standards ist wünschenswert. Nach Wickens und anderen in [Wic+07] dauerte die Kopplung der kognitiven Architekturen ACT-R, Air-MIDAS, D-OMAR, IMPRINT/ACT-R mit der Simulationsumgebung etwa 10-35% der Entwicklungszeit (nur für die Kommunikation). Dabei ist zu berücksichtigen, dass jeder Simulator über

eine dedizierte Schnittstelle verbunden war und nicht über ein Framework oder eine Norm. Für die Kopplung der kognitiven Architektur CASCaS mit Simulationsumgebungen fiel die Entscheidung daher auf die HLA, welche Mechanismen für die Zeitsynchronisation, den Datenaustausch sowie andere nützliche Funktionen zur Verfügung stellt. Nachteilig an der HLA ist die Komplexität ihrer Schnittstellen und der damit verbundenen steilen Lernkurve. Aus diesem Grund wurde in [Wic+07] keine HLA verwendet, obwohl für einige Simulatoren eine HLA-Schnittstelle vorhanden gewesen wäre. Das Ziel in diesem Zusammenhang war es daher, eine minimale Teilmenge der benötigten HLA-Funktionen zu identifizieren und eine einfach zu bedienende Schnittstelle bereitzustellen, die Simulator-Experten dabei unterstützt, einen Simulator schnell mit der High Level Architecture zu koppeln.

Aufbau der High Level Architecture

Die Basisdefinition der HLA wurde 1996 unter Federführung des DMSO (Defense Modeling and Simulation Office) unter dem allgemeinen Ziel der Wiederverwendung bestehender Simulatoren erstellt. HLA ist als Nachfolger des Distributed Interaktive Simulation (DIS)-Protokolls [IEE12] bekannt und eliminiert einige seiner Schwächen, z. B. durch die Unterstützung objektorientierter Konzepte und gerichteter Kommunikation anstelle von Broadcast. Der Standard besteht insgesamt aus drei einzelnen Dokumenten. Das erste „Rahmen und Regeln“ definiert die HLA, seine Komponenten und Regeln, welche die Teilnehmer einer HLA-Simulation, die sogenannten Föderaten (*Federates*), zu berücksichtigen haben und die Gesamtsimulation, welche als Föderation (*Federation*) bezeichnet wird, siehe Abbildung 3.5.

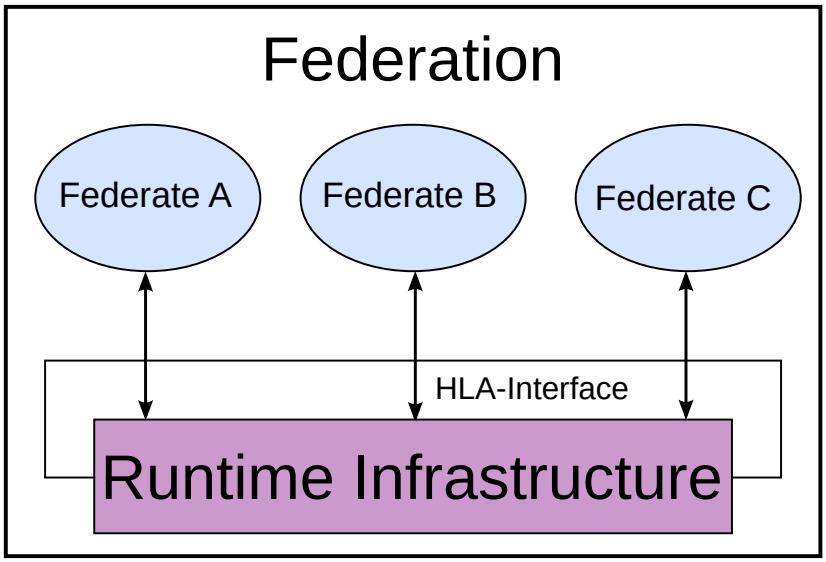

Abbildung 3.5: Übersicht einer HLA Simulation [Puc+12a].

Der zweite Teil bietet einen gemeinsamen Rahmen für die Zusammenschaltung von interagierenden Simulationen in Form einer Schnittstellenspezifikation und der dritte Teil definiert die Syntax sowie das Format des Object Model Template (OMT), welches verwendet wird, u. a. die Daten, Interaktionen und Synchronisationspunkte zu definieren, welche zwischen allen Föderaten während der Ausführung einer Föderation benutzt werden.

Das Simulation Object Model (SOM) beschreibt die Objekte, die von einem einzigen Föderaten ausgetauscht werden können, während das Federation Object Model (FOM) eine gesamte Simulation beschreibt, so dass ein FOM eine Obermenge aller SOMs darstellt. Da die HLA nur die Definition einer Architektur spezifiziert, wird eine Software zum Ausführen einer HLA-Simulation benötigt, die so genannte Runtime Infrastructure (RTI), siehe Abbildung 3.5. Zwar ist im IEEE-Standard keine Referenzimplementierung verfügbar, es existieren aber sowohl von kommerziellen Anbietern wie Pitch oder Mäk Implementierungen, als auch Open-Source-RTIs wie zum Beispiel CERTI. Ein Problem der Open-Source-Implementierungen ist, dass sie nicht alle Schnittstellen vollständig zur Verfügung stellen, dass sie nicht zertifiziert sind, HLA-konform zu sein, und ein Benutzer keinen Herstellersupport erwarten kann. Darüber hinaus gibt es weitere Nachteile für den Einsatz von HLA in Forschungsprojekten. Insbesondere für Einsteiger stellt die komplexe Funktionalität von HLA mit ca. 60 verschiedene Callbacks, die in der aktuellen Version der Standards IEEE 1516-2010 [IEE10] (genannt „HLA Evolved“) implementiert werden müssen, eine hohe Akzeptanzschwelle dar und auch für Rapid Prototyping ist die HLA nicht gut geeignet. Die Möglichkeit des Wechsels zwischen verschiedenen RTI-Implementierungen war zumindest beim Standard IEEE 1516-2000 sehr aufwendig, da die Interface Specification mehrere Fehler hatte, die zu zwei unterschiedlichen Interpretationen des Standards führte – eine vom Department of Defense (DoD) und eine von der Simulation Interoperability Standards Organization (SISO). Die Umstellung der RTI von einer Interpretation auf die andere erfordert auf Grund unterschiedlicher Namensräume eine Neukompilierung und bei einem Wechsel zwischen RTIs mit der gleichen Interpretation muss zumindest der Quellecode neu verknüpft (im Sinne von Linken) werden [Läs11]. Der HLA Evolved Standard zielt darauf ab, diese Art von Implementierungsproblemen zu lösen. Eine weitere Schwierigkeit besteht darin, dass es bis auf die Ausnahme des Real-time Platform Reference Federation Object Model im militärischen Kontext [SIS15], derzeit keine standardisierten OMTs gibt.

Das folgende Konzept ist ein Ansatz, um mit den genannten Schwierigkeiten der HLA umzugehen und eine Implementierung zur Beseitigung der meisten von ihnen zur Verfügung zu stellen, so dass sich Entwickler auf die Vorteile der HLA konzentrieren können wie eine standardisierte API und TimeManagement Services, die verschiedene Zeitmodelle unterstützen, z. B. schrittweises (mit festen oder variablen Schrittgrößen) oder ereignisbezogenes Voranschreiten.

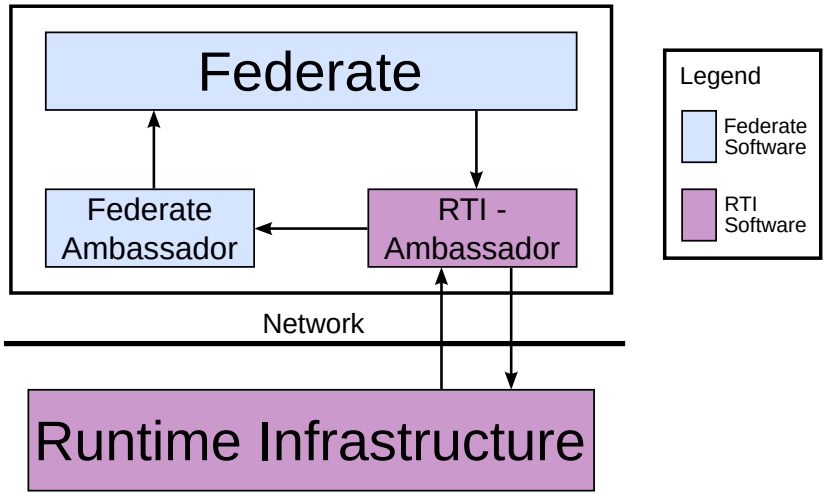

Abbildung 3.6: Das Botschafterkonzept einer HLA Simulation [Puc+12a].

Der HLAWrapper als minimale Teilmenge der High Level Architecture

Für ein besseres Verständnis, welche minimale Funktionalität der komplexen HLASchnittstelle notwendig ist, um eine HLA-Simulation durchführen zu können, ist es zunächst notwendig, das sogenannte Botschafterkonzept (*Ambassador concept*) zu betrachten, siehe Abbildung 3.6.

Der *Federate Ambassador* eines Föderaten ist aus Implementiertungssicht eine abgeleitete Klasse des *RTI Ambassador* und muss alle Callback-Funktionen implementieren, die von der Runtime Infrastuktur (RTI) gemäß des HLA-Standards aufgerufen werden können. Der *RTI Ambassador*, welcher Teil der RTI ist, hat eine Referenz auf den *Federate Ambassador* eines Föderaten und kann damit unterschiedliche Funktionsaufrufe während einer Simulationsausführung durchführen. Es ist jedoch nicht notwendig, alle Callback-Funktionen der komplexen HLA-Schnittstelle zu implementieren, wenn das Ziel nicht die Realisierung eines HLA-zertifizierten Föderaten ist, sondern eine schnelle Integration eines Simulators in eine HLA-Simulation. Dienste für das Speichern oder Wiederherstellen einer Föderation sind ebenso optional wie zahlreiche andere (z. B. *Ownership Management*) und können unberücksichtigt bleiben. Der Fokus liegt an dieser Stelle auf einer minimalen Teilmenge von essentiellen Callback-Funktionen, beispielsweise zum Erstellen und Zerstören, Beitreten und Verlassen einer Föderation, dem Abonnieren des Datenaustausch- oder Zeitmanagementdienstes, da optionale Callback-Funktionen auch zu einem späteren Zeitpunkt, wenn sie benötigt werden, hinzugefügt werden können. Diese minimale Teilmenge ist für alle Föderaten identisch und muss immer implementiert werden, um eine Simulationskomponente an einer HLA-Simulation teilnehmen lassen zu können. Ein weiterer wichtiger Aspekt ist der Datenaustausch zwischen verschiedenen Föderaten, weil sich die lokale Datenstruktur eines

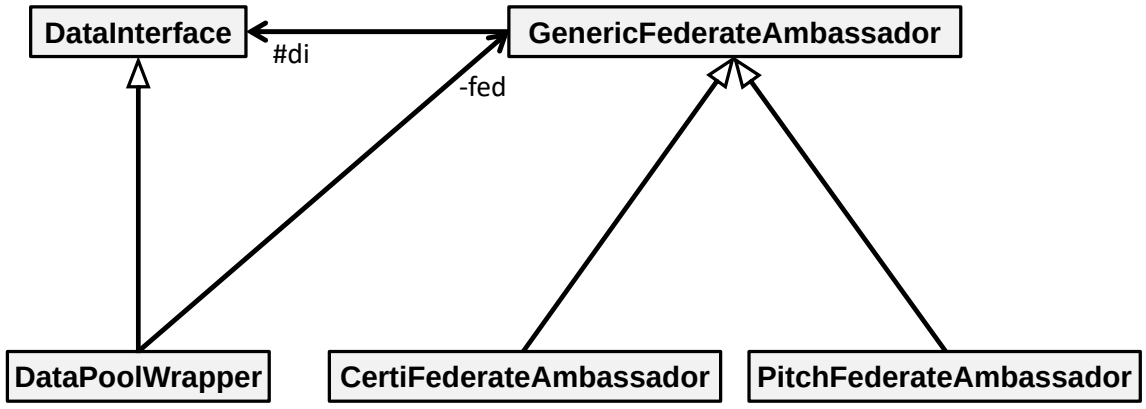

Abbildung 3.7: Framework für die Wiederverwendung einer HLA-Implementierung in unterschiedlichen Simulationskomponenten.

jeden Föderaten im Allgemeinen unterscheidet. Während die Funktionen zum Versenden und Empfangen von Daten und Nachrichten einheitlich sind, ist die Integration der auszutauschenden Daten in das Format, welches durch die Struktur des OMT im HLA-Standard vorgegeben ist, für jeden Simulator individuell umzusetzen.

Das folgende in Abbildung 3.7 dargestellte Klassendiagramm des Frameworks – HLAWrapper genannt – kapselt in seiner Implementierung die Callback-Funktionen des Datenaustauschdienstes in einem Dateninterface, um die Wiederverwendbarkeit des Quellcodes innerhalb des Frameworks zu maximieren. Die kontext- und domänenabhängigen OMTs Definitionen in der Luft- und Raumfahrt, Automotive oder Navigation werden unterstützt, indem beim Start einer Simulation die lokalen Datenstrukturen für Klassen, Nachrichten und Objektinstanzen aus den SOMs eines Föderaten generiert werden. So kann eine Implementierung auch für verschiedene OMTs in wechselnden Domänen wiederverwendet werden und ist darüber hinaus im Zusammenspiel mit unterschiedlichen RTIen einsetzbar.

Derzeit ist der HLAWrapper für die RTI von Pitch in der Version 4.4.2 [Pit] und CERTI in der Version 3.4.2 [CER] geeignet. Die Klasse *GenericFederateAmbassador* ist nicht spezifisch für eine RTI und beinhaltet daher allgemeine Daten wie Name des Föderaten, Name der Föderation, Einstellungen für Zeitmanagementdienste, etc., die für jede Föderation notwendig sind. Auch die beim Start einer Simulation aus den domänenabhängigen OMTs zu generierenden Datenstrukturen werden hier in abstrakter Art definiert. Parallel dazu ist auf gleicher Ebene die abstrakte Klasse *DataInterface* abgebildet, die als Klassenattribut den Datenaustausch von und zu einer Simulationskomponente übernimmt. In der ersten Vererbungsstufe sind die RTI-spezifischen Unterschiede berücksichtigt wie Namensräume und verschiedene Implementierungen von logischen Zeitwerten. Die Kapselung der verschiedenen Callback-Funktionen erfolgt aus dem IEEE 1516-2000 Standard im *CertiFederateAmbassador* auf der einen Seite und dem IEEE 1516-2010 Standard im *PitchFederateAmbassador* auf der an-

deren Seite. Zusätzlich enthalten beide Klassen Funktionen zum Starten und Stoppen eines Föderaten, zum initialen Synchronisieren beim Simulationsstart und zum Beantragen eines Zeitfortschritts während einer Simulation. Die Klasse *DataPoolWrapper* stellt eine mögliche Implementierung der abstrakten Klasse *DataInterface* bereit. Der vom Wrapper benutzte Datenpool stellt eine einfache Schnittstelle zum Speichern und Abrufen von Variablen zur Verfügung. Das bedeutet, dass die originalen Callback-Funktionen des HLA-Standards für den jeweiligen Simulator versteckt (im Sinne von gekapselt) sind, was eine Kopplung mit der HLA wesentlich vereinfacht. Der Wrapper selbst hat wiederum die Möglichkeit, in Form eines Klassenattributes über den *Certi-* bzw. *PitchFederateAmbassador* auf die entsprechende RTI zuzugreifen.

Mit diesem Framework wird die Integration eines neuen Simulators in eine HLA-Simulation auf einen minimalen Implementierungsaufwand reduziert und gleichzeitig bietet es einem Entwickler zwei Möglichkeiten in Abhängigkeit von der Nutzung des abstrakten Dateninterfaces:

1. Integration der lokalen Simulationsdaten mittels der Schnittstellen der Klasse DataPoolWrapper und dem Aufruf der Wrapperfunktionen in den Klassen *CertiFederateAmbassador* und *PitchFederateAmbassador* während der Simulationsausführung.

2. Integration der von der RTI bereitgestellten Daten mittels der Funktionen aus dem abstrakten Dateninterface direkt in die lokale Datenstruktur des Simulators und Aufruf der Wrapperfuktionen (wie in Option 1) bzw. direkter Aufruf der originalen HLA-Funktionen des RTI-Ambassador. Letzteres impliziert mehr Arbeit bei der Implementierung, stellt aber innerhalb des Frameworks die Kompatibilität mit dem HLAStandard sicher.

Der HLAWrapper beschränkt den notwendigen Implementierungsaufwand für die Integration einer neuen Softwarekomponente in die HLA auf die Implementierung der Dateninterfaces und für die Integration einer weiteren RTI wäre lediglich eine vom *GenericFederateAmbassador* abgeleitete Klasse zu erstellen.

Als Proof of Concept, dass das entwickelte Framework wie gewünscht funktioniert, wurden drei verschiedene Simulatoren und ein modellbasiertes Entwicklungswerkzeug in eine HLA-Simulation integriert (siehe Abbildung 3.8), wobei ein Wechsel der RTI von der pRTI Evolved zu CERTI mit der Bearbeitung nur eines Parameters in einer Konfigurationsdatei (und anschließendem Neustart der Simulation) ermöglicht wurde.

Als Entwicklungswerkzeug wurde MATLAB / Simulink verwendet, welches mit der ersten Option genutzt wurde. Auf diese Weise müssen nur die entsprechenden C/C++-Funktionen aufgerufen werden, was das Entwicklungswerkzeug nativ unterstützt. So bleiben die Vorteile von Simulink (z. B. Onlineanalyse von Blöcken und Schaltübergängen) während einer

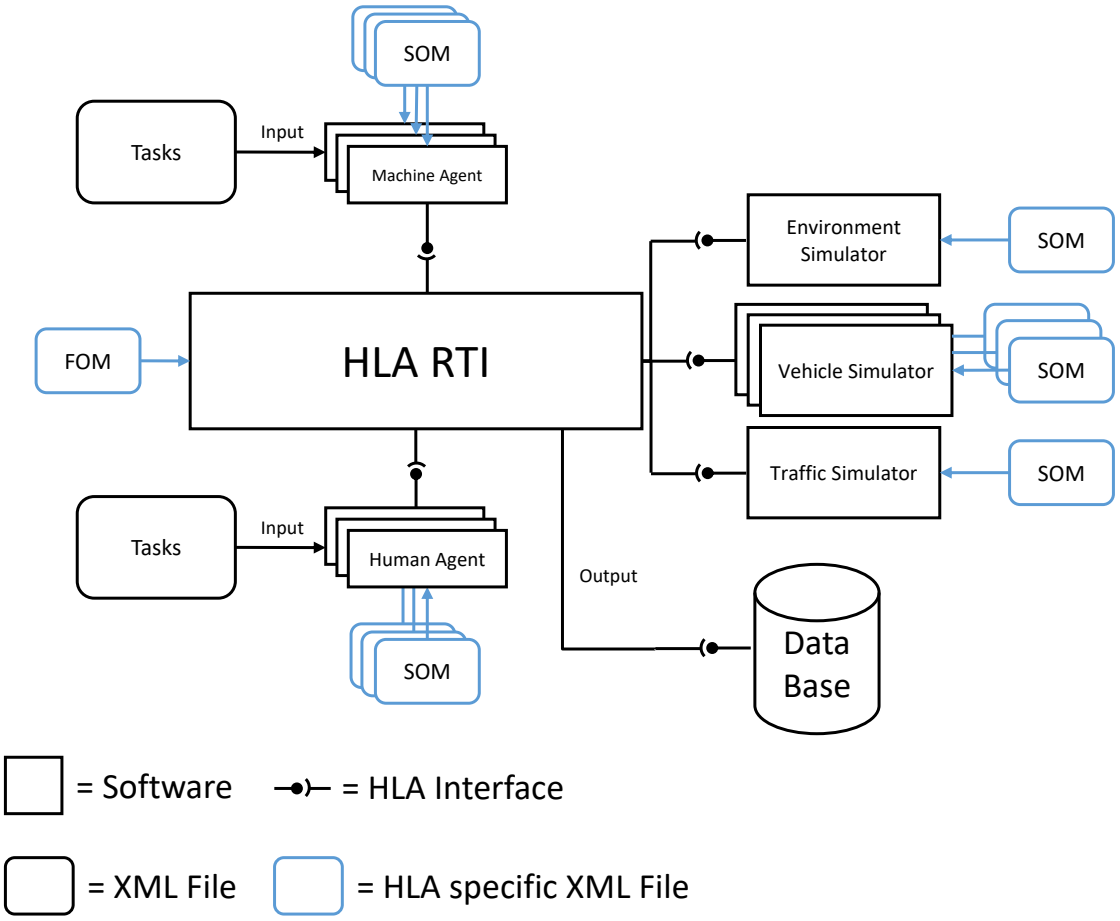

Abbildung 3.8: Generelle HLA-Plattform bei Benutzung des entwickelten Frameworks zur Wiederverwendung einer HLA-Implementierung [Puc+12a].

HLA-Simulation erhalten und es sind keine zusätzlichen Add-Ons wie beispielsweise MATLAB Coder oder RealTime Workshop für eine Generierung von Maschinencode notwendig. Dieser Föderat steht in Abbildung 3.8 stellvertretend für eine konkrete Instanz eines *Machine Agent*. Als Realzeitsimulatoren, stellvertretend für die Instanzen auf der rechten Seite in Abbildung 3.8, wurden die Fahrsimulationssoftware SILAB [WIV19] und der Flugsimulator [Lam] angebunden. Während für SILAB die zweite Option zum Einsatz kam, wurde X-Plane wiederum mit der ersten Option des Datenaustausches umgesetzt. Schließlich ist mit der kognitiven Architektur CASCaS (vgl. Unterabschnitt 3.1.2 auf Seite 37) eine Instanz für den menschlichen Agenten (*Human Agent* in Abbildung 3.8) in den Kontext einer HLA-Simulation integriert worden, die je nach globaler Situation entweder als Autofahrer oder Pilot genutzt werden kann. Die Kopplung von CASCaS mit der HLA wurde ebenfalls mit der ersten Option realisiert.

Evaluation der technischen Umsetzung

Für die Evaluation des Frameworks wurden zwei verschiedene Ansätze verfolgt. Zuerst wurde getestet, wie einfach andere Simulatoren mit dem HLAWrapper gekoppelt werden können und zweitens wurde ein Leistungsvergleich zwischen der CERTI RTI und der Pitch pRTI durchgeführt.

Für den ersten Test wurde ein hauptamtlicher Entwickler[2] von CASCaS, der nicht mit der HLA selbst, aber mit der Schnittstelle des Datenpools vertraut ist, gebeten, das Framework zu benutzen, um CASCaS, X-Plane und MATLAB mit CERTI zu koppeln. Es stellte sich heraus, dass die Implementierung der Kopplung verhältnismäßig einfach war und für jede Komponente ungefähr einen Arbeitstag am Implementierungsarbeit erforderte. Das Hauptproblem bestand darin

a) zu verstehen, welche Möglichkeiten ein Simulator für eine Kopplung bereitstellt (z. B. den Plugin Mechanismus von X-Plane oder die Mex-Schnittstelle von MATLAB),

b) die Testmodelle einzurichten, einschließlich des Schreibens der notwendigen Konfigurationsdateien (SOM und FOM) und

c) die Laufzeitumgebung der RTI einzurichten, d. h. alle Laufzeitbibliotheken an die passenden Stellen zu kopieren, an denen sie auch gefunden werden können, wenn die Simulatoren intern den normalen systemweiten Suchpfad ändern.

Ein erster Leistungsvergleich beider RTI Implementierungen wurde – u. a. als Vorbereitung zur Integration von CERTI in das Framework – in der Masterarbeit von Läsche [Läs11] durchgeführt. Ein Ziel seiner Arbeit war es, die Leistung der CERTI RTI Implementierung im Vergleich mit der Pitch pRTI, unter Berücksichtigung der Übertragungszeiten von Nachrichten, zu evaluieren. Für einen zweiten Leistungsvergleich beider RTI Implementierungen auf Basis des Frameworks wurden im Anschluss zwei Föderaten (ein Nachrichten-Sender und ein Nachrichten-Bouncer) implementiert. In diesem Zusammenhang konnte auch das Problem beseitigt werden, dass CERTI und Pitch pRTI unterschiedliche Interpretationen des IEEE 1516-2000 Standards verwendeten. Die Dauer einer Nachrichtenübertragung wurde wie in Abbildung 3.9 dargestellt gemessen.

„Föderat A“ sendet 10.000 Nachrichten mit typischer Größe für den angedachten Simulationskontext mit einer bestimmten Frequenz an „Föderat B“ und merkt sich den Zeitstempel zu Beginn der Übertragung jeder Nachricht. Jede Nachricht enthält dabei ein Objekt mit 31 Attributen, mit etwa 194 Bytes pro Objekt.

„Föderat B“ sendet die Nachrichten an „Föderat A“ zurück, der dann für jede Nachricht die Zeitdifferenz zwischen dem Senden und dem Empfangen der Nachricht berechnet. Die

[2] Der Entwickler war an der Implementierung des Frameworks beteiligt.

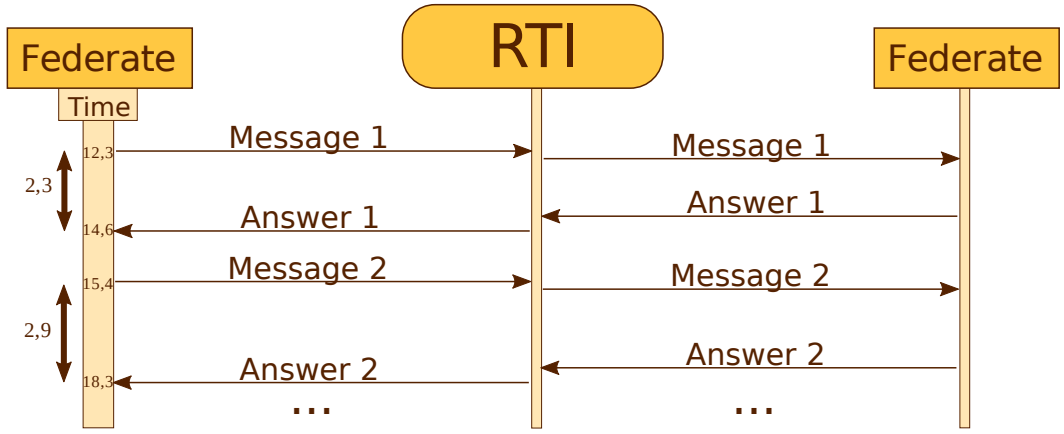

Abbildung 3.9: Schematisches Konzept zur Leistungsmessung einer RTI-Implementierung [Puc+12a].

Periodenlänge, mit der Nachrichten ausgesendet wurden, variiert dabei zwischen 100 ms, 50 ms, 25 ms und 12 ms. Darüber hinaus wurde die Messung durchgeführt mit

a) beiden Föderaten und der RTI auf demselben Computer, und

b) „Föderat A“ sowie der RTI auf demselben Computer und „Föderat B“ auf einem entfernten Computer, welcher mit einem lokalen 100 MBit/s Netzwerk verbunden war.

Beide Computer waren mit einem Intel Core2Duo Prozessor (2,33 GHz / 3 GHz) sowie 4 GB Arbeitsspeicher ausgestattet und als Betriebssystem kam ein 32-Bit Windows XP mit Service Pack 3 zum Einsatz. Für jede Verteilung der Föderaten a) und b) wurden beide RTI-Implementierungen (CERTI und Pitch pRTI) mit den vier zuvor genannten Perioden gemessen, siehe dazu Abbildung 3.10.

Für die Pitch-Implementierung kann festgestellt werden, dass es keinen großen Unterschied der durchschnittlichen Dauer pro Nachricht zwischen der Periodenlänge von 100 ms, 50 ms, 25 ms und 12 ms gibt, egal ob die gesamte Föderation auf einem einzelnen Computer oder im Zwei-Computer-Setup durchgeführt wurde. CERTI hingegen ist im Zwei-Computer-Setup etwa 2,5 ms - 3 ms schneller. Eine mögliche Ursache dafür könnte die interne Struktur von CERTI sein, da sie in einen RTIa- und RTIg-Prozess unterteilt wird, was einen gewissen Kommunikationsaufwand verursacht, welcher das Setup des einzelnen Computers an seine Grenzen bringt. Diese Hypothese wird durch die Messung von 12 ms unterstützt, die auf dem Single- und dem Zwei-Computer-Setup bei etwa 15 ms nahezu identisch ist. Im direkten Vergleich der beiden RTI-Implementierungen für die drei größeren Periodenlängen ist die Implementierung von CERTI auf dem Single-Computer-Setup etwa 5 ms und auf dem Zwei-Computer-Setup 3 ms langsamer. Bei einer Periodenlänge von 12 ms gibt es keinen signifikanten Unterschied, weder zwischen den RTI-Implementierung noch bei der Variation der Computer-Setups. Letzteres führt zu dem Schluss, dass die Leistungsgrenzen der verschiedenen Computeraufbauten erreicht wurden. Da für eine mit dem Kosimulationsframe-

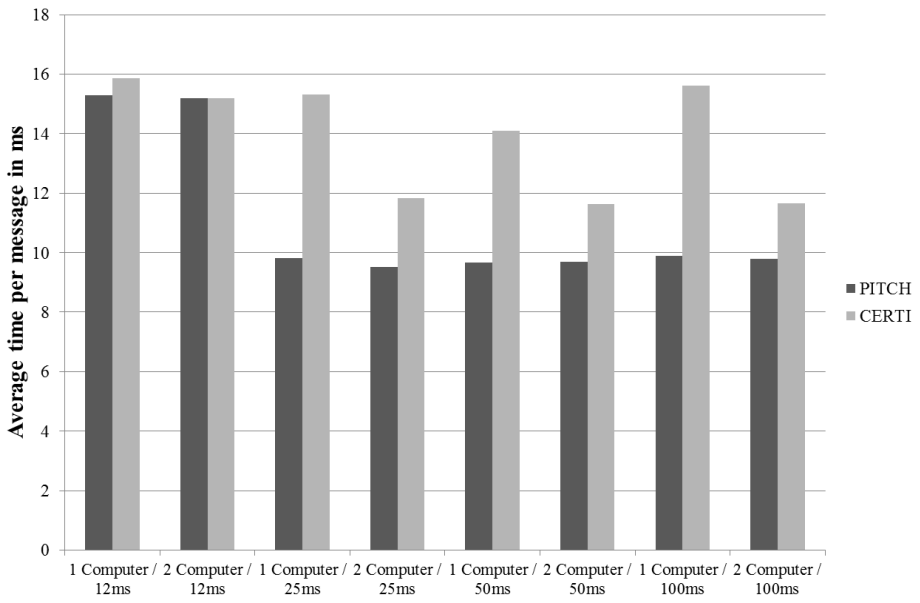

Abbildung 3.10: Vergleich der Leistungsergebnisse für CERTI und Pitch pRTI [Puc+12a].

work zu erstellende Human-in-the-Loop-Simulation unter Verwendung von CASCaS eine Periodenlänge von 50 ms benötigt wird (vgl. Unterabschnitt 3.1.2 auf Seite 38), zeigt die Evaluation, dass beide RTI-Implementierungen schnell genug sind, solange die Föderaten auf verschiedene Computer verteilt werden.

3.3 Batchsimulation zur Bewertung eines Fahrerassistenzsystems

Nachdem im letzten Abschnitt die Technische Umsetzung einer Kosimulation auf Basis der HLA beschrieben und getestet wurde, erfolgt in diesem Abschnitt der nächste Schritt, indem eine Instanz des Kosimulationsframeworks in einem praxisrelevanten Szenario der Fahrerassistenzsystementwicklung angewendet wird.

Mit einem Simulationsaufbau, bestehend aus probabilistischem Fahrermodell, Fahrsimulationssoftware und Fahrerassistenzsystem, sollen in einem Fahrszenario kritische Situationen mittels Onlinemonitoring identifiziert werden, in denen ein FAS den Fahrer unterstützen soll. Der Prozess folgt einem iterativen Ansatz aus Batchsimulationen, die hinsichtlich kritischer Situationen überwacht werden, und einer anschließenden Eingrenzung des großen Zustandsraums auf die im Batchlauf identifizierten Situationen.

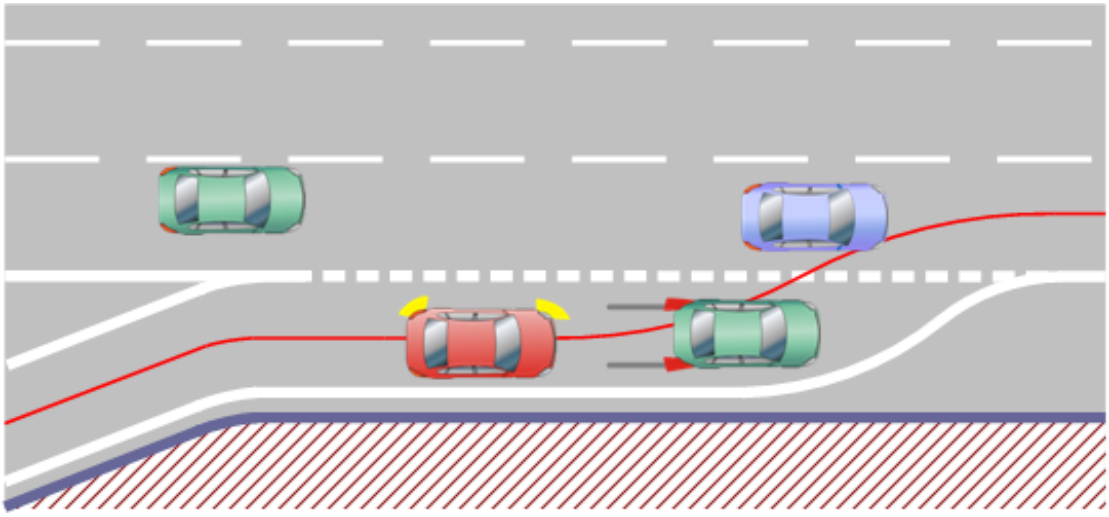

Abbildung 3.11: Anwendungsszenario: „Auffahren auf die Autobahn“, mit Abstand $dist$ zwischen grünem und blauem Fahrzeug, sowie Geschwindigkeitsdifferenz v_{diff} zwischen rotem und rückwärtigem grünen Fahrzeug [Frä+11].

Im Rahmen eines Batchlaufs von Simulationen wird zunächst grob der Verhaltensraum des probabilistischen Fahrermodells evaluiert, indem systematisch externe Parameter des Szenarios (z. B. unterschiedliche Konstellationen des Umgebungsverkehrs) ausprobiert und hinsichtlich kritischer Simulationen analysiert werden. Die Analyse erfolgt durch eine Online-Monitoringfunktion, welche jedem Simulationslauf einen numerischen Kritikalitätswert zuordnet. Nachdem der Zustandsraum grob mit einem Batch an Simulationen abgedeckt wurde, werden Bereiche mit hohen Kritikalitätswerten – unter Rückgriff auf die zugehörigen externen Parameter – einer genaueren Analyse (in einem weiteren Batch) unterzogen. Auf diese Weise können Simulationen, basierend auf vorherigen Beobachtungen in unterschiedlichen Batch-Simulationen, zu Situationen innerhalb eines Fahrszenarios gebracht werden, die hinsichtlich ihrer tatsächlichen Wahrscheinlichkeit seltener sind, aber ein hohes Unfallrisiko beinhalten. Anschließend können identifizierte Situation mit viel höherer Genauigkeit analysiert werden.

In Abbildung 3.11 ist das beispielhafte Anwendungsszenario dargestellt, bei dem ein *Advanced Driver Assistance System (ADAS)* einen Autofahrer beim Auffahren auf die Autobahn unterstützen soll, indem es beim Suchen einer geeigneten Lücke und der dazu notwendigen Geschwindigkeitsanpassung assistiert. Aus Sicht des Fahrermodells in dem roten Fahrzeug sind bei diesem Manöver, neben dem vorausfahrenden grünen Fahrzeug auf dem Beschleunigungsstreifen, auch die Verkehrsteilnehmer auf der rechten Fahrspur der Autobahn zu berücksichtigen. Dieses Szenario umfasst eines der kritischsten Manöver auf Autobahnen. Im Vergleich zu potentiell anderen kritischen Verkehrssituationen (z. B. an Kreuzungen) ist das

Szenario hinsichtlich seiner Variabilität stärker limitiert und daher für die Entwicklung und Beurteilung des Ansatzes geeignet. Als externe Variablen wurden die Anzahl der anderen Verkehrsteilnehmer, Lückengrößen (*dist*) und Geschwindigkeitsunterschiede (v_{diff}) berücksichtigt.

3.3.1 Kosimulationsaufbau

Für den technischen Aufbau der Kosimulation wurden die kommerzielle Fahrsimulationssoftware ST Software [STS11], ein kognitives Fahrermodell auf der Basis von CASCaS [Web+09], ein ADAS zur Unterstützung beim Spurwechsel in Form eines MATLAB Modells, ein Observer für das Onlinemonitoring [Gez09] und ein Rekorder zur Datenaufzeichnung mittels der HLA zu einer Föderation gekoppelt. Ein schematischer Aufbau kann der Abbildung 3.12 entnommen werden. Um die Variabilität zwischen unterschiedlichen Simulationsläufen mit den gleichen Parametern zu reduzieren, d. h. um ein hohes Maß an Reproduzierbarkeit zu erreichen, wurden die Zeitmanagementdienste der HLA benutzt, um die Simulatoren gegenseitig zu synchronisieren. Das Zeitmanagement stellt sicher, dass der Datenaustausch in Übereinstimmung mit dem Fortschritt der logischen Simulationszeit erfolgt, im Gegensatz zu sogenannten *best-effort* Simulationen, bei denen die Daten genau dann während einer Simulation verteilt werden, wenn sie im Simulator entstehen. Die Fahrsimulationssoftware fungiert bei der Kosimulation als Taktgeber in Realzeit, was innerhalb einer Föderation als *time regulating* bezeichnet wird. Damit beeinflusst der Realzeit-Simulator den Zeitfortschritt der anderen Simulationsteilnehmer, welche ihrerseits dann als *time constrained* (durch andere eingeschränkt) bezeichnet werden.

Da die Fahrsimulationssoftware ST Software, in Abbildung 3.12 auf der rechten Seite zu sehen, technisch bedingt aus vielen Einzelmodulen besteht, musste zur Integration des Simulators ein zusätzlicher Wrapper implementiert werden, welche die relevanten Daten aus den unterschiedlichen Modulen während der Simulation zusammenstellt und ebenso ein Feedback von außerhalb zurückspielen kann. Auf der linken Seite in Abbildung 3.12 sind das ADAS und das Fahrermodell dargestellt, sowie der Observer für das Onlinemonitoring (vgl. Abbildung 3.4) und ein Rekorder zum Aufzeichnen der Simulationsdaten. Mit Ausnahme des probabilistischen Fahrermodells hängen bei allen Modellen die Ausgaben deterministisch von den Eingangsgrößen der Umgebungssimulation ab, wobei die Verkehrssimulation (*Trafficsim.*) durch Skripte hinsichtlich der Anzahl und des Verhaltens des Umgebungsverkehrs parametrierbar ist.

Die Kosimulation wurde auf einem Cluster von 6 Standard-PCs durchgeführt. Jeder Föderat verfügte über ein diskretes Zeitschrittmodell zwischen 20 und 35 Hz, wobei die Synchronisation einschließlich Datenaustausch innerhalb der Föderation von der RTI überwacht und gesteuert wurden. Ein kompletter Lauf des Szenarios bestand im Durchschnitt aus ca.

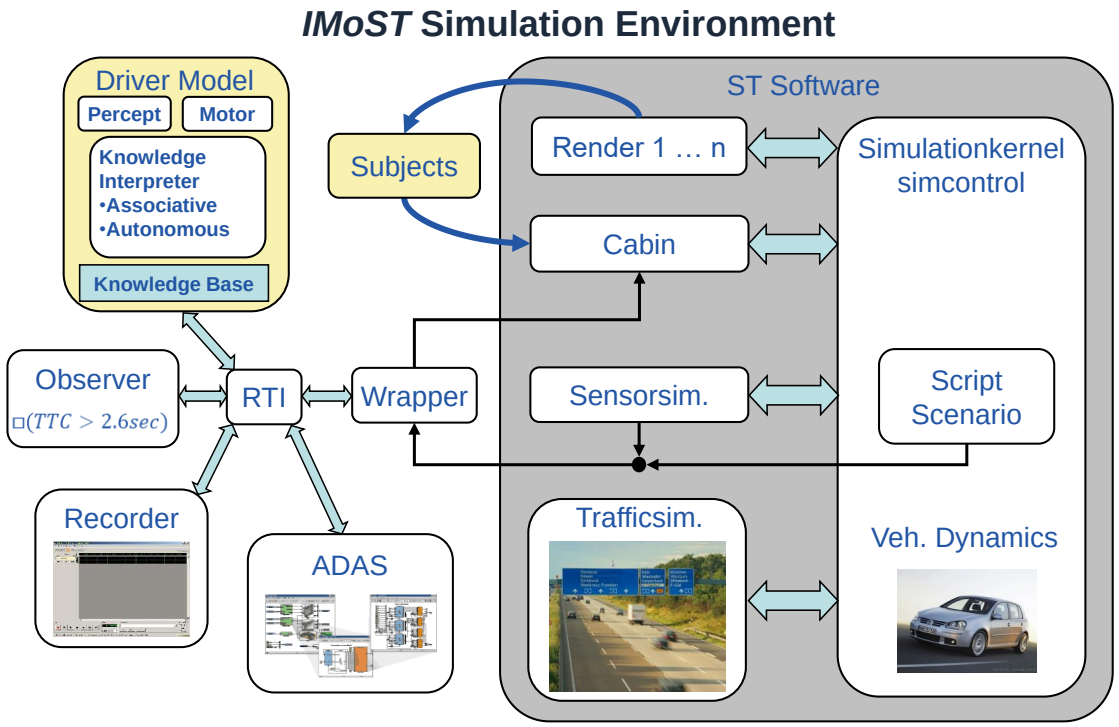

Abbildung 3.12: Schematischer Simulationsaufbau bestehend aus ST Software, Fahrermodell, ADAS, Observer und Rekorder, angelehnt an [Frä+10].

2700 diskreten Zeitschritten. Das Fahrermodell beinhaltet verschiedene Arten von Verhaltensschwankungen, z. B. in Bezug auf den Beschleunigungsstil (sportlich versus entspannt), bei Blickstrategien (Variationen der Blickdauer und -häufigkeit) und hinsichtlich Sicherheitsmargen (bevorzugter Abstand zum vorausfahrenden Fahrzeug bzw. einem Folgefahrzeug), welche durch eine Reihe von Studien mit Probanden ermittelt wurden. Während der Simulation kann das Fahrermodell probabilistisch zwischen diesen unterschiedlichen Verhaltensweisen auswählen. Diese probabilistischen Prozesse sind von besonderem Interesse, wenn im folgenden Kapitel 4 die Entwicklung einer geführten Simulation beschrieben wird, bei der sie durch eine systematische Variation der möglichen Verhaltensweisen ersetzt werden.

3.3.2 Spezifikation von Analyseeigenschaften

Um die Evaluation von Sicherheits- und Funktionsaspekten der Simulation zu automatisieren, wurden Monitore verwendet, die beobachteten, inwieweit Eigenschaften von Interessen erfüllt oder verletzt wurden. Die Eigenschaften wurden in einer linear-temporal-logischen Sprache in Formeln spezifiziert [Gez09] und in die Simulation als Observer integriert. Die

Atome der Formeln beziehen sich auf Eigenschaften des Ego-Fahrzeugs wie die aktuelle Position und Geschwindigkeit oder sichtbare Aktionen des Fahrers (beispielsweise die Veränderung des Lenkwinkels). Typische Sicherheitseigenschaften sind die Zeit bis zum Aufprall (*Time To Collision (TTC)*) und die Zeitlücke zum vorausfahrenden Fahrzeug (*Time HeadWay (THW)*). Die üblichen Zeitoperatoren „immer, schließlich, bis“ (*always, eventually, until*) erlauben es, zeitliche Zusammenhänge des Auftretens von Eigenschaften auszudrücken und somit spezifische Anforderungen für unterschiedliche Simulationsphasen. So lässt sich zum Beispiel ausdrücken, dass „zu keinem Zeitpunkt in einer Simulation die Zeit bis zu einer Kollision mit einem vorausfahrenden Auto auf der gleichen Fahrspur unter einen bestimmten Wert fällt“. Mit einer geeigneten Subformel „TimeToCollision“, welche aus der Distanz und der relativen Geschwindigkeit eines (existierenden) vorausfahrenden Fahrzeugs berechnet wird, kann die Formel

$$\Box(TimeToCollision > 2.6\,sec) \tag{3.1}$$

spezifiziert werden, die besagt, dass der Wert der Subformel niemals unter 2,6 Sekunden fällt. Nach Hülsbusch in [Hül18, S. 77] zitieren Minderhoud in [MB01] und Vogel in [Vog03] „mehrere Studien, die als kritische Grenze einer Time To Collision Werte zwischen 1,5 s bis 5 s angeben“.

Die Standardinterpretation von temporal-logischen Operatoren liefert für jeden Lauf einen Wahrheitswert. Um jedoch zu entscheiden, ob das Ergebnis eines bestimmten Laufs im Vergleich zu anderen besser oder schlechter ist, wäre es hilfreich, den exakten minimalen Wert des Ergebnisses zu berücksichtigen. Deshalb wurde eine nicht standardisierte, quantitative Semantik [AFS04; Sau+09] verwendet, die jedem Simulationslauf einen numerischen Wert einer Formel zuordnet: Eine positive Zahl bedeutet, dass die Formel erfüllt wurde, und der Ergebniswert gibt den minimalen Sekundenabstand des Simulationslaufes an, der die Formel falsch gemacht hätte (analog bedeuten negative Werte, dass die Formel verletzt wurde). Eine Formel definiert also eine Funktion, die jedem Simulationslauf einen numerischen Wert zuordnet. Der HLAObserver von Gezgin [Gez09] übersetzt solche Formeln in ausführbaren Programmcode, welcher innerhalb einer Kosimulation als Federate ausgeführt werden kann. Nach Abschluss eines Simulationslaufs kann mithilfe der numerischen Bewertung des Observers eine Klassifizierung vorgenommen werden, wie gut bzw. schlecht (im Sinne von kritisch oder unkritisch) das Ergebnis einzustufen ist. Auch die Grenzwerte (Minimun und Maximum) stehen nach Abschluss eines Batchlaufes zur Verfügung. Auf diese Weise lassen sich Konfigurationen, die als nicht relevant angesehen werden, in späteren Batchläufen ausschließen, wodurch Simulationszeit eingespart werden kann. Wie sich mehrere Teilformeln zu einer komplexen Gesamtformel integrieren lassen und welche Atome für das beispielhafte Anwendungsszenario „Auffahren auf die Autobahn“ zur Identifikation kritischer Situationen verwendet wurden, kann [Frä+10; Frä+11] entnommen werden.

3.3.3 Evaluation des Onlinemonitoring

Im Folgenden soll jetzt das Onlinemonitoring anhand der oben beschriebenen Kosimulation evaluiert werden. Da Monte-Carlo-Simulation für die zuverlässige Exploration von Randbereichen (seltenen Ereignissen) ungeeignet ist, wurde für das Anwendungsszenario „Auffahren auf die Autobahn“, wie in Abbildung 3.11 skizziert, folgender Ansatz verfolgt:

- Mit einer temporal-logischen Formel wurde kritisches Verhalten[3] spezifiziert.
- Ein Batch an Simulationen wurde für eine grobe Untersuchung des Verhaltensspektrums durchgeführt. Der Batch liefert ein Raster von Stichproben im Wertebereich des Szenarios für das Modell des Fahrermodells. Durch Interpolation der Stichproben kann eine Approximation gewonnen werden.
- Weitere Batches verfeinern die Approximation in den Bereichen, die hinsichtlich der spezifizierten Formel kritisches Verhalten aufzeigen.

Mit diesem Ansatz konnten maximale und minimale Kritikalitätswerte mit deutlich weniger Simulationsläufen erkannt werden als mit fortwährender naiver Monte-Carlo-Simulation. Analysiert wurden für das Anwendungsszenario die Parameter v_{diff} $[km/h]$ und $dist$ $[m] \in \{20, 30, 40\}$, wobei v_{diff} die Geschwindigkeitsdifferenz zwischen Fahrzeugen auf der rechten Fahrspur und dem Ego-Fahrzeug charakterisiert, und $dist$ die Lückengröße zwischen den Fahrzeugen auf der rechten Fahrspur, in die das Ego-Fahrzeug einfädeln möchte. Das Gesamtergebnis von zwei Batchläufen kann Abbildung 3.13 entnommen werden, wobei der erste Batchlauf im linken Teil der Abbildung die Stichproben und der zweite Batchlauf die Verfeinerung, im rechten Teil der Abbildung, repräsentiert.

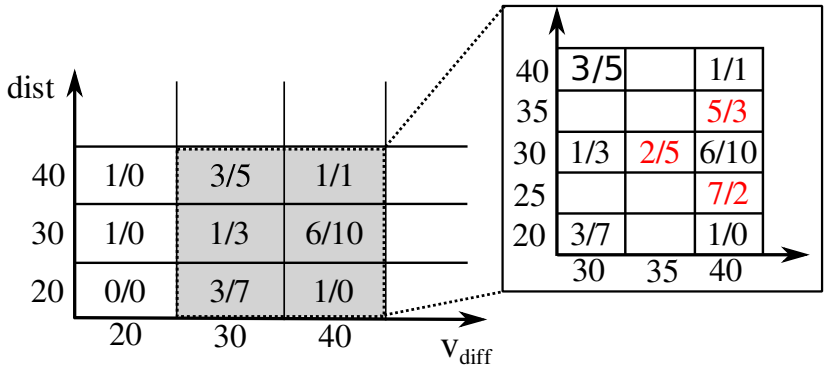

Abbildung 3.13: Exploration kritischer Parameterkombinationen mittels Onlinemonitoring im Anwendungsszenario: „Auffahren auf die Autobahn“ [Frä+10].

[3] Eine Kombination aus TTC und THW.

Die Einträge in der Ergebnismatrix repräsentieren die Anzahl an inakzeptablen und äußerst kritisch Simulationsläufen von jeweils 20 Durchläufen je Kombination. Das schlechteste Ergebnis ist mit 6 inakzeptablen und 10 äußerst kritischen Simulationsläufen bei der Kombination $v_{diff} = 40\,km/h$ und $dist = 30\,m$ ersichtlich. Der Grund für die relativ hohen Kritikalitätswerte ist dem Umstand geschuldet, dass das Szenario in dieser Kombination – in Übereinstimmung mit Vergleichsfahrten von menschlichen Probanden im selben Szenario [Web+09] – äußerst anspruchsvoll ist. Der rechte Teil in Abbildung 3.13 zeigt nach einem zweiten Batchlauf eine Verfeinerung in der Nähe der kritischsten Kombination. Eine noch bessere Exploration kann allerdings erst erreicht werden, wenn die nach wie vor rein zufällig bestimmten Entscheidungen innerhalb eines Batches durch eine systematische Untersuchung der Wahrscheinlichkeiten ersetzt werden, was – anders als bei Blackbox-Modellen – bei Whitebox-Modellen, wie dem verwendeten Fahrermodell, durchaus praktikabel ist. Wie genau eine solche Ersetzung möglich ist, wird im folgenden Kapitel Kapitel 4 detailliert beschrieben.

3.4 Zusammenfassung

Das vorgestellte Kosimulationsframework bietet in Form des HLAWrappers eine Möglichkeit, Entwicklungswerkzeuge, Tools und Fahrsimulationssoftware, die unter anderem im Rahmen von modellbasierter Entwicklung zu Einsatz kommen, zu einer Kosimulation zusammenzuschließen. Damit wurde ein Weg aufgezeigt, mit dem die Funktionalität eines sicherheitskritischen Fahrerassistenzsystem in einer Kosimulation evaluiert werden kann. Durch die zusätzliche Integration eines Observers konnte mit Batchsimulationen gezeigt werden, dass sich potenziell gefährliche Situationen identifizieren lassen, in denen ein Fahrerassistenzsystem (FAS) zum Einsatz kommen soll. Das anschließende sukzessive Verfeinern des Suchraumes grenzt den komplexen Zustandsraum zunehmend ein und bietet die Chance den Entwicklungszustand eines FAS zu analysieren und ggf. notwendige Verbesserungen abzuleiten. Damit gibt das Kapitel eine Antwort auf die erste der vier Forschungsteilfragen.

Für eine weitere Reduktion notwendiger Simulationszeit sowie eine Ableitung von statistisch ermittelten Sicherheitsgarantien, welche beispielsweise für eine Zertifizierung von Fahrerassistenzsystemen erforderlich sind, müssen aber in einem weiteren Schritt auch die zufallsbehafteten Modellteile des Fahrermodells kontrollier- und nachvollziehbar zum Einsatz gebracht werden. In diesem Zusammenhang lassen sich dann die übrigen drei Forschungsteilfragen beantworten.

Kapitel 4

Geführte Simulation

Nachdem im vorhergehenden Kapitel die technische Basis geschaffen wurde, um die im Rahmen einer modellbasierten Entwicklung verwendeten Modelle und Entwicklungswerkzeuge in einer Kosimulation zu integrieren, und deren angedachte Funktionsweise von Fahrerassistenzsystemen evaluieren zu können, wird in diesem Kapitel zunächst dargelegt, welche Anforderungen Kosimulationsteilnehmer z. B. hinsichtlich probabilistischer Eigenschaften erfüllen müssen, damit sich eine Kosimulation nachvollziehbar steuern bzw. führen lässt, und welche Eigenschaften eine Simulationssteuerung bereitstellen muss. Den Schwerpunkt dieses Kapitels bildet der Threshold Uncertainty Tree Search (TUTS) Algorithmus, welcher basierend auf Adaptive Importance Sampling (AIS), (vgl. Unterabschnitt 2.5.2 auf Seite 22) die in Kapitel 3 vorgestellte Kosimulationsplattform mit dem Ziel führen kann, seltene Ereignisse sichtbar zu machen und eine quantitative Evaluation von deren Auftretenswahrscheinlichkeit zu ermöglichen. Zwei empirische Evaluationen in Fahrszenarien zeigen dabei die Funktionsweise auf, bevor im Rahmen der möglichen Übertragbarkeit einer geführten Simulation auf andere Domänen ein Vergleich mit einem anderen Algorithmus auf Basis von AIS diskutiert wird. Abschließend werden die Grenzen des beschriebenen Ansatzes der geführten Simulation kritisch analysiert. Insgesamt beantwortet das Kapitel die Forschungsteilfragen zwei, drei und vier.

Für ein besseres Verständnis des Lesers hinsichtlich der Unterschiede einer systematischen Exploration von Parametern, wie im vorhergehenden Kapitel bei der Batchsimulation, und einer *geführten Simulation*, sei an dieser Stelle dem weiteren Verlauf ein kurzer Vergleich vorangestellt: Während bei einer systematischen Exploration von Parametern, wie beispielsweise im Anwendungsszenario in Abschnitt 3.3 mit v_{diff} und $dist$ der Umgebungsverkehr variiert werden konnte, die Wertebereiche bekannt sind, adressiert eine geführte Simulation im weiteren Verlauf der Arbeit die Problemstellung, bei der zwar die Klasse von Parametern bekannt ist, nicht jedoch deren zeitliche Abhängigkeit bzw. deren Ausprägung zu einem bestimmten Simulationszustand.

Wesentliche Teile dieses Kapitels wurden in [Puc+12b; Puc+13; PFG18] veröffentlicht.

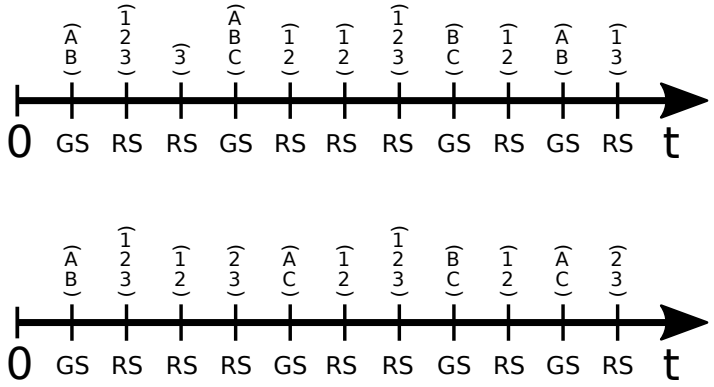

Abbildung 4.1: Exemplarische Parameterklassen GS und RS der kognitiven Architektur CASCaS, dargestellt über die Zeit mit fiktiven Ausprägungen.

Ein Beispiel basierend auf dem Fahrermodell der kognitiven Architektur CASCaS (vgl. Unterabschnitt 3.1.2) wäre folgendes: Ein Parameter des Fahrermodells, welcher zu Beginn eines Simulationslaufs gewählt wird, entscheidet über den Fahrstil oder das Alter des Fahrers. Ist diese Entscheidung einmal getroffen, ändert sie sich für den weiteren zeitlichen Ablauf eines Simulationslaufs nicht. Derartige Parameter lassen sich beispielsweise mittels Gridsuche explorieren.

Ziele und Regeln, welche das Verhalten des Modells während der Laufzeit beschreiben, sind hingegen von der Umgebung des Szenarios abhängig, und lassen sich als Parameterklasse bezeichnen. Sie können sich je nach Kontext hinsichtlich ihrer Auswahlwahrscheinlichkeit ändern, z. B. wenn mehrere Ziele gleichzeitig zur Auswahl stehen, aber auch während des Ablaufs eines Simulationslaufs je nach Umgebungssituation hinsichtlich ihrer Ausprägung unterschiedlich ausgeprägt sein. Die Abbildung 4.1 zeigt als Erläuterung zwei exemplarische Simulationsläufe von CASCaS in einem Simulationsszenario. Auf der x-Achse sind über die Zeit probabilistische Entscheidungen über die Auswahl von Zielen (GS) bzw. Regeln (RS) aufgetragen. Die bei einer Entscheidung verfügbaren Optionen sind auf der y-Achse aufgetragen. Bei einem Vergleich der beiden Abläufe sind die ersten zwei Entscheidungen mit GS (AB) und RS (123) noch identisch. Im weiteren Verlauf unterscheiden sich beide Läufe aber deutlich, sei es hinsichtlich der Abfolge von Entscheidungen bzgl. GS und RS oder deren möglicher Optionen. Beim Start einer Simulation ist die Menge aller Ziele und Regeln zwar bekannt, an einem bestimmten Simulationszeitpunkt stehen aber zumeist nur Teilmengen bereit, welche auch in ihrer Ausprägung deutliche Unterschiede aufweisen können.

Der folgende Abschnitt beschreibt, welche Anforderungen an Simulationskomponenten gestellt werden, damit sie für die Methodik der geführten Simulation im Rahmen dieser Arbeit geeignet sind.

4.1 Anforderungen an Simulationskomponenten

Welche Anforderungen an Simulatoren (Entwicklungsumgebungen, -tools, ausführbare Modelle sind an dieser Stelle synonym gemeint) gestellt werden müssen, um an einer geführten Simulation teilnehmen zu können, lässt sich nicht pauschal beantworten. Je nach Anwendungskontext mag es Spezialfälle geben, die eine Einzellfallbetrachtung erforderlich machen. Es gibt für die Methodik, welche im Rahmen dieser Arbeit beschrieben wird, einige grundlegende Aspekte, die berücksichtigt werden müssen und daher als Anforderung angesehen werden. So ist z. B. die HLA selbst keine Anforderung für eine geführte Simulation, prinzipiell könnte auch eine andere Technik zur Kopplung von Simulationsteilnehmern genutzt werden, da das Kosimulationsframework, wie in Kapitel 3 beschrieben, als Basis benutzt wird, müssen Simulatoren eine Schnittstelle bereitstellen, über die sie mit der High Level Architecture in eine gemeinsame Simulation integriert werden können. Da die HLA ein abstraktes Zeitmodell zur Synchronisation aller Kosimulationsteilnehmer hinsichtlich Datenaustausch und zeitlichem Verlauf benutzt, können Rücksprünge (z. B. für *backtracking*) in der Simulationszeit nur innerhalb eines lokalen Zeitmodells durchgeführt werden und nicht hinsichtlich des übergeordneten abstrakten Zeitmodells einer Kosimulation. Mit dieser Einschränkung müssen Simulatoren umgehen können. Eine weitere Eigenschaft, die bereits in Abschnitt 2.6 auf Seite 26 diskutiert wurde, betrifft die Möglichkeit, Simulatoren zu initialisieren. Insbesondere bei commercial off-the-shelf-Fahrsimulationssoftware gibt es häufig nur definierte Situationen (z. B. zu Beginn eines Szenarios), an denen integrierte Fahrdynamiken oder Umgebungsverkehr initialisiert werden können. Aus diesem Grund wurde der Fokus für diese Arbeit auf Importance Sampling (IS) statt auf Importance Splitting (ISp) gelegt. Soll als Technik zur Simulationsoptimierung ISp zum Einsatz kommen, so muss als Anforderung an einen Simulator die Aufsetzbarkeit an (nahezu) jeder Stelle möglich sein, eine Einschätzung die auch von Gollücke in [Gol16, S. 64 Anforderung [A_S4]] benannt wird. Eine etwas differenziertere Betrachtung erfordert die Eigenschaft zufallsabhängiger Prozesse innerhalb von Simulatoren. Welche Anforderung diesbezüglich für die Umsetzung einer geführten Simulation gestellt wird, diskutieren die folgenden Unterabschnitte.

Handhabung probabilistischer Elemente

Probabilistische Elemente sind aus Sicht eines Entwicklers ein zweischneidiges Schwert. Auf der einen Seite sind sie ein probates Mittel, um spontane oder unvorhersehbare Ereignisse, wie sie auch im realen Leben vorkommen, in einer Simulation abzubilden. Auf der anderen Seite ist es aber hinsichtlich einer Nachvollziehbarkeit bzw. Reproduzierbarkeit von Ergebnissen einer Simulation immer zusätzlicher Aufwand, wenn Simulationsmodelle von vielen Zufallsprozessen Gebrauch machen. Soll beispielsweise nach einem Batch von Simulationen eine Aussage über die Auftretenswahrscheinlichkeit von bestimmten Ereignissen

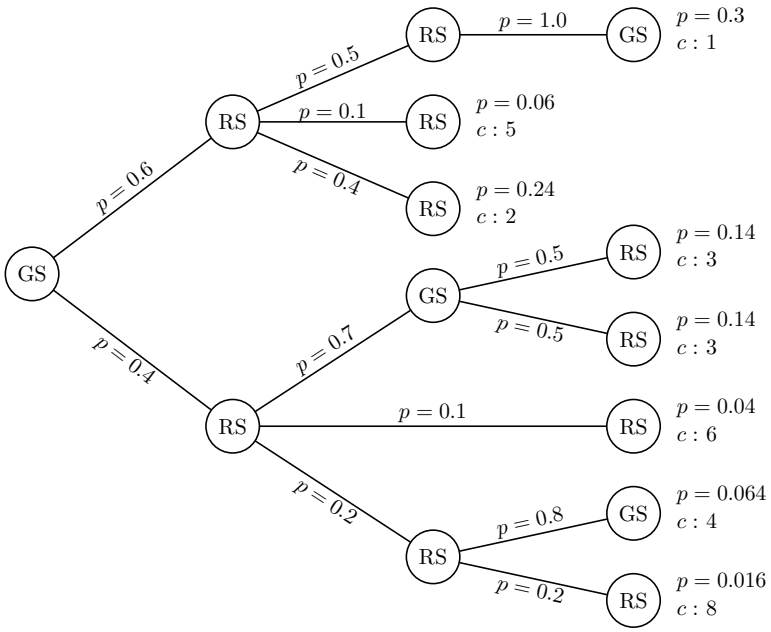

Abbildung 4.2: Beispielhafter Ereignisbaum von CAScAS in einem Simulationsszenario mit Ziel- bzw. Regelauswahl, den Wahrscheinlichkeiten p der jeweiligen Option und der Kritikalität c eines Pfades.

getroffen werden, so ist zwangsweise die Kenntnis aller Wahrscheinlichkeiten notwendig, mit denen die Zufallsereignisse bis zum beobachteten Ergebnis erzeugt worden sind. Als Beispiel sei erneut ein Fahrermodell der kognitiven Architektur CASCaS genannt, bei dem während einer Simulation zu diversen Zeitpunkten zwischen verschiedenen Vorgehensweisen mit Hilfe eines Zufallsgenerators eine Auswahl getroffen wird, welche Handlung als nächstes ausgeführt werden soll.

Der in Abbildung 4.2 beispielhaft dargestellte Ereignisbaum soll bei der Veranschaulichung helfen. Sei der Wurzelknoten des Baums – hier eine Zielauswahl (GS) – die erste Zufallsentscheidung des Fahrermodells während der Simulation, dann kennzeichnen die abgehenden Äste die Anzahl an verfügbaren Optionen. Die jeweilige Eintrittswahrscheinlichkeit p der Optionen addieren sich in der Summe zu 1 zusammen. Jeder nachfolgende Knoten steht für eine weitere Zufallsentscheidung während der Simulation, wobei trotz identischer Anzahl an Optionen nicht direkt auf deren Ausprägung geschlossen werden kann. Das wird deutlich, wenn die Grafik in Bezug zu Abbildung 4.1 gesetzt wird. Entsprechen beispielsweise von der Wurzel aus gesehen die beiden äußersten Äste dem zeitlichen Verlauf der ersten drei Zufallsentscheidungen von Abbildung 4.1, so ist ersichtlich, dass zwar die Abfolge der probabilistischen Elemente mit GS, RS, RS, etc. identisch ist, die Wahrscheinlichkeiten bei der Regelauswahl (RS) in der ersten Baumebene aber bereits verschieden sind und die RS in der zweiten Baumebene nur bedingt gleiche Regeln zur Auswahl vorgibt.

Konzeptionell ergibt sich also für das Fahrermodell in einem Simulationsszenario ein Ereignisbaum von möglichen Abläufen, wobei jeder Pfad von der Wurzel zu einem Blatt einen möglichen, zeitbehafteten Ablauf des Fahrermodells repräsentiert. Die Verzweigungswahrscheinlichkeiten der einzelnen Knoten entlang eines Pfades können zur Erfassung der Wahrscheinlichkeit eines Ereignisses während der Simulation aufmultipliziert werden. Allerdings kann der Ereignisbaum nicht a priori aus der Definition eines Fahrermodells abgeleitet werden, sondern er ergibt sich erst in Kombination mit der Umgebung während der Simulationsausführung innerhalb eines Szenarios, so dass sich seine genaue Ausprägung erst in einem iterativen Simulationsablauf durch Exploration ergibt.

Bei probabilistischen Blackbox-Modellen, bei denen lediglich die Eingangsparameter gesetzt und die Ausgabeparameter beobachtet werden können, lässt sich auf diese Weise folglich kein Ereignisbaum erstellen. Dem gegenüber steht die Problematik, dass dem Entwickler eines Fahrerassistenzsystems nicht unbedingt eine Whitebox-Sicht auf das Fahrermodell zur Verfügung steht, da dessen OEM das eingeflossene technologische Know-how vor Konkurrenz (als Blackbox) schützt. Da für die Methodik der vorliegenden Arbeit die Nachvollziehbarkeit modellinhärenter Wahrscheinlichkeitsverteilungen zwecks Analyse und quantitativer Beurteilung unverzichtbar ist, leiten sich direkt folgende Anforderungen ab:

Zufallsbehaftete Modelle müssen an einer Schnittstelle

1. die Wahrscheinlichkeiten ihrer probabilistischen Elemente bei einer Zufallsentscheidung zwecks Nachverfolgbarkeit offenlegen und
2. eine externe Vorgabe der Zufallsentscheidung ermöglichen.

Damit soll sichergestellt werden, dass Methoden zur Simulationsoptimierung wie IS und ISp einerseits die für eine quantitative Analyse notwendigen Daten erhalten und andererseits eine Eingriffsmöglichkeit bei Zufallsentscheidungen haben.

Anforderungen an ein Guiding-Modell

Das Guiding-Modell, im weiteren Verlauf der Arbeit synonym mit *Simulationguide* verwendet, übernimmt die intelligente Führung einer Simulation im Hinblick auf ein definiertes Ziel, wobei die notwendige Intelligenz vom jeweiligen Kontext abhängt. Der Anwendungskontext dieser Arbeit liegt auf der Simulation seltener Ereignisse bei der Fahrerassistenzsystementwicklung in einer Kosimulation. Wie in Abbildung 4.3 dargestellt, bedient das Guiding-Modell die Schnittstelle, an denen probabilistische Modelle die inhärenten Zufallsentscheidungen mit ihren möglichen Optionen und Wahrscheinlichkeitsverteilungen zwecks Nachverfolgbarkeit zugänglich machen und eine definierte Auswahl ermöglichen. Auf welche Weise das Guiding-Modell diese Informationen zur Simulationsführung nutzt, bleibt hingegen der jeweiligen Instanziierung vorbehalten.

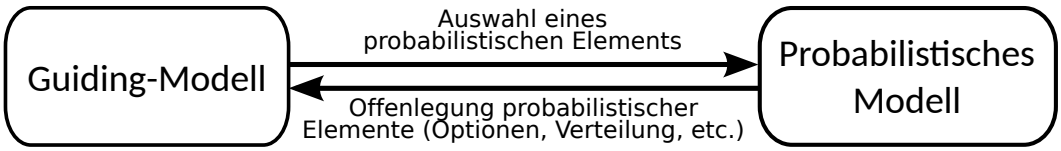

Abbildung 4.3: Abstrakte Darstellung der Schnittstelle zwischen Guiding-Modell und einem Modell mit probabilistischen Eigenschaften.

Welche „Intelligenz“ für eine Simulationssteuerung notwendig ist, hängt vom jeweiligen Kontext ab. Der Anwendungskontext dieser Arbeit (zur Erinnerung) liegt auf der Simulation seltener Ereignisse bei der Fahrerassistenzsystementwicklung in einer Kosimulation, sodass die Ereignisse zumeist kritischer Natur aus Sicht eines Autofahrers sind. Da eine Simulationssteuerung selbst Bestandteil einer Kosimulation ist, gelten für sie zunächst die selben Anforderungen wie für Simulatoren (vergleiche Abschnitt 4.1). Zusätzlich muss die Schnittstelle bedient werden, welche die Zufallsentscheidungen probabilistischer Modelle verfügbar macht, so dass dies als eine weitere Anforderung angesehen werden kann (siehe vorhergehender Unterabschnitt „Handhabung probabilistischer Elemente)“. Im folgenden Unterabschnitt wird eine prototypische Umsetzung in Form eines Algorithmus beschrieben.

4.2 TUTS-Algorithmus

Das Ziel des Threshold Uncertainty Tree Search (TUTS) Algorithmus ist es, eine Simulation basierend auf Sampling (rein zufälligen Simulationsläufen) so zu führen, dass kritische (seltene) Ereignisse häufiger sichtbar werden. Dazu ist es notwendig, die im letzten Abschnitt beschriebenen zufallsbehafteten Prozesse eines Modells systematisch zu explorieren. Der TUTS-Algorithmus kann als eine Implementierung des Guiding-Modells angesehen werden. Um einen passenden Algorithmus herzuleiten, muss zunächst das Problem gelöst werden, dass probabilistische Elemente in zwei Klassen unterschieden werden können:

1. Diskrete probabilistische Elemente (discrete probabilistic element, DPE)
2. Kontinuierliche probabilistische Elemente (continuous probabilistic element, CPE)

Ein DPE e_d einer bestimmten Verteilung kann dabei einem endlichen Satz von Optionen zugeordnet werden, bezeichnet als O_{e_d}. Ein CPE e_c ist hingegen einer kontinuierlichen Wahrscheinlichkeitsdichtefunktion (probability density function (PDF)) zugeordnet, bezeichnet mit f_{e_c}. Um das Problem einer unendlichen Menge von Optionen bei einem CPE etwas einzuschränken (bei einer Baumdarstellung wären hier unendlich viele Zweige notwendig),

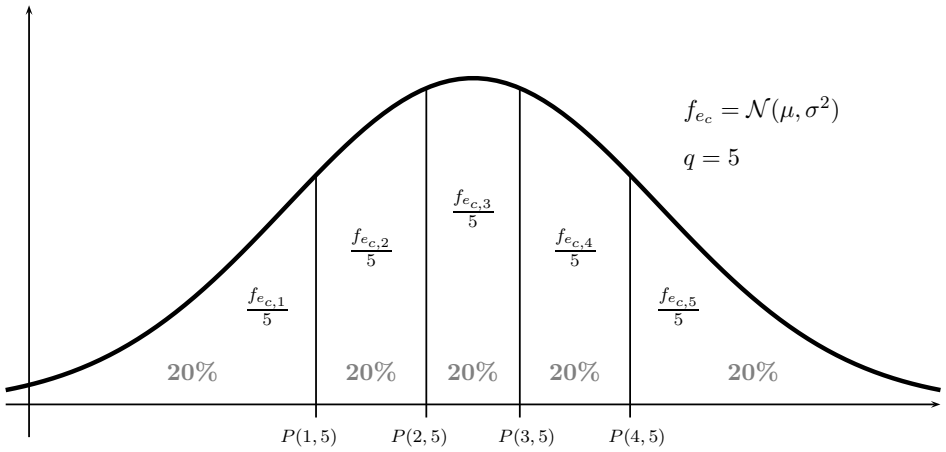

Abbildung 4.4: Beispielhafte Diskretisierung einer Wahrscheinlichkeitsdichtefunktion für eine kontinuierliche Normalverteilung in fünf Segmente mit gleicher Stichprobenwahrscheinlichkeit [Puc+12b].

werden CPEs zunächst diskretisiert, so dass zunächst nur ein rein diskretes Entscheidungsszenario zu betrachten ist. Dazu wird für jedes CPE e_c ein DPE e_d mit zugehörigem Satz O_{e_d} abgeleitet, indem f_{e_c} in einen Satz von q PDFs mit gleicher Stichprobenwahrscheinlichkeit aufgeteilt wird. Erreichen lässt sich diese Aufteilung durch eine Zerlegung der originalen PDF f_{e_c} in äquidistante Perzentile.

Abbildung 4.4 illustriert das Verfahren der Diskretisierung anhand einer Normalverteilung $\mathcal{N}(\mu, \sigma^2)$ mit Erwartungswert μ und Varianz σ^2 in $q = 5$ Perzentile. Die Wahrscheinlichkeit jeder einzelnen Option beträgt dabei $1/q$. Die zu jedem Perzentil gehörenden PDFs lassen sich wie folgt definieren: Sei $P(i,q)$ das $\frac{i \cdot 100}{q}$-te Perzentil von f_{e_c} mit $i \in \{1, \ldots, q\}$, dann ist die i-te PDF definiert als

$$f_{e_c,i}(x) = \begin{cases} q \cdot f_{e_c}(x) & \text{wenn } P(i-1,q) \leq x < P(i,q) \\ 0 & \text{sonst.} \end{cases} \tag{4.1}$$

$f_{e_c,i}(x)$ definiert damit die Höhe der Wahrscheinlichkeit zu gegebenem x in einem Perzentil i, wobei die Gesamtwahrscheinlichkeit eines Perzentils bei 100% liegt.

4.2.1 Identifizierung von kritischen Verhaltensweisen

Eine Grundvoraussetzung bei der Simulation kritischer Ereignisse betrifft deren Identifizierung. Sei S die Menge aller erreichbarer Zustände in einer Simulation, $S_0 \subseteq S$ die Menge der Initialzustände, welche eine benutzerdefinierte Startbedingung erfüllen, sowie $S_t \subseteq S$ die Menge aller Endzustände, welche eine benutzerdefinierte Abbruchbedingung erfüllen. Jeder

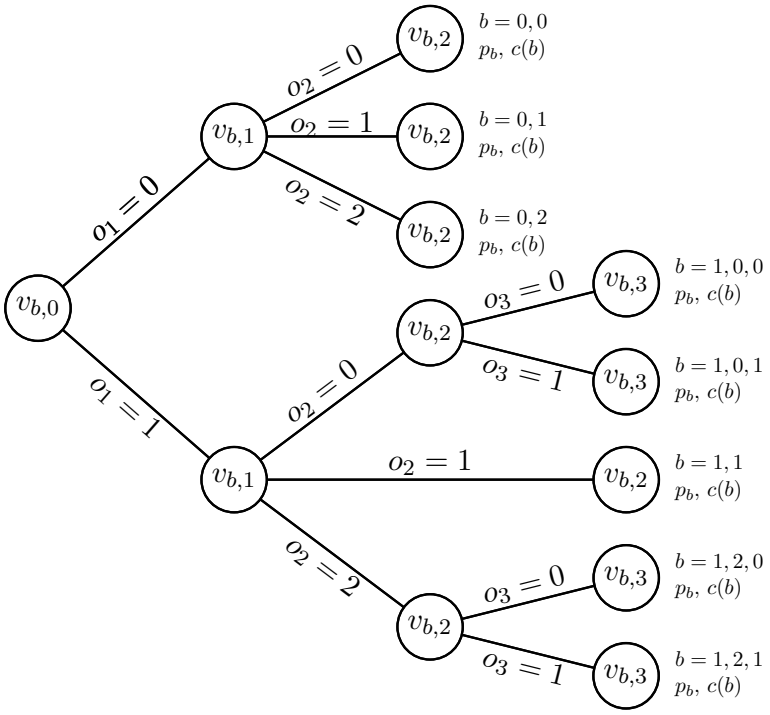

Abbildung 4.5: Beispielhafte Darstellung von Zufallsereignissen und kritischem Verhalten während einer Simulation als Ereignisbaum.

Simulationslauf startet in einem Initialzustand und stoppt, sobald ein Endzustand erreicht ist. Es wird davon ausgegangen, dass mittels einer geeigneten Abbruchbedingung sichergestellt ist, dass jeder Simulationslauf schließlich einen Endzustand erreicht. Dann definiert eine Funktion $c(b)$ zu einem Verhalten b (*behaviour*) eines einzelnen Simulationslaufs einen numerischen Wert als Bewertung seiner Kritikalität. Hohe Werte der Funktion $c(b)$ zeigen kritisches Verhalten von b. Unter der Annahme, dass eine Startbedingung für alle Simulationsläufe identisch ist und dass ein Algorithmus in der Lage ist, jedes probabilistische Element zu kontrollieren, kann jedes Verhalten b als Sequenz probabilistischer Ereignisse $b = o_1, o_2, \ldots, o_{n_b}$ charakterisiert werden, wobei $o_i \in O$ diejenige Option aller verfügbaren (probabilistischen) Optionen beschreibt, die beim i-ten Ereignis gewählt wurde. Die Anzahl an Ereignissen n_b, welche bei einem Simulationslauf bis zum Erreichen eines Endzustandes auftreten können, kann je nach Verhalten unterschiedlich ausfallen.

Zur besseren Lesbarkeit wird der Ansatz anhand eines Ereignisbaums beschrieben, wobei jeder Knoten des Baums – analog zu den bisherigen Darstellungen – einem Simulationszustand entspricht, in dem ein Zufallsereignis bestimmt werden muss. Sei $v_{b,i} \in V$ derjenige Knoten, der nach dem i-ten Ereignis eines Verhaltens b erreicht wurde, dann beschreibt die Funktion $t\colon V \times O \to V$ die Eltern-Kind-Beziehung im Ereignisbaum, siehe Abbildung 4.5. Wird

die gleiche Sequenz von Optionen in verschiedenen Simulationsläufen verwendet, lässt sich ein Verhalten identisch reproduzieren, da die gleiche Abfolge von Zuständen simuliert wird. Ein Algorithmus mit dieser Eigenschaft ermöglicht eine deterministische Simulationsführung, welche mit mehrfachen aufeinanderfolgenden Simulationsläufen einen vollständigen Ereignisbaum eines Modells ableiten kann, wobei jeder einzelne Simulationslauf einen Pfad in diesem Baum repräsentiert. Der linke Teil von Abbildung 4.5 zeigt einen beispielhaften Ereignisbaum. Jedem Blatt im Baum ist ein Verhalten b zugeordnet, welches die Sequenz der Optionen (den Pfad) von der Wurzel beschreibt, die Kritikalität, die das Verhalten $c(b)$ charakterisiert, und eine Wahrscheinlichkeit p_b des Verhaltens, die durch Produktbildung der Wahrscheinlichkeiten aller Optionen des Pfades berechnet werden kann.

Nach jedem Simulationslauf wird die beobachtete Kritikalität rückwärts im Pfad an jedem Knoten annotiert, wobei das Blatt den Startpunkt markiert. Dies erfolgt durch die Abbildung $\tilde{c}\colon V \to \mathbb{R}$. Sei $m_{b,i}$ die Anzahl möglicher Optionen des $(i+1)$-ten Ereignisses innerhalb des Knotens $v_{b,i}$ und v_{b,n_b} der Zustand eines Blattes, dann wird die Kritikalität eines Zustands wie folgt definiert:

$$\begin{aligned} \tilde{c}(v_{b,n_b}) &= c(b) \\ \tilde{c}(v_{b,i}) &= \max_{1 \le j \le m_{b,i}} \tilde{c}(t(v_{b,i}, j)) \end{aligned} \tag{4.2}$$

Auf diese Weise erhält der Endzustand eines Simulationslaufs die gerade ermittelte Kritikalität und an den vorhergehenden Zuständen des Laufes bis zur Wurzel ist, bei nachfolgenden Simulationsläufen, die maximal zu erwartende Kritikalität aus vorherigen Simulationsläufen direkt zugänglich.

Mit der beschriebenen Funktionsweise zur Identifikation kritischer Verhaltensweisen und deren Annotation in einem Ereignisbaum ließe sich eine einfache Strategie zur Simulationsführung ableiten. Für jedes diskrete probabilistische Element e_d werden nacheinander alle Optionen O_{e_d} ausgewählt. Wurden alle Optionen einmal simuliert, wird anschließend die Wahrscheinlichkeit der Optionen mit der zu erwartenden maximalen Kritikalität gewichtet. Folglich neigt die Simulation dazu, Teilbäume des Ereignisbaums zu untersuchen, die zuvor zu kritischen Konsequenzen geführt haben, und vermeidet unkritische Teilbäume. Das Prinzip dieser einfachen Strategie ist in Abbildung 4.6 dargestellt.

Die blaue Kurve steht beispielhaft für ein Verhalten b, welches in einer Simulation beobachtet werden kann. Die y-Achse gibt die zugehörige Wahrscheinlichkeit des Verhaltens an, so dass folglich das Verhalten am Scheitelpunkt der blauen Kurve am häufigsten zu erwarten ist. Die rote Linie gibt, gemessen an der y-Achse, die Kritikalität $c(b)$ des während eines Simulationslaufs beobachteten Verhaltens an, welche als Vergütung (*Reward*) für folgende Simulationsläufe benutzt wird. Der rote Bereich am linken Rand charakterisiert kritisches Verhalten von b, welches selten ist und mit Simulationsläufen (grün gestichelte Linien) erreicht werden soll. Je weiter die Simulationsläufe vom kritischen Bereich entfernt sind, desto niedriger ist ihre Bewertung durch die Kritikalitätsfunktion (und umgekehrt). Während die

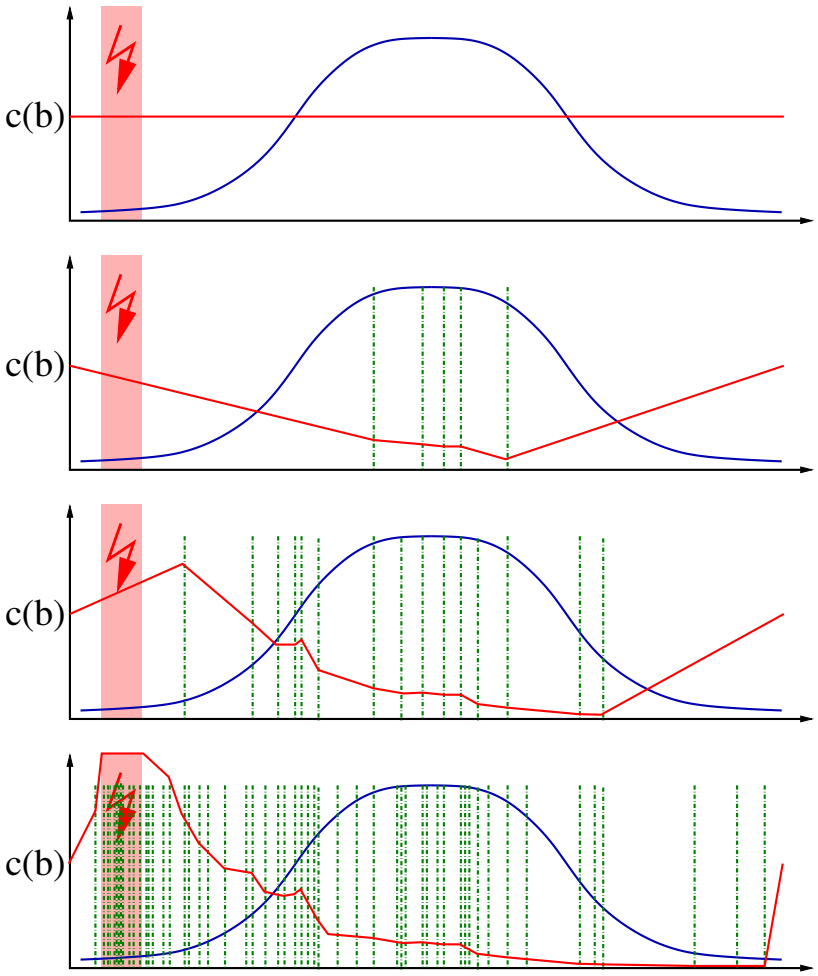

Abbildung 4.6: Prinzip der geführten Simulation mit der Kritikalitätsfunktion $c(b)$ zu seltenen kritischen Ereignissen [Puc+12b].

ersten Simulationsläufe in den Bereich der höchsten Wahrscheinlichkeit für das Verhalten von b fallen, steigt mit zunehmender Anzahl an Simulationsläufen – bedingt durch den Reward der Kritikalitätsfunktion – die Anzahl der Simulationsläufe im Bereich des kritischen Verhaltens.

4.2.2 Abstraktion als Maßnahme gegen Zustandsraumexplosion

Ein Problem bei der im letzten Unterabschnitt vorgestellten Strategie ist die Handhabung von Modellen mit sehr vielen Zufallsentscheidungen wie beim verwendeten Fahrermodell. Die

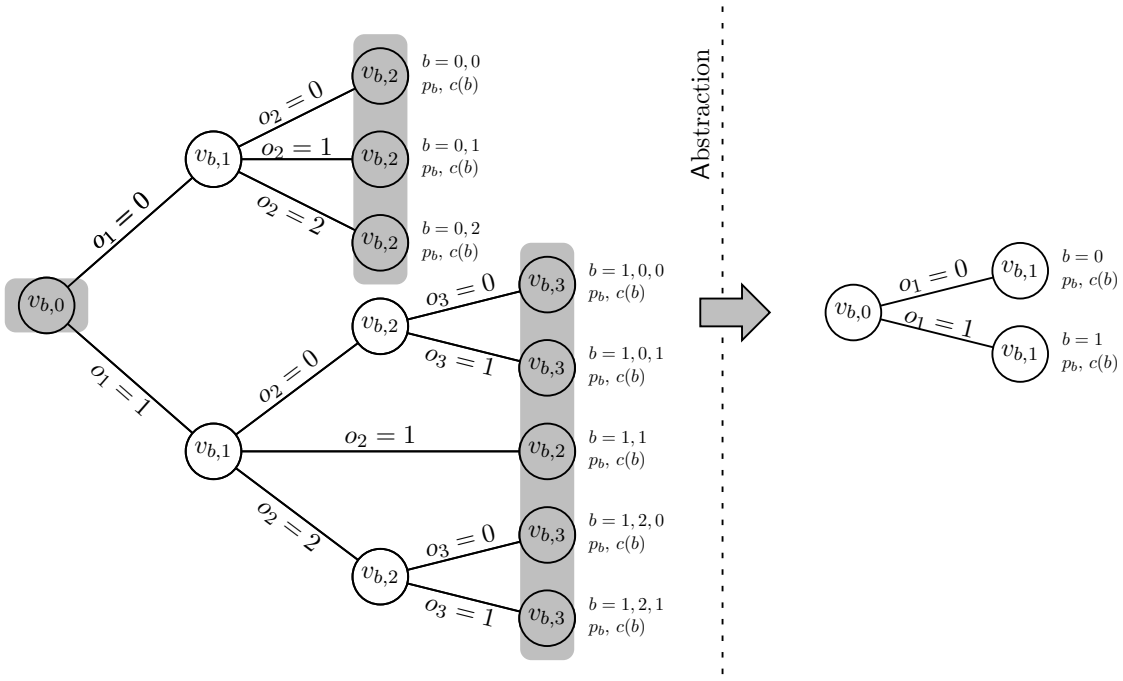

Abbildung 4.7: Abstraktion des Ereignisbaums unter Selektion von Ereignissen in den grau markierten Knoten [Puc+12b].

Vielzahl an Zufallsentscheidungen führt zu einer Zustandsraumexplosion und die Ereignisbaumgröße nimmt exponentiell zu. Eine Möglichkeit diesem Problem entgegen zu treten, ist die Einführung einer Abstraktion, ähnlich der CEGAR-Methode (vgl. Unterabschnitt 2.5.1). Dabei werden nur diejenigen Zufallsentscheidungen betrachtet, die den größten Einfluss auf das Modellverhalten haben. Im Sinne der Abstraktion entspricht ein Knoten des Ereignisbaumes folglich einer Menge an ähnlichen Simulationszuständen. Unter der Annahme, dass in Abbildung 4.7 nur die grau markierten Zustände auf der linken Seite von Interesse sind, weil sie den größten Einfluss auf das Modellverhalten haben und eine Simulationsführung die anderen Ereignisse unberücksichtigt lässt, entstünde der deutlich kleinere Baum auf der rechten Seite in Abbildung 4.7.

Bezogen auf die eingangs beschriebene einfache Strategie zur Simulationsführung bedeutet diese Abstraktion jedoch, dass die gleiche Sequenz von Optionen in verschiedenen Simulationsläufen nicht mehr zwangsweise zu identischem Verhalten führt, obwohl eine identische Abfolge von Zuständen eines Ereignisbaums durchlaufen wurde. Unter der Prämisse, dass die vernachlässigten Entscheidungen aber keinen erheblichen Einfluss haben, sollte das beobachtete Verhalten zumindest hinreichend ähnlich sein. Diese Annahme ist allerdings für jedes probabilistische Modell separat zu evaluieren und aufgrund der möglichen Abweichungen von Konsequenzen für ein Verhalten ist die Definition von $c(b)$ in Gleichung 4.2 auf das

Maximum der beobachteten Konsequenzen eines Blattes zu ändern. Während fortan abstraktionsbedingt jeder Zustand eines Ereignisbaums mit einer gewissen Unsicherheit über den korrespondierenden Modellzustand während einer Simulation verbunden ist, müssen kontinuierliche probabilistische Elemente nicht länger diskretisiert werden, sondern können direkt von ihrer originalen PDF $f_{e_c,i}$ gezogen werden, und eine Startbedingung bzw. ein Startzustand muss für alle Simulationsläufe nicht mehr identisch, sondern nur noch hinreichend ähnlich sein. Ein Knoten eines Ereignisbaums repräsentiert somit eine Klasse mit ähnlichem Verhalten, verbunden mit einer gewissen Unsicherheit.

4.2.3 Bewertung von kritischem Verhalten

Nachdem in den vorhergehenden Unterabschnitten beschrieben wurde, wie mit einer Kritikalitätsfunktion kritisches Verhalten identifiziert wird, und wie sich Abstraktion als Maßnahme gegen eine Zustandsraumexplosion einsetzen lässt, geht es um die konkrete Bewertung von kritischem Verhalten. Eine Kritikalitätsfunktion bestimmt den Gewichtungsfaktor (im Kontext von Importance Sampling die Vorschlagsdichtefunktion, welche den *importance factor* bestimmt), einen numerisch berechenbaren Wert für jeden Simulationslauf, welcher im weiteren Verlauf zur Simulationsführung benutzt wird. Die allgemeine Schwierigkeit zur Bestimmung einer guten Vorschlagsdichtefunktion wurde bereits in Unterabschnitt 2.5.2 auf Seite 22 f. behandelt. Im aktuellen Kontext, wo es um kritisches Verhalten geht, sind beispielsweise die minimale Zeit bis zum Aufprall von Interesse, aber auch hohe Beschleunigungen. Es hängt daher vom jeweiligen Anwendungsszenario ab, ob der angestrebte Zielwert ein hohes oder niedriges Maß hat.

Die für den TUTS-Algorithmus verwendete Funktion definiert einen Zielwert τ, der die akzeptablen und inakzeptablen kritischen Situationen voneinander trennt, und versucht, die Simulation in eine Region zu lenken. Um als Zielwert beliebige Maße (hohe, niedrige, minimale, maximale, etc.) nutzen zu können, wird die Nähe zum Zielwert τ in Standardabweichungen gemessen als sogenannter z-Wert (*z-score*). Der z-Wert z, in der Literatur auch unter den Begriffen Standardisierung oder z-Transformation zu finden, berechnet sich für eine Zufallsvariable x

$$z = \frac{x - \mu}{\sigma} \tag{4.3}$$

wobei μ das Verteilungsmittel und σ die Standardabweichung der Verteilung ist. Die Zufallsvariable x wird hierbei mit dem Schwellwert τ instanziiert, also dem Wert, welcher mittels geführter Simulation erreicht werden soll.

Unter der Annahme, dass sich eine Simulation im Zustand $v_{b,0}$ in Abbildung 4.8 befindet, soll die Simulationsführung zwischen den Optionen $o_1 = 0$ oder 1 so wählen, dass ein kritisches Verhalten in der Nähe von τ erreicht wird. Sei dazu $C(v)$ die Menge von Kritikalitätswerten,

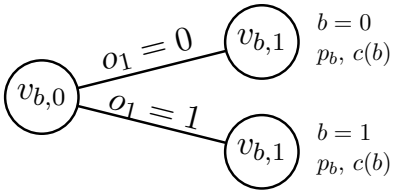

Abbildung 4.8: Abstrakter Ereignisbaum, bei dem ein Knoten für eine Menge von ähnlichen Entscheidungspunkten steht [Puc+12b].

die in allen Simulationsläufen beobachtet wurden, welche den Knoten v passiert haben. Dann wird der z-Wert des Knotens gemäß Gleichung 4.3 wie folgt berechnet:

$$z(v) = \frac{\tau - \mu(C(v))}{\sigma(C(v))} \tag{4.4}$$

Das Vorgehen gleicht einer Studentisierung, bei der die resultierenden z-Werte das arithmetische Mittel Null und die empirische Varianz Eins besitzen. Der wesentliche Unterschied ist in diesem Fall allerdings das Festsetzen von τ auf einen mit der Simulation zu erreichenden Wert, so dass das arithmetische Mittel von Null um τ verschoben wird.

Um die Wahrscheinlichkeit zu erhöhen, einen Kritikalitätswert von τ zu beobachten, sollte der τ-zentrierte Führungsalgorithmus diejenige Option bevorzugen, die zu einem absolut gesehen kleinen z-Wert führt. Ein kleiner z-Wert kann einerseits erreicht werden, wenn die empirische Standardabweichung σ eines Knotens groß ist, – was bedeutet, dass die Varianz des beobachteten Verhaltens groß ist – so dass sich eine genauere Exploration lohnt oder andererseits, wenn das Verteilungsmittel μ nahe am Zielwert liegt. Dies geschieht auf probabilistische Weise, indem alle Optionen mit einem geeigneten Gewicht bemessen werden. Dazu wird für jeden bereits im Baum vorhandenen Knoten v ein Gewicht $w(v)$ definiert. Damit Knoten mit niedrigen z-Werten ein höheres Gewicht bekommen, wird die Gewichtungsfunktion wie folgt definiert:

$$w(v) = \frac{1}{(|z|+1)^{f(v_p)}} \tag{4.5}$$

Der Algorithmus verwendet die Gewichte der Knoten, um die probabilistische Auswahl von Optionen aus dem aktuellen Satz von Optionen $\tilde{O}$ zu verändern. Im Detail wird die Wahrscheinlichkeit einer Option $P(o)$, mit der Auswahl von Option $o \in \tilde{O}$, wenn der aktuelle Knoten v ist, wie folgt definiert:

$$\mathrm{P}(o) = \frac{w(t(v,o))}{\sum\limits_{p \in \tilde{O}} w(t(v,p))} \tag{4.6}$$

Das bedeutet, dass Optionen, die zu höher gewichteten Knoten führen, mit einer größeren Wahrscheinlichkeit ausgewählt werden. Bevor ein Knoten v zweimal besucht wurde, existiert allerdings keine Standardabweichung $\sigma(C(v)) > 0$, so dass z undefiniert ist und keine Gewichtung erfolgen kann. Beim ersten Besuch eines Knotens v wählt der Algorithmus daher zunächst probabilistisch eine Option entsprechend ihrer originalen Wahrscheinlichkeitsverteilung aus. Bei jedem folgenden Besuch des Knotens werden nacheinander alle übrigen Kind-Knoten einmal ausgewählt, so dass eine implizite Breitensuche erfolgt und jeder Ast zunächst einmal beschritten wird.

Damit die Gewichtsfunktion in Gleichung 4.5 für alle existierenden z-Werte – einschließlich $z = 0$ – definiert ist, wird im Nenner der Summand von eins hinzuaddiert und für eine Feinjustierung der Exponent $f(v_p)$ benutzt. Der Knoten v_p kennzeichnet den Eltern-Knoten, so dass für jeden Geschwisterknoten der gleiche Exponent verwendet wird. Mit der Feinjustierung soll die im Laufe von mehreren Simulationen gewonnene Erkenntnis über die Verteilung der Kritikalitätswerte $C(v)$ als Vertrauensfaktor einbezogen werden, so dass mit zunehmender Erfahrung eine Situation besser eingeschätzt werden kann.

Besonders für Knoten in der Nähe der Wurzel eines Ereignisbaums kann die Varianz der Kritikalitätswerte $\sigma(C(v))$ hoch sein und Geschwisterknoten können ähnliche Mittelwerte $\mu(C(v))$ haben. Diese Knoten stehen ganz am Anfang einer Simulation und viele nachfolgende Entscheidungen beeinflussen die beobachteten Kritikalitätswerte, was zu hohen Varianzen in Knoten auf niedrigen Baumebenen führt. Um die während einer Simulation gemachten Erfahrungen und das damit einhergehende Vertrauen in die resultierenden z-Werte zu berücksichtigen, wird die Funktion f verwendet. Diese Funktion soll mit der empirischen Genauigkeit der z-Werte ansteigen und zu einer besseren Anpassung der Gewichte führen, je sicherer die Erkenntnis in Bezug auf die z-Werte ist. Für einen ersten Versuch wird zunächst eine einfache Definition mit freien Parametern a und b verwendet, um die Suchgeschwindigkeit anzupassen:

$$f(v_p) = a + b \cdot n_{min} \tag{4.7}$$

Dabei bezeichnet n_{min} die minimale Anzahl an Besuchen aller Knoten v, für die v_p Eltern-Knoten ist.

4.2.4 Rücktransformation in den originalen Wahrscheinlichkeitsraum

Mit der Bewertung von kritischem Verhalten aus dem vorhergehenden Unterabschnitt lässt sich jedem einzelnen Simulationslauf ein Faktor zuordnen, welcher für zukünftige Simulationsläufe eine Verstärkung bzw. Dämpfung bewirkt. Durch die Rückannotation dieses Faktors im Simulationspfad an jeden besuchten Knoten (vgl. Unterabschnitt 4.2.1 in Gleichung 4.2) werden sukzessive die originalen Auswahlwahrscheinlichkeiten der Optionen jedes Knotens im Entscheidungsbaum verändert, so dass seltene Ereignisse mit deutlich grö-

ßerer Wahrscheinlichkeit erreicht werden können. Das resultierende Simulationsergebnis, beispielsweise die Schätzung eines Risikos, ist bedingt durch die veränderten Wahrscheinlichkeiten folglich ebenso verzerrt. Da für eine anschließende Bewertung des Simulationsergebnisses oder eine weitere Verbesserung der verwendeten Modelle allerdings die Originalwahrscheinlichkeit relevant ist, müssen die gemäß Proposal Distribution gesammelten Stichproben mathematisch mit dem Likelihood-Quotient zurückgerechnet werden. Dafür ist es notwendig, während einer geführten Simulation neben der originalen Wahrscheinlichkeit einer Zufallsentscheidung auch die sich ändernden (gewichteten) Wahrscheinlichkeiten zu vermerken, um nach Simulationsabschluss das Ergebnis so ausweisen zu können, wie es tatsächlich existiert. Die folgende Gleichung 4.8 zeigt, wie die (originale) Wahrscheinlichkeit $\hat{p}$ eines Risikos mit Hilfe des unbefangenen Importance Sampling-Schätzers berechnet werden kann (vgl. Gleichung 2.9):

$$\hat{p} \approx \frac{1}{N} \sum_{i=1}^{N} \left(\begin{cases} 1, & \text{wenn } x_i \text{ kritisch ist} \\ 0, & \text{sonst} \end{cases} \right) \frac{p(x_i)}{q(x_i)}, \quad x_i \sim q_i. \tag{4.8}$$

N repräsentiert dabei die Anzahl an Simulationsläufen, $p(x_i)$ die ursprüngliche Wahrscheinlichkeit und $q(x_i)$ die mit dem Faktor q gewichtete Wahrscheinlichkeit von x_i.

4.2.5 Formale Analyse des TUTS-Algorithmus

Wie im letzten Abschnitt beschrieben wurde, basiert die methodische Umsetzung von TUTS auf Importance Sampling (vgl. Unterabschnitt 2.5.2). Zusätzlich werden, wie im Bereich des Verstärkungslernens (*reinforcement learning*), die Ergebnisse von Simulationsläufen gelernt und für anschließende Simulationen zur Verfügung gestellt. Der bei sukzessiven Simulationsläufen entstehende Ereignisbaum repräsentiert damit nicht nur von der Wurzel zu den Blättern betrachtet zeitbehaftete Simulationsläufe, bei denen die Knoten die probabilistischen Entscheidungspunkte widerspiegeln, sondern beinhaltet auch die TUTS gesammelten Daten zu jedem Entscheidungspunkt. So können z. B. beliebige Umgebungsvariablen an den Zuständen gespeichert werden und auch die nach jedem beendeten Simulationslauf berechnete Kritikalität wird hier vorgehalten. Durch die iterative Veränderung der Gewichtungsfunktion nach jedem Simulationslauf implementiert TUTS die Methode des Adaptive Importance Samplings.

Hinsichtlich einer numerischen Betrachtung der Simulationsergebnisse, beispielsweise der Rückrechnung für eine quantitative Bewertung (siehe Unterabschnitt 4.3.5), ist zu beachten, dass der TUTS-Algorithmus mit der z-Transformation eine Besonderheit benutzt, die einer genaueren Analyse bedarf. Die z-Transformation benötigt zur Berechnung eine Standardabweichung $\sigma > 0$, so dass mindestens zwei (verschiedene) Samples vorhanden sein müssen. Der TUTS-Algorithmus verwendet daher einen zweigeteilten Ansatz. Beim ersten Besuch

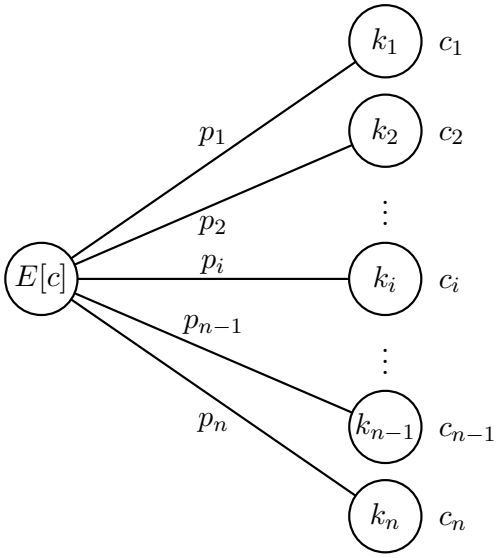

Abbildung 4.9: Einfacher Baum mit n Kind-Knoten in einer Ebene.

eines Knotens wird eine Zufallsentscheidung gemäß der originalen Wahrscheinlichkeiten der einzelnen Optionen gezogen. Bei jedem weiteren Eintreffen der gleichen Situation, d. h. beim Besuch desselben Knotens, werden zunächst nacheinander alle weiteren verfügbaren Optionen einmal ausgewählt. Damit steigt die Wahrscheinlichkeit, dass nach zwei Simulationsläufen die denselben Knoten besucht haben, zwei Samples verfügbar sind, die beim dritten Besuch die Berechnung des z-Werts und damit eines Gewichtungsfaktors ermöglichen. Der Einbezug des gelernten Verhaltens aus früheren Simulationsläufen bei der Auswahl einer Option, wie es beim IS gebräuchlich ist, findet bei TUTS folglich nicht von Anfang an statt. Für eine spätere Rückrechnung der im Laufe von vielen Simulationen angepassten Wahrscheinlichkeiten auf die originalen Wahrscheinlichkeiten muss dieses Verfahren speziell berücksichtigt werden.

So unterscheidet das Urnenmodell der Wahrscheinlichkeitstheorie, bei dem farbige Kugeln verdeckt aus einer Urne gezogen werden, zwischen dem Ziehen mit anschließendem Zurücklegen (so dass die gleiche Kugel beim nächsten Zug erneut verfügbar ist) und dem Ziehen ohne Zurücklegen. Während reines IS der ersten Variante (mit Zurücklegen) entspricht, benutzt TUTS eine Kombination aus beiden Verfahren, wobei der erste Teil dem Ziehen ohne Zurücklegen entspricht, bevor der spätere Teil, unter Einbezug des gelernten Verhaltens, dem des IS entspricht. Eine formale Betrachtung des TUTS-Algorithmus soll im Folgenden prüfen, ob für die erste Phase, in der ohne Zurücklegen gezogen wird, der mittels geführter Simulation geschätzte Erwartungswert für kritisches Verhalten $\tilde{E}[X]$ auch dem tatsächlichen $E[c]$ entspricht.

Gegeben sei ein einfacher Baum wie in Abbildung 4.9 mit einem Wurzel- und n Kind-Knoten. Ferner sei die zu einem Zweig gehörige wahre Wahrscheinlichkeit mit p_i bezeichnet und jedem Kind-Knoten eine Kritikalität c_i zugeordnet. Der reale Erwartungswert für die Kritikalität c, den der TUTS-Algorithmus schätzen müsste, ist damit gegeben durch:

$$E[c] = \sum_{i=0}^{n-1} p_i c_i \tag{4.9}$$

Innerhalb der ersten Phase wird der erste Kind-Knoten k_1 gemäß dieser Wahrscheinlichkeit gezogen. Alle Kind-Knoten desselben Eltern-Knotens werden (bis eine Berechnung der Standardabweichung möglich ist) nach einer festen Permutation π der Indizes $\{0,\ldots,n-1\}$ bestimmt. Für den ersten Kind-Knoten k_1 bezeichnet i_0 den Index innerhalb von π. Der zweite Kind-Knoten k_2 ist mit $i_1 = \pi((i_0+1) \bmod (n))$ folglich der nächste Index innerhalb der Permutation π. Das bedeutet, dass immer der nächste Index in π genommen wird, solange i nicht dem letzten Index entspricht. Anderenfalls wird wieder vorne angefangen. Allgemein kann auf diese Weise der nächste Index bestimmt werden:

$$i_j = \pi((i_0 + j) \bmod (n)) \tag{4.10}$$

Die allgemeine bijektive Zuordnung eines Kind-Knoten k_l aus einer Menge von n Kind-Knoten mit $l \in \{1,\ldots,n\}$ zu einem Index innerhalb der Permutation von π ist gegeben durch:

$$\begin{aligned} k_l &= \pi^{-1}(i_n) \\ i_{n+1} &= \pi((k_l+1) \bmod (n)) \end{aligned} \tag{4.11}$$

Für eine derartige Sequenz soll nun der Erwartungswert $\tilde{E}[X]$ für einen Satz aus m Samples berechnet werden, mit $m < n$. Die Kritikalität eines Zufallsexperiments, bei dem der erste Kind-Knoten k_1 zufällig gewählt wird und damit die nachfolgenden Kind-Knoten deterministisch sind, ist gegeben durch:

$$X = \frac{1}{m} \sum_{j=0}^{m-1} c_{i_j} \frac{p_{i_j}}{q_{i_0}} \tag{4.12}$$

Dabei wird die Kritikalität der deterministisch gezogenen Kind-Knoten mit der originalen Wahrscheinlichkeit des ersten (zufällig bestimmten) Kind-Knotens $p_{i_0} = q_{i_0}$ gewichtet. Da immer nur der erste Kind-Knoten $(k_1,\ldots,k_n)$ zufällig gezogen wird, zeigt folgende Gleichung 4.13, dass der geschätzte Erwartungswert $\tilde{E}[X]$ der Kritikalität c (im Mittel) dem tatsächlichen Erwartungswert $E[c]$ entspricht:

$$
\begin{aligned}
\tilde{E}[X] &= \tilde{E}\left[\frac{1}{m}\sum_{j=0}^{m-1} c_{i_j}\frac{p_{i_j}}{q_{i_0}}\right] \qquad (4.13)\\
&= \sum_{i_0=0}^{n-1}\left(\frac{1}{m}\sum_{j=0}^{m-1} c_{i_j}\frac{p_{i_j}}{q_{i_0}}\right) q_{i_0}\\
&= \left(\frac{1}{m}\sum_{j=0}^{m-1}\sum_{i_0=0}^{n-1} c_{i_j} p_{i_j}\right)\\
&= \left(\frac{1}{m}\sum_{j=0}^{m-1}\sum_{i_0=0}^{n-1} c_{\pi((i_0+j) \bmod (n))}\; p_{\pi((i_0+j) \bmod (n))}\right)\\
&= \left(\frac{1}{m}\sum_{j=0}^{m-1}\sum_{i_0=j}^{n-1+j} c_{\pi((i_0) \bmod (n))}\; p_{\pi((i_0) \bmod (n))}\right)\\
&= \left(\frac{1}{m}\sum_{j=0}^{m-1}\left(\sum_{i_0=j}^{n-1} c_{\pi((i_0) \bmod (n))}\; p_{\pi((i_0) \bmod (n))} + \sum_{i_0=n}^{n-1+j} c_{\pi((i_0) \bmod (n))}\; p_{\pi((i_0) \bmod (n))}\right)\right)\\
&= \left(\frac{1}{m}\sum_{j=0}^{m-1}\left(\sum_{i_0=j}^{n-1} c_{\pi(i_0)}\; p_{\pi(i_0)} + \sum_{i_0=n}^{n-1+j} c_{\pi((i_0) \bmod (n))}\; p_{\pi((i_0) \bmod (n))}\right)\right)\\
&= \left(\frac{1}{m}\sum_{j=0}^{m-1}\left(\sum_{i_0=j}^{n-1} c_{\pi(i_0)}\; p_{\pi(i_0)} + \sum_{i_0=0}^{j-1} f_{\pi(i_0)}\; p_{\pi(i_0)}\right)\right)\\
&= \left(\frac{1}{m}\sum_{j=0}^{m-1}\left(\sum_{i_0=0}^{n-1} c_{\pi(i_0)}\; p_{\pi(i_0)}\right)\right)\\
&= \left(\frac{1}{m}\sum_{j=0}^{m-1}\left(\sum_{i_0=0}^{n-1} c_{i_0}\; p_{i_0}\right)\right)\\
&= \frac{1}{m} m E[c] = E[c]
\end{aligned}
$$

□

Für die erste Phase beim Sampling mit dem TUTS-Algorithmus, die dem Ziehen ohne Zurücklegen entspricht, ist damit der Nachweis einer korrekten Schätzung im Mittel erbracht. Die zweite Sampling-Phase, dem Ziehen mit Zurücklegen, entspricht der Schätzung des klassischen Importance Sampling, welche Gleichung 2.9 auf Seite 23 entnommen werden kann. Zwar gilt der Nachweis in Gleichung 4.13 zunächst lediglich für einstufige Entscheidungsbäume wie in Abbildung 4.9 dargestellt, er lässt sich allerdings durch Induktion generalisieren, indem er auf Bäume endlicher Tiefe ausgerollt wird. Da die betrachteten Simulationen

im Kontext dieser Arbeit einer zeitlichen Beschränkung unterliegen, ist der Nachweis für Bäume endlicher Tiefe ausreichend.

4.3 Evaluation des TUTS-Algorithmus

Nach der Beschreibung des TUTS-Algorithmus in Abschnitt 4.2, einschließlich einer formalen Analyse, erfolgt in diesem Kapitel die praktische Evaluation[1]. Anhand eines Fahrermodells, welches mit der kognitiven Architektur CASCaS erstellt wurde, soll in einem ersten Evaluationsschritt anhand eines einfachen Fahrszenarios überprüft werden, ob sich die in Unterabschnitt 4.2.2 beschriebenen Maßnahmen innerhalb eines Fahrermodells umsetzen lassen. In einem zweiten Evaluationsschritt soll anschließend anhand eines realitätsnahen Fahrszenarios getestet werden, wie sich mit TUTS innerhalb einer Kosimulation seltene Ereignisse sichtbar machen lassen.

4.3.1 Abstraktion probabilistischer Prozesse in CASCaS

Bei der ersten Evaluation des TUTS-Algorithmus soll die Frage geklärt werden, ob sich das in Unterabschnitt 4.2.2 beschriebene Verfahren für CASCaS eignet. Während einer Simulation von CASCaS wird durch den TUTS-Algorithmus ein zeitbehafteter Ereignisbaum erstellt, welcher die probabilistischen Entscheidungen des Fahrermodells repräsentiert. Die Simulationsführung des Algorithmus startet am Wurzelknoten und endet an einem Blatt. Jeder Knoten des Baums repräsentiert eine Zufallsentscheidung, jeder Zweig eine mögliche Entscheidung. Um einer Zustandsraumexplosion entgegenzuwirken, werden in CASCaS jedoch nicht alle, sondern zwecks Abstraktion nur einige Klassen von Zufallsentscheidungen berücksichtigt. So haben die diskreten probabilistischen Elemente Ziel- und Regelauswahl (vergleiche Initialer Abstraktionslevel in Abbildung 4.10 auf der rechten Seite) typischerweise den größten Einfluss auf das simulierte Verhalten, die Auswirkung aller weiteren probabilistischen Prozesse auf das Gesamtverhalten sind sehr stark vom simulierten Kontext abhängig. Bei einer Abstraktion von probabilistischen Prozessen, die Rauschkomponenten beinhalten (Fixationszeit, Handbewegung, etc.), wird angenommen, dass ein Knoten im Ereignisbaum repräsentativ für ähnliche Situationen während mehrerer Simulationsläufe steht. Einen abstrakten, zeitbehafteten Ereignisbaum von CASCaS, bei dem in den Knoten nur die probabilistische Ziel- und Regelauswahl berücksichtigt wurde, zeigt die schematische Darstellung in Abbildung 4.10 auf der linken Seite. Probabilistische Prozesse, die nicht berücksichtigt werden, sind im Ereignisbaum nicht als eigene Zustände vorhanden, sondern

[1] Die Rohdaten der Simulationsergebnisse und zugehörige Scripte für die Generierung ihrer Visualisierung sind unter https://uol.de/hs/downloads zugänglich.

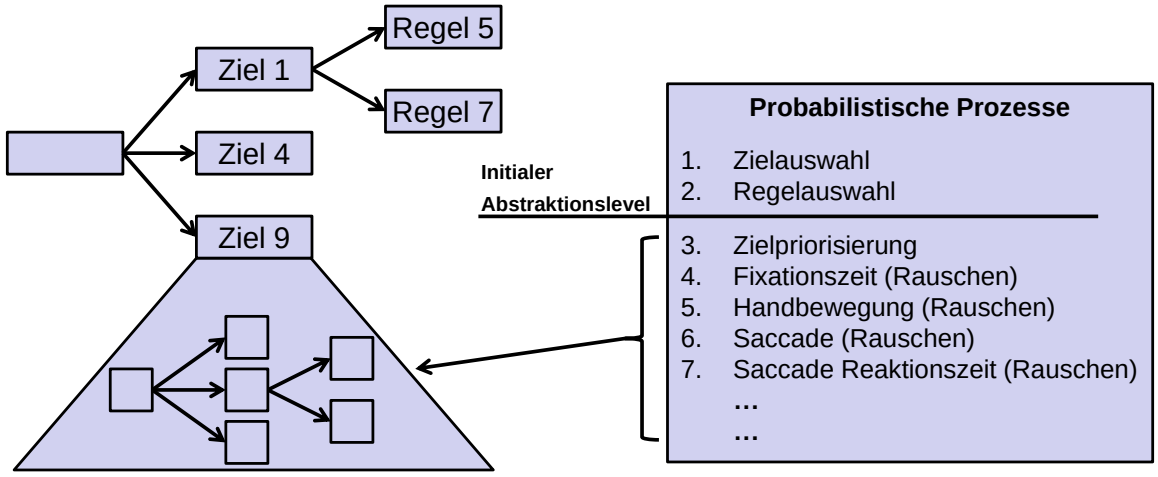

Abbildung 4.10: Schematischer Ereignisbaum mit probabilistischer Ziel- und Regelauswahl von CASCaS.

versteckt innerhalb eines abstrakten Knotens, was in Abbildung Abbildung 4.10 im Knoten mit Ziel 9 angedeutet wurde. Eine hohe Varianz von Simulationswerten aus der Umgebung (z. B. Simulationszeit, Position, etc.), die bei einem Knoten mitgespeichert werden können, würde eine notwendige Verfeinerung der Abstraktion implizieren, da in diesem Fall ein Knoten nicht für eine hinreichend ähnliche Situation steht.

4.3.2 Evaluation der Abstraktionsidee in CASCaS

Zur Evaluation der Annahme, dass ein Knoten im Ereignisbaum repräsentativ für ähnliche Situationen während mehrerer Simulationsläufe steht, wurde eine Kosimulation im Automobilbereich benutzt, bestehend aus einem in CASCaS entwickelten Fahrermodell und der Fahrsimulationssoftware SILAB (Version 3.0), zur Simulation von Fahrzeug, Umgebung und Umgebungsverkehr. Das Szenario (siehe Abbildung 4.11) besteht aus einer zweispurigen Autobahn mit mäßigem Verkehr, bei dem sich das Fahrermodell einem langsamer vorausfahrenden Fahrzeug annähert, während auf der linken Fahrspur schnellere Fahrzeuge überholen. Die Aufgabe des Fahrermodells besteht darin, seine Geschwindigkeit entsprechend anzupassen und zu überholen, wenn es die Situation bezüglich des umliegenden Verkehrs erlaubt. Der Startzustand für die Simulationsführung wird erreicht, wenn der Abstand des EgoCar zum führenden Fahrzeug unter 200 m fällt und die Geschwindigkeit über 25 m/s (entspricht 90 km/h) liegt. Die Simulation wird abgebrochen, sobald entweder

1. die Geschwindigkeit des EgoCar unterhalb der Geschwindigkeit des vorausfahrenden Fahrzeugs liegt und somit hinter dem Fahrzeug ein sicherer Zustand erreicht ist,

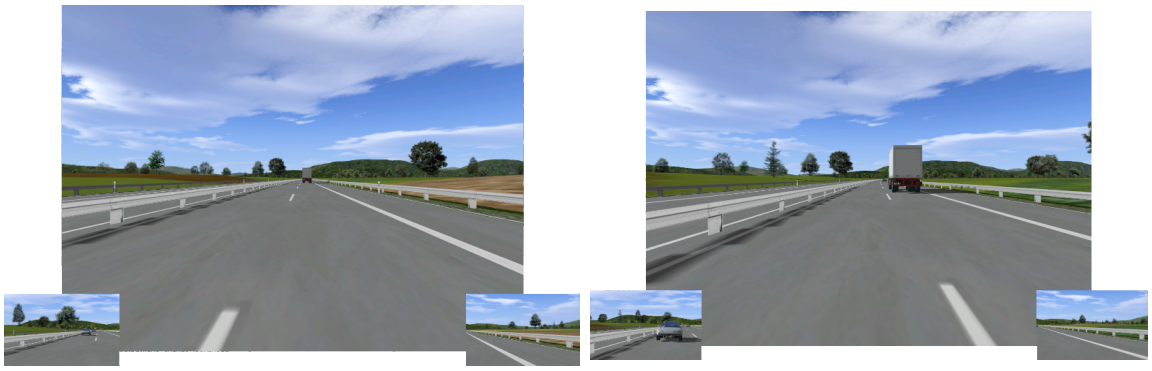

Abbildung 4.11: Autobahnszenario mit Annäherung an ein vorausfahrendes Fahrzeug.

2. das EgoCar sicher auf die linke Fahrspur gewechselt hat, um das führende Fahrzeug zu überholen oder

3. mehr als 180 s seit Erreichen des Startzustands vergangen sind, da ein einzelner Simulationslauf in der Regel nicht länger als eine Minute dauert.

Die Bewertung der Kritikalität $c(b)$ eines Simulationslaufs (vergleiche Abbildung 4.5) erfolgte anhand der von Hydéns definierten Schwere eines Konfliktlevels [Hyd87], welche mit einer entsprechenden *Deceleration to Safety Time*[2] *(DST)* einhergeht [Hup97a; Hup97b]. Da es bei der Evaluation nicht um den Sachverhalt eines geeigneten Kritikalitätsmaßes geht, wird auf eine genauere Beschreibung an dieser Stelle verzichtet. Die Beantwortung der Fragestellung, ob ein Knoten innerhalb eines während der Simulation erzeugten Ereignisbaumes stellvertretend für ähnliches Verhalten steht, ist in Abbildung 4.12 bei der grafischen Darstellung der Deceleration to Safety Time ersichtlich. Auf der linken Seite ist ein schematischer Ereignisbaum des Simulationsergebnisses dargestellt. Da während der Simulation des Fahrermodells nur die probabilistische Regelauswahl betrachte wurde, ist in jedem Knoten, bedingt durch die weiteren vernachlässigten probabilistischen Prozesse, eine Verteilung des DST-Werts festzustellen. An den grau gestrichelten Linien wird der Baum entlang der Pfadtiefe in Ebenen zerlegt und jeweils die Standardabweichung über alle Knoten der Ebene (rote Kurve an den Blättern) bestimmt. Im Ergebnis der roten Kurve auf der rechten Seite von Abbildung 4.12 ist zu erkennen, dass die Standardabweichung von der Wurzel bis zur Blattebene in Tiefe 60 deutlich höhere Werte hat als die blaue Kurve. Letztere entspricht dem Mittelwert der Standardabweichungen aller Knoten einer Ebene. Dabei wurde, wie im linken Teil der Grafik an den blauen Kurven dargestellt, zunächst für jeden Knoten einer Ebene die Standardabweichung bestimmt und anschließend der Mittelwert der Standardabweichungen über alle Knoten berechnet.

[2]Deceleration to Safety Time ist definiert als die notwendige Verzögerung zum Erreichen eines Sicherheits-Zeitabstandes.

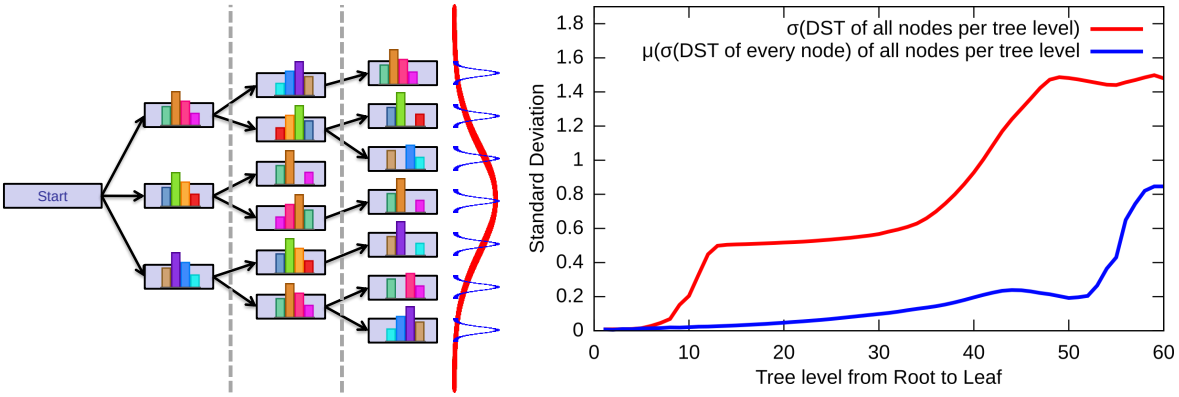

Abbildung 4.12: Standardabweichung der DST pro Ereignisbaumebene.

Das Ergebnis, dass die gemittelte Standardabweichung (blaue Kurve) unterhalb der Standardabweichung aller Knoten (rote Kurve) liegt, entspricht der Erwartung, dass ein Knoten ähnliches Verhalten gruppiert. Wäre dem nicht so, beispielsweise bei einer rein zufälligen Gruppierung, müsste die blaue Kurve nahezu identisch zur roten Kurve sein. Es ist nun zu untersuchen, wie viel besser das Ergebnis der TUTS-geführten Simulation im Vergleich zur rein zufälligen Monte-Carlo-Simulation ist. Um zusätzlich eine bessere Einschätzung zu erlangen, wie groß der Einfluss nicht berücksichtigter probabilistischer Prozesse ist, werden im folgenden Evaluationsszenario zwei verschiedene Abstraktionslevel simulativ erfasst und anschließend bewertet.

4.3.3 TUTS-geführte Simulation eines Fahrer-Fahrzeug-Assistenzsystem-Szenarios

In diesem Unterabschnitt erfolgt die Evaluation des TUTS-Algorithmus anhand eines möglichst realistischen Anwendungsszenarios, so wie es eingangs der Arbeit bei der Forschungsfrage formuliert wurde. Das Hauptziel besteht darin, mit der geführten Simulation das Verhalten des Fahrermodells zu explorieren und dabei valides Fahrerverhalten zu finden, welches seltene und kritische Situationen weitaus häufiger aufweist, als es bei reinen Monte-Carlo-Simulationen der Fall ist. Abgeleitet wurde der Aufbau von einer Studie in einem Fahrsimulator [WBL13; WLB13], deren Ziel es war, die Aufmerksamkeitsverteilung eines Fahrers beim Fahren auf einer kurvenreichen Straße zu untersuchen, während er zeitweise mit einer sekundären Aufgabe im Fahrzeug beschäftigt war. Die Fahrer wurden angewiesen, sich auf drei unterschiedliche Ziele zu konzentrieren:

1. Das Steuern des Fahrzeugs in der Mitte der Fahrspur,

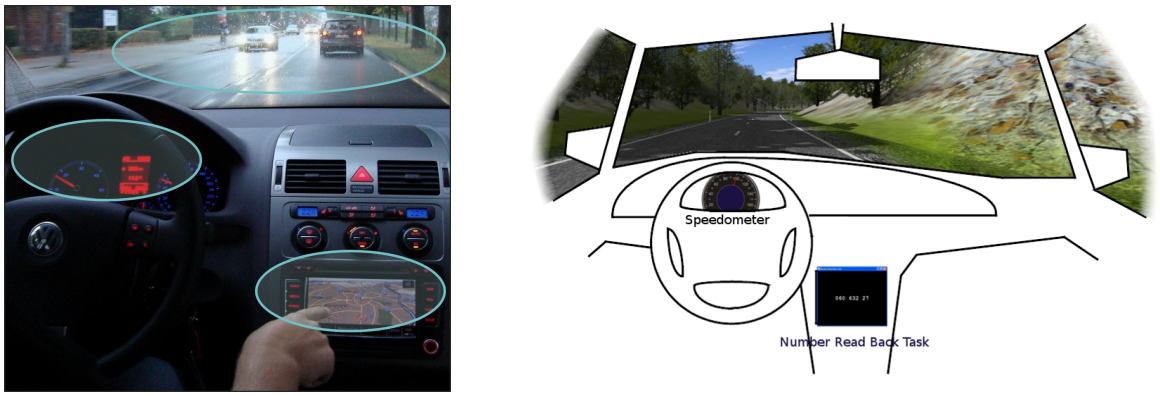

Abbildung 4.13: Fahrzeugcockpit (links) und Schematische Darstellung für das Fahrermodell (rechts) mit drei Informationsquellen: 1. Die Straße voraus durch die Frontscheibe. 2. Der Tachometer im Kombiinstrument. 3. Das Display des Infotainmentsystems in der Mittelkonsole.

2. das konstante Beibehalten einer Geschwindigkeit von 100 km/h so gut wie möglich und

3. das möglichst schnelle Lösen einiger Aufgaben, die auf einem Display im Fahrzeug in unterschiedlichen Zeitintervallen angezeigt wurden.

Die dritte Aufgabe, der sogenannte *Number Read Back Task (NRBT)* [HWC06], steht dabei repräsentativ für die Interaktion mit einem Infotainment- oder Assistenzsystem, welches in der Mittelkonsole des Fahrzeugs verbaut wurde. Um den Anforderungen aller Ziele gerecht zu werden, muss die Aufmerksamkeit des Fahrers zwischen den drei Aufgaben mit den zugehörigen Informationsbereichen (Straße, Tachometer, Display) gewechselt werden. Abbildung 4.13 zeigt einen entsprechenden Ausschnitt aus einem Fahrzeugcockpit.

Basierend auf den Ergebnissen der Studie mit insgesamt 17 Probanden wurde von Wortelen in [WBL13; WLB13] ein Fahrermodell mit CASCaS entwickelt, welches in der Lage ist, das Fahrverhalten und das visuelle Blickverhalten nachzubilden. Eine schematische Darstellung des im Szenario simulierten Fahrzeugcockpits kann ebenfalls Abbildung 4.13 entnommen werden. Der Aufbau des Fahrszenarios ist einfach gehalten und wurde in SILAB modelliert. Es besteht aus einer kurvenreichen Straße mit Kurvenradien zwischen 375 m und 750 m. Der Fahrer hat keine komplexen Aufgaben zu lösen und es existieren weder Umgebungsverkehr, noch Verkehrszeichen oder Kreuzungen. Die wichtigste Aufgabe des Fahrers ist die Verteilung der Aufmerksamkeit auf die drei Ziele. Wenn zu wenig darauf geachtet wird, das Auto innerhalb der Fahrspur zu halten, können die seitlichen Fahrbahnlinien überschritten werden. Wird der Tachometer nicht beachtet, kann es zu Abweichungen bei der Geschwindigkeitsvorgabe kommen und wenn der Fahrer seinen Blick nicht ab und zu auf das Display in der Mittelkonsole richtet, sinkt die Performanz für diese Aufgabe.

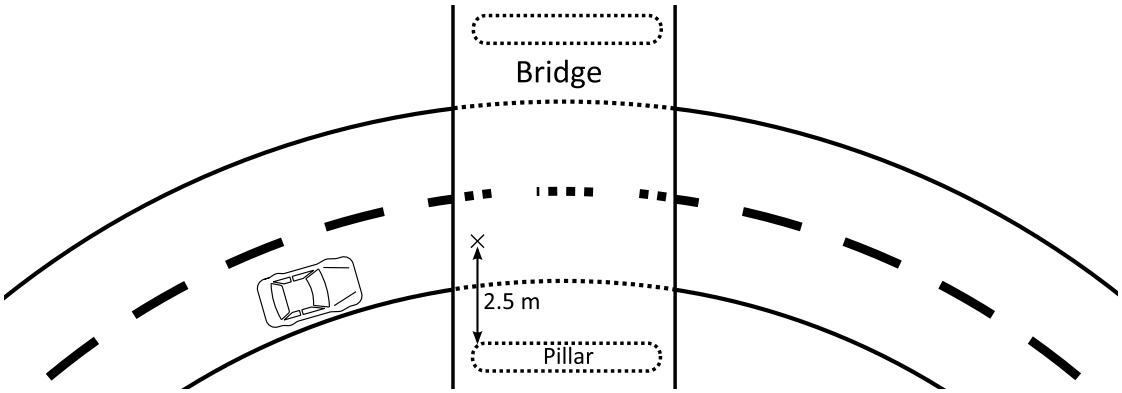

Abbildung 4.14: Realistisches Fahrszenario mit Brücke. Der Abstand des Brückenpfeilers zur Mitte der Fahrspur beträgt 2,5 m [Puc+13].

Die Aufgabe des TUTS-Algorithmus in diesem Szenario besteht darin, ein Verhalten des Fahrers zu finden, bei dem die Interaktion mit dem Display des Infotainmentsystems bzw. stellvertretend die Bearbeitung des NRBT durch das Fahrermodell zu kritischen Situationen führt. Um die Funktionsweise des Algorithmus, d. h. das Führen der Simulation hin zu kritischen Situationen, zu demonstrieren, wurde abweichend zum ursprünglichen Szenario an einer Stelle im Szenario eine Brücke über die Straße eingefügt. Der Abstand von der Mitte der Fahrspur zum seitlichen Brückenpfeiler beträgt, wie in Abbildung 4.14 dargestellt, 2,5 m. Eine kritische Situation existiert folglich in dem Bereich, wenn das Fahrzeug die Brücke passiert, da es eine Kollision mit dem Pfeiler wie in der Realität zu vermeiden gilt. Als Kritikalitätswert für die Simulationsführung wurde der minimale Abstand des Fahrzeugmittelpunktes zum rechten Brückenpfeiler verwendet, sowie die freien Parameter $a = b = 0,5$ für die Gewichtungsfunktion (vgl. Gleichung 4.7 auf Seite 70).

Die Simulation ist aus Sicht des Fahrermodells in zwei Phasen unterteilt. Zu Beginn jedes Simulationslaufs ist das Fahrzeug auf eine Geschwindigkeit von 100 km/h zu beschleunigen. Die zweite Phase startet mit dem Erreichen der Geschwindigkeit und definiert die Anfangszustände der geführten Simulation. Jeder Simulationslauf endet, nachdem das Fahrzeug den Brückenpfeiler um 20 m passiert hat, oder wenn ein vordefiniertes Zeitlimit erreicht ist. Die mit kleinen Abweichungen typische Zeitspanne zwischen Anfangs- und Endzustand der geführten Simulation beträgt etwa 8 Sekunden und der kleinste Abstand zum Brückenpfeiler wird ca. 7 Sekunden nach einem Anfangszustand erreicht. Für die vom TUTS-Algorithmus kontrollierten probabilistischen Elemente von CASCaS wurden zur Abstraktion die in Unterabschnitt 4.3.1 beschriebenen diskrete Prozesse ausgewählt, welche den größten Einfluss auf das simulierte Verhalten haben. So entspricht der wichtigste Aspekt des Fahrers in dem Fahrszenario (die Verteilung der Aufmerksamkeit auf drei unterschiedliche Bereiche) der Aufgabe einer Zielauswahl (*Goal Selection*) und der Abarbeitung einer Aufgabe, jeweils mit

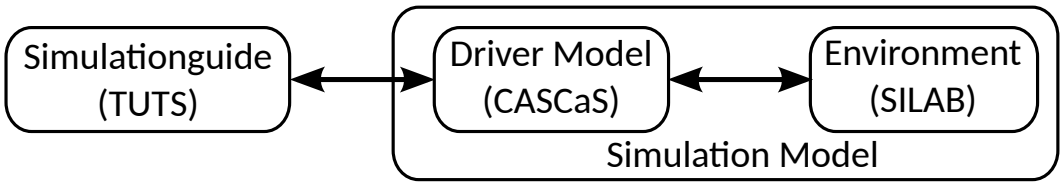

Abbildung 4.15: Schematischer Aufbau für die TUTS-geführte Simulation [Puc+13].

einem Satz von Regeln (*Rule Selection*), vergleiche Abbildung 4.10 auf der rechten Seite. Der technische Simulationsaufbau ist schematisch in Abbildung 4.15 dargestellt und entspricht einer Instanz des Kosimulationsframeworks, vergleiche Abbildung 3.1 auf Seite 34. Die Umgebungssimulation in SILAB wurde in diesem Aufbau zusammen mit der Pitch RTI auf einem Computer instanziiert, das Fahrermodell CASCaS zusammen mit dem Simulationguide auf einem anderen Computer. Beide Computer waren über ein lokales 100 MBit/s Netzwerk miteinander verbunden.

4.3.4 Qualitative Auswertung der Simulationsergebnisse

Um die Ergebnisse der geführten Simulation mit denen einer Simulation ohne intelligente Steuerung vergleichen zu können, wurde das Szenario in drei Konfigurationen simuliert. Zuerst wurden 10.000 Simulationsläufe mit Monte-Carlo-Simulation ohne jegliche Simulationsführung durchgeführt, um eine Ausgangsbasis für Vergleiche zu schaffen. Im Anschluss wurden 10.000 Simulationsläufe durchgeführt, bei dem der Simulationguide die Zielauswahl geführt hat und weitere 10.000 Simulationsläufe, bei dem zusätzlich die Regelauswahl geführt wurde. Alle anderen probabilistischen Prozesse von CASCaS wurden, gemäß der vom Fahrermodell definierten Verteilung, zufällig bestimmt. Jeder Knoten des nach 10.000 Simulationsläufen gewonnenen Ereignisbaums sollte, bedingt durch die unberücksichtigten probabilistischen Prozesse, eine Varianz hinsichtlich des aktuellen Simulationszustands aufweisen. Die Varianz sollte kleiner sein, je mehr probabilistische Prozesse der Simulationguide geführt bzw. kontrolliert (im Sinne von beeinflusst) hat. Gleichzeitig erhöht eine zunehmende Kontrolle die Größe des Ereignisbaums. Dieser Effekt ist in Abbildung 4.16 und Abbildung 4.17 dargestellt und sollte wie folgt gelesen werden:

Bei jedem Erreichen eines Knotens während eines Simulationslaufs wurde der aktuelle Zeitabstand (Δt) seit Anfangszustand der geführten Simulation und die laterale Position des Fahrzeugs von der Mitte der Fahrspur (d_L) aufgezeichnet. Die laterale Position gemessen in Metern beträgt 0 m, wenn sich das Fahrzeug in der Mitte der Fahrspur befindet. Der Wert erhöht sich, wenn das Fahrzeug zum rechten Rand der Fahrspur fährt und nimmt bei einer Abweichung nach links ab. Nach Ende von jeweils 10.000 Simulationsläufen wurden für jeden Knoten die Mittelwerte für Δt und d_L sowie deren Standardabweichung berechnet. Damit

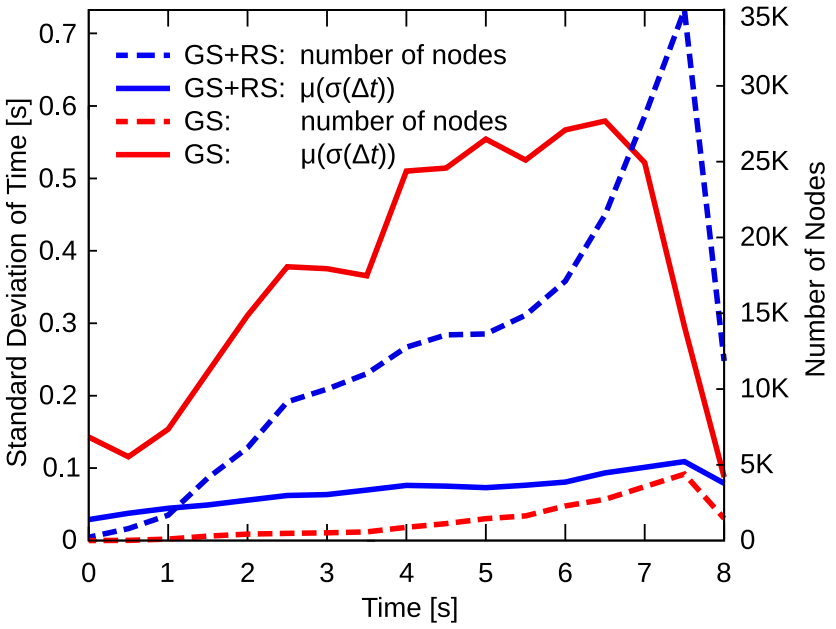

Abbildung 4.16: Die Anzahl an Knoten sowie die mittlere Standardabweichung $\mu(\sigma(x))$ der Simulationszeit innerhalb eines Zeitfensters von 0,5 s [Puc+13].

eine Standardabweichung definiert ist, wurden nur Knoten berücksichtigt, die mindestens zweimal erreicht wurden. Die x-Achse mit der Simulationszeit seit Beginn der Simulationsführung wurde in 500 ms-Schritten diskretisiert und entspricht damit den Pfaden von der Wurzel des Ereignisbaums zu den Blättern. Alle Knoten mit einer mittleren Zeit innerhalb eines 500 ms-Fensters wurden aggregiert.

Die beiden durchgezogenen Linien in Abbildung 4.16 zeigen die mittlere Standardabweichung $\mu(\sigma(x))$ für Δt aller Knoten in einem 500 ms Fenster an der rechten y-Achse. Es ist ersichtlich, dass die mittlere Standardabweichung für die Simulationsläufe, wo sowohl die Ziel- als auch die Regelauswahl (GS+RS in blau) vom Simulationsguide kontrolliert wurde, deutlich kleiner ist als die Simulationsläufe, bei denen lediglich die Zielauswahl (GS in rot) berücksichtigt wurde. Bei der reinen Zielauswahl hatten die Knoten zum Zeitpunkt 6,5 s beispielsweise bereits eine Standardabweichung von über einer halben Sekunde, während sie bei der gleichzeitigen Berücksichtigung von Regel- und Zielauswahl lediglich bei einer Zehntelsekunde liegt. Das Verhaltensspektrum des Fahrermodells, welches durch einen einzelnen Knoten repräsentiert wird, ist folglich bei einer höheren Simulationskontrolle deutlich geringer. Gleichzeitig ist anhand der gestrichelten Linien und der y-Achse an der rechten Seite ersichtlich, dass die Größe des Ereignisbaums während einer Simulation mit hoher Simulationskontrolle viel schneller steigt. Es existiert also ein gewisser Zielkonflikt für eine geführte Simulation zwischen der Komplexität des Zustandsraumes (Anzahl an Knoten, die

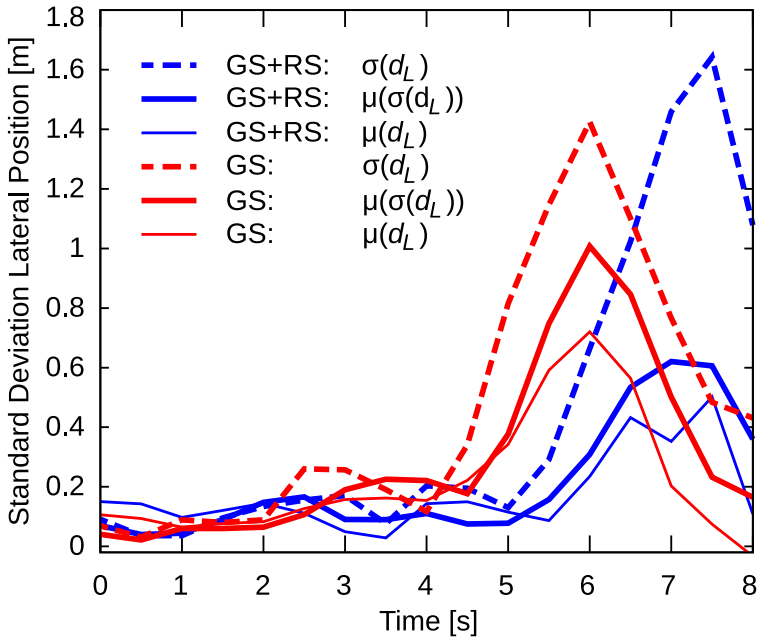

Abbildung 4.17: Der Mittelwert $\mu(x)$, die Standardabweichung $\sigma(x)$ und die mittlere Standardabweichung $\mu(\sigma(x))$ der lateralen Fahrzeugpositionen aller Knoten innerhalb eines Zeitfensters von 0,5 s [Puc+13].

betrachtet werden müssen) und der Simulationskontrolle. Letztere ist bei einer geringerer Standardabweichung innerhalb eines Knotens deutlich besser. Der starke Abfall aller Kurven nach etwa 7 s ist darauf zurückzuführen, dass, bedingt durch das Abbruchkriterium der Simulation nach dem Passieren des Brückenpfeilers, einige Simulationsläufe etwas früher als andere enden und die Anzahl der Knoten damit geringer ausfällt.

Die Daten der lateralen Fahrzeugposition in Abbildung 4.17 wurden auf die gleiche Weise aggregiert. Die dicken durchgezogenen Linien zeigen die mittlere Standardabweichung für d_L, die gestrichelten Linien zeigen die gesamte Standardabweichung, welche über alle beobachteten d_L-Werte von allen Knoten innerhalb jedes Zeitfensters berechnet wurden. Insbesondere bei einem Fokus auf die letzten 4 - 5 Sekunden der Simulationsläufe ist zu erkennen, dass, wie schon bei dem Ergebnis des ersten Evaluationsbeispiels in Abbildung 4.12 auf Seite 78, die mittlere Standardabweichung der Zielauswahl (GS in rot) geringer ausfällt als die Standardabweichung über alle Knoten. Ferner ist ersichtlich, dass bei gleichzeitiger Kontrolle von Ziel- und Regelauswahl (GS+RS in blau) zum einen der Abstand zwischen der gestrichelten und der dicken durchgezogenen Linie vergrößert werden konnte, was für eine bessere Simulationskontrolle steht, und zum anderen der Anstieg beider Standardabweichungen vom zeitlichen Verlauf (Vergleich GS in rot versus GS+RS in blau) erst später beginnt. Für die ersten 4 Sekunden der Simulationsläufe hingegen ist das beobachtete Verhalten beider Konfigurationen mit Blick auf die Standardabweichungen sehr ähnlich. Gleiches trifft auf die mittlere laterale Fahrzeugposition zu, welche mit den dünnen durchgezogenen Lini-

en dargestellt ist. Im weiteren Verlauf der Simulationen driftet die laterale Fahrzeugposition in beiden Konfigurationen in Richtung des Brückenpfeilers. Bei der GS-Konfiguration ist die höchste laterale Abweichung bei etwa $\Delta t = 6$ Sekunden erreicht, wobei der Brückenpfeiler erst in der Nähe von $\Delta t = 7$ Sekunden erreicht wird. Im Gegensatz dazu ist die höchste laterale Abweichung bei der GS+RS-Konfiguration genau in dem Bereich bei $\Delta t = 7$ Sekunden zu erkennen, in dem auch der Brückenpfeiler erreicht wird. Ein möglicher Grund für diesen Unterschied ist darin zu sehen, dass der Simulationguide in der GS-Konfiguration zu wenig Kontrolle über das Verhalten des Fahrermodels hat und damit eine präzise geführte Simulation nicht möglich ist. Zur besseren Absicherung dieser These wären weitere Analysen notwendig, die aber nicht im Kontext dieser Arbeit erbracht werden konnten.

Hinsichtlich des Hauptziels des Evaluationsszenarios, mit der geführten Simulation seltene und kritische Situationen weitaus häufiger zu finden als es bei reiner Monte-Carlo-Simulation der Fall ist, wurden in Abbildung 4.18 die Ergebnisse aller drei Konfigurationen verglichen. Die Abbildung zeigt auf der y-Achse die Häufigkeitsverteilung aus jeweils 10.000 Simulationsläufen gegenüber dem in 0,1 m diskretisierten Kritikalitätswert auf der x-Achse, welcher zur Erinnerung dem minimalen Abstand des Fahrzeugmittelpunktes zum rechten Brückenpfeiler entspricht. Die Verteilung der Monte-Carlo-Simulation in der roten Kurve ist sehr schmal um den Wert verteilt, welcher der Entfernung des Mittelpunktes der Fahrspur zum Brückenpfeiler entspricht. Insgesamt weisen 7.272 der 10.000 Simulationsläufe einen Kritikalitätswert von rund 2,4 m auf und 9.994 von 10.000 lateralen Abweichungen liegen innerhalb der Spurgrenzlinien, obwohl das Fahrermodell zwischendurch mit der Nebenaufgabe des NRBT beschäftigt ist. Der geringste Abstand, der während der gesamten 10.000 Simulationsläufe beobachtet werden konnte, liegt bei etwa 0,7 m, was von einer frontalen Kollision mit dem Brückenpfeiler noch sehr weit entfernt ist. Die Wahrscheinlichkeit, dass eine Korrekturmaßnahme des Fahrermodells die Situation nach einer Ablenkung durch die Nebenaufgabe noch retten kann, ist hingegen sehr hoch; so hoch, dass durch eine Simulationszeit von 1 Woche, welche für die 10.000 Simulationsläufe benötigt wurde, kein einziger Unfall (auch kein Beinaheunfall) bedingt durch die Nebenaufgabe aufgedeckt werden konnte.

Anders sieht es hingegen aus, wenn der Simulationguide mit dem TUTS-Algorithmus die Simulation mit der GS-Konfiguration, repräsentiert durch die grüne Kurve, oder mit der GS+RS-Konfigurationen, dargestellt mit der blauen Kurve, geführt hat. Beide Verteilungen sind einseitig stark in Richtung kleinerer Abstandswerte zum Brückenpfeiler hin ausgerichtet und insbesondere bei der GS+RS-Konfigurationen ist eine hohe Anzahl an Simulationsläufen ersichtlich, die nahe an einem Abstand von 0 m liegen. Wird das Eintreten einer kritischen Situation beispielsweise mit einem Abstand zum Brückenpfeiler von $< 0{,}5$ m definiert, so enthalten von 10.000 Simulationsläufen mit der Monte-Carlo-Simulation 0%, mit der geführten Simulation bei der GS-Konfiguration 4,1% und bei der geführten Simulation mit GS+RS-Konfiguration 9,2% eine kritische Situation.

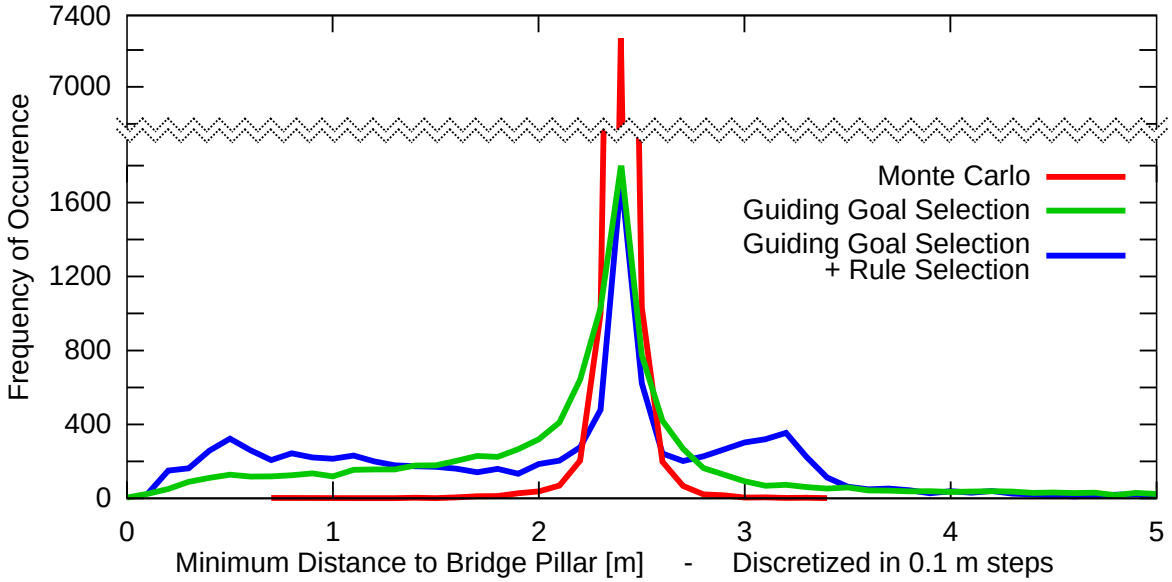

Abbildung 4.18: Häufigkeitsverteilungen der Kritikalitätswerte bei geführter Simulation im Vergleich zu Monte-Carlo-Simulation. Größere Häufigkeiten bei kleinen Abständen zeigen, dass die geführte Simulation erfolgreich die kritischen, seltenen Ereignisse ansteuert. [Puc+13].

Auffällig ist, dass bei den geführten Simulationen im Gegensatz zur Monte-Carlo-Simulation auch höhere Abstände zum Brückenpfeiler zu erkennen sind. Die Ursache hierfür ist darin begründet, dass der Simulationguide das Verhalten des Fahrermodells zunächst in seiner Breite exploriert, was exemplarisch auch zu höheren Abständen führen kann, bevor der TUTS-Algorithmus mit seiner Kombination aus Verstärkungslernen (*reinforcement learning*) und Adaptive Importance Sampling vermehrt die Simulationsläufe bevorzugt, die in Richtung kleinerer Abstände zum Brückenpfeiler führen. Ein weiterer Grund für die Ausreißer hin zu größeren Abständen ist darin zu sehen, dass viele Entscheidungen des Simulationguides die latarale Kontrolle des Fahrermodells destabilisieren, was ebenfalls dazu führt, dass nicht nur seitliche Abweichungen von der Fahrspurmitte nach rechts, sondern ebenso nach links im Simulationsergebnis sichtbar werden.

Die qualitative Evaluation des TUTS zeigt, dass seltene, im gewählten Kontext des Szenarios auch sicherheitskritische Situationen simulativ erfasst werden können. Dem Entwickler eines Fahrerassistenzsystems für eine Kollisionsvermeidung ermöglicht diese Methodik beispielsweise die Identifikation von Situationen, in denen sein System korrekt funktionieren sollte. Der Entwickler eines Fahrermodells kann die Methodik einerseits einsetzen, um die Randbereiche seines Modells zu explorieren oder andererseits, um Situationen zu identifizieren, die im Anschluss einmal spezifisch im Rahmen einer Fahrsimulatorstudie analysiert werden können. Der folgende Abschnitt widmet sich nun der Belastbarkeit der erreichten Simulationsergebnisse einhergehend mit einer statistischen Datenanalyse.

4.3.5 Quantitative Auswertung der Simulationsergebnisse

Nachdem im vorhergehenden Abschnitt aufgezeigt wurde, wie sich seltene Ereignisse mittels geführter Simulation innerhalb eines Fahrer-Fahrzeug-Assistenzsystem-Szenarios sichtbar machen lassen, stellt sich daran anschließend die Frage, wie zuverlässig bzw. valide die gewonnenen Daten sind. Da sich die Gültigkeit von simulativ erhobenen Daten in einem derart komplexen Aufbau zumeist nicht durch einen formalen Beweis belegen lässt, wird als anerkannte alternative Möglichkeit auf statistische Analyse zurückgegriffen, die beispielsweise mit der Konfidenz zum Ausdruck bringt, inwieweit einem Datensatz vertraut werden kann oder mit dem Signifikanzniveau, welche Irrtumswahrscheinlichkeit vorliegt. Für die Zulassung von Assistenzsystemen, insbesondere in sicherheitskritischen Bereichen wie der Luftfahrt, ist die quantitative Angabe der Sicherheit eines Systems anhand einer Risikoanalyse nachzuweisen. Dazu wird der Fehler, welcher zu einem Systemausfall führt, zunächst kategorisiert (geringfügig, schwer, gefährlich, katastrophal) und anhand einer Tabelle, welche einen Zusammenhang zwischen Wahrscheinlichkeit und Schwere der Fehlerbedingung herstellt, kann anschließend ermittelt werden, wie groß das Risiko für diesen Fehler sein darf. Ein Systemausfall, welcher katastrophale Folgen hat (beispielsweise mehrere Todesopfer), darf höchstens mit einer Wahrscheinlichkeit von 10^{-9} auftreten [EAS07].

Für die quantitative Bewertung der erzielten Simulationsergebnisse lässt sich zunächst Folgendes feststellen: Unter der Definition, dass laterale Abstände des Fahrzeugs zum Brückenpfeiler von $< 0{,}5\,\mathrm{m}$ als kritisch zu bezeichnen sind, gelingt es mit reiner Monte-Carlo-Simulation in 10.000 Simulationsläufen nicht, ein einziges kritisches Ereignis zu erzeugen. Folglich ist die Monte-Carlo-Simulation nicht mit vertretbarem Aufwand in der Lage, die Sicherheitsauswirkungen bedingt durch die Interaktion des Fahrers mit einem Assistenzsystem in einem typischen Szenario zu quantifizieren.
Ausgehend von der folgenden Signal-Temporal-Logic-ähnlichen [DM10] Formel (der Abstand zum Brückenpfeiler ist überall größer als 0,5 m)

$$\Box(||(x,y)-(p_x,p_y)|| > 0{,}5\,\mathrm{m}), \tag{4.14}$$

wobei x und y die aktuelle Längs- und Querposition des Fahrzeugs und p_x und p_y die entsprechenden Positionen der Brückenpfeiler darstellen, würde naives Statistisches-Model-Checking die Wahrscheinlichkeit einer Verletzung der Formel als Null schätzen. Die geführte Simulation hingegen verwendet eine kontinuierliche Interpretation [FH05; DM10] der Gleichung 4.14, nämlich das Minimum über die Zeit (aufgrund des $\Box$ Operators) von der Distanz zum Brückenpfeiler (aufgrund des Terms $||(x,y)-(p_x,p_y)||$) minus des – im Rahmen der Minimierung irrelevanten – Offsets von 0,5 m als kontinuierliche Zielfunktion, welche fortlaufend minimiert wird. Die Minimierung wurde beim TUTS-Algorithmus dadurch erreicht, dass wie beim Importance Sampling die Wahrscheinlichkeiten der probabilistischen Elemente des Fahrermodells verändert werden. Als Resultat dieser Veränderungen konnten

mit der TUTS-geführten Simulation (GS+RS-Konfiguration) in beinahe 10% der Simulationsläufe kritische Situationen beobachtet werden, welche es im Anschluss ermöglichen, eine aussagekräftige Statistik und damit eine quantitative Risikoaussage abzuleiten.

Um die Ergebnisse verschiedener Simulationstechniken (MCS, AIS) vergleichen zu können, müssen zunächst ihre Konfidenzintervalle (*Confidence Interval* (CI)) berechnet werden. Da die beiden Simulationstechniken auf zufälligen, voneinander unabhängigen Simulationsläufen (*Samples*) basieren, stellt ihr Ergebnis immer nur eine Schätzung dar. Im Folgenden wird daher auch der Begriff des „Schätzers“ verwendet, wenn eine Simulationstechnik gemeint ist.

Binomial Konfidenzintervall. Zur Berechnung des Konfidenzintervalls des naiven Schätzers (Monte-Carlo-Simulation) wird das einfache binomiale Konfidenzintervall verwendet, auch bekannt als das Clopper-Pearson-Konfidenzintervall oder das genaue Konfidenzintervall [CP34].

Bootstrap Konfidenzintervalle. Bei der Verwendung von Importance Sampling wird von einer ursprünglichen Binomialverteilung auf eine multinomiale Verteilung umgestellt, da einzelne Simulationsläufe (*samples*) nicht länger mit 0 (sicherer Simulationslauf) oder 1 (unsicherer Simulationslauf) bewertet werden, sondern mit einer Vielzahl von verschiedenen Gewichtungen (*importance factor*). Um eine Approximation des entsprechenden Konfidenzintervalls zu berechnen, werden das Bootstrap Konfidenzintervalle verwendet [ET93].

Um als Beispiel Bootstrap Konfidenzintervalle für einen Schätzer mit der Stichprobe $S_0 = (x_1, \ldots, x_n)$ zu berechnen, werden mehrere neue Bootstrap-Stichproben mit der gleichen Größe wie die Stichprobe S_0 erstellt, in dem aus der Menge $\{x_1, \ldots, x_n\}$ mit Zurücklegen gezogen wird. Durch das Zurücklegen können in den neu generierten Stichproben Duplikate enthalten sein, die Anzahl der Datenpunkte n ist dabei für jede Stichprobe identisch. Auf diesen neugenerierten Stichproben wird dann für jede Stichprobe einzeln die Variabilität berechnet.

Konkret seien $S_1, \ldots S_m$ neue Bootstrap-Stichproben, die durch Re-Sampling gewonnen wurden und sei f der Schätzer, der als Eingabe eine Stichprobe bekommt und als Ausgabe die zugehörige Schätzung liefert. Um ein $1-\alpha$ Konfidenzintervall zu erhalten, wobei α die Irrtumswahrscheinlichkeit beschreibt, werden die Ergebnisse $f(S_1), \ldots, f(S_m)$ vom niedrigsten zum höchsten Wert sortiert. Das Konfidenzintervall selbst ist dann durch $[f(S_l), f(S_u)]$ gegeben, wobei l und u Indizes entsprechend $(\alpha/2)m$ und $(1-\alpha/2)m$ der geordneten Liste sind.

Risikoabschätzung seltener Ereignisse: Monte-Carlo-Simulation versus TUTS-geführte Simulation

Im Evaluationsszenario trat ein seltenes Ereignis x_i immer dann auf, wenn der Abstand zwischen dem Fahrzeug und dem Brückenpfeiler eine bestimmte Entfernung unterschreitet. Die Wahrscheinlichkeit $\hat{p}$ des Ereignisses kann mit Hilfe des unbefangenen Importance Sampling-Schätzers berechnet werden (vgl. Gleichung 2.9):

$$\hat{p} \approx \frac{1}{N} \sum_{i=1}^{N} \left(\begin{cases} 1, & \text{wenn } x_i \text{ kritisch ist} \\ 0, & \text{sonst} \end{cases} \right) \frac{p(x_i)}{q(x_i)}, \quad x_i \sim q_i. \tag{4.15}$$

N repräsentiert dabei die Anzahl an Simulationsläufen, $p(x_i)$ die ursprüngliche Wahrscheinlichkeit und $q(x_i)$ die mit dem Faktor q gewichtete Wahrscheinlichkeit von x_i. Abbildung 4.19 zeigt die Wahrscheinlichkeit verschiedener Kritikalitäten (Annäherung an den Brückenpfeiler näher als eine bestimmte Entfernung) mit dem 99% Konfidenzintervall.

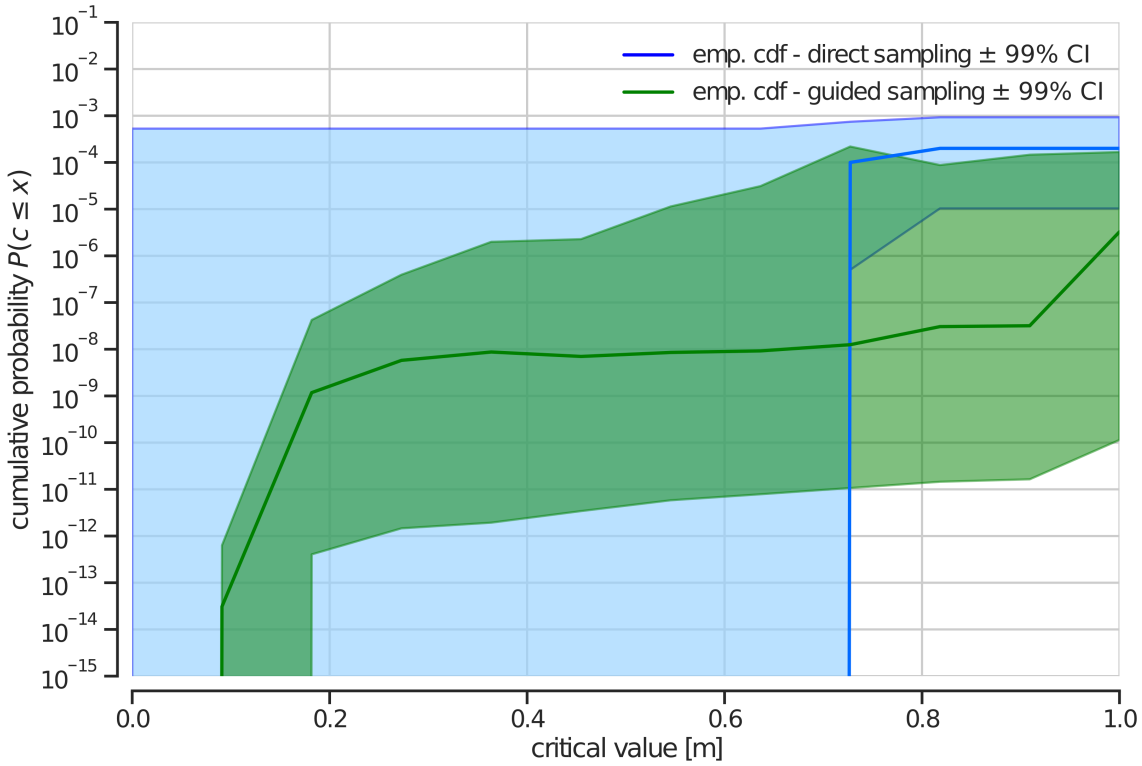

Abbildung 4.19: Geschätzte Wahrscheinlichkeiten mit 99% Konfidenzintervall für die Annäherung an den Brückenpfeiler näher als eine bestimmte kritische Entfernung. Blau: naive Monte-Carlo-Simulation, grün: TUTS-geführte Simulation [PFG18].

Auf der x-Achse ist der Abstand zwischen dem Fahrzeug und dem Brückenpfeiler aufgetragen, während die y-Achse die kumulierte Wahrscheinlichkeit darstellt, dass die Kritikalität kleiner oder gleich einem Abstand ist. Zu beachten ist, dass die y-Achse für eine bessere Darstellung sehr kleiner Wahrscheinlichkeiten eine logarithmische Skalierung verwendet. Die Ergebnisse wurden unabhängig voneinander mit Hilfe von MCS (blauer Graph) und TUTS-geführter Simulation (grüner Graph) berechnet. Letztere zeigt offensichtlich zwei signifikante Verbesserungen:

1. Eine deutlich bessere Beurteilung von kritischen Abständen unterhalb von 0,7 m, welche die seltenen Ereignisse in dem Szenario darstellen. Die kleinste Entfernung zum Brückenpfeiler wurde bei ca. 0,1 m mit einer geschätzten Wahrscheinlichkeit von $5 \cdot 10^{-13}$ und einem zugehörigen 99% Bootstrap Konfidenzintervall von $[10^{-15}, 10^{-11}]$ bei 2.000 Bootstrap-Stichproben aufgezeichnet. Im gleichen Bereich kann die MCS nur mit 99% Konfidenz die Aussage treffen, dass die Wahrscheinlichkeit unterhalb von $7 \cdot 10^{-4}$ liegt.
2. Eine engere obere Grenze für die Wahrscheinlichkeit, kritische Abstände über 0,7 m zu erreichen. Eine quantitative Sicherheitsspezifikation wie beispielsweise „die Wahrscheinlichkeit, dass sich das Fahrzeug dem Brückenpfeiler weniger als 1 m nähert, sollte höchstens 10^{-4} betragen“, kann von der TUTS-geführten Simulation mit 99%iger Sicherheit verifiziert werden, während die MCS ergebnislos bleibt. Zwar hat die MCS engere untere Grenzen als die geführte Simulation, diese sind für die quantitative Verifikation aber nutzlos: Sowohl die Akzeptanz als auch die Widerlegung quantitativer Sicherheitsziele hängen davon ab, ob der Schwellenwert durch die obere Grenze des Konfidenzintervalls überschritten wird.

Die quantitative Evaluation des TUTS-Algorithmus zeigt, dass sich basierend auf den qualitativen Simulationsergebnissen mittels statistischer Methoden auch Sicherheitsspezifikationen ableiten lassen. Durch die mathematische Rückrechnung der bei der TUTS-geführten Simulation veränderten Wahrscheinlichkeiten des Fahrermodells lassen sich auch im Nachhinein valide Aussagen über das originale Modell treffen. Die Gültigkeit der Ergebnisse wird dabei anhand der 99%igen Konfidenzintervalle angegeben. Wie bei allen statistischen Verfahren, die Aussagen über Wahrscheinlichkeiten treffen, bleibt ein gewisser Restfehler, im Sinne der Irrtumswahrscheinlichkeit, unvermeidlich bestehen.

4.4 Übertragbarkeit der Methodik auf andere Domänen

Bezugnehmend auf die Vision hinter der Forschungsfrage dieser Arbeit wurde eine Entwicklungs- und Testmethodik für einen Entwicklungsingenieur im Kontext sicherheitskritischer Fahrerassistenzsysteme beschrieben. Mit Hilfe eines Kosimulationsframeworks lassen sich

unterschiedliche Simulatoren, Entwicklungsumgebungen etc. zu einer Kosimulation zusammenschließen. Ein Simulationsguide realisiert mit dem TUTS-Algorithmus ermöglicht es, seltene, im gegebenen Kontext auch sicherheitskritische, Ereignisse sichtbar zu machen und anschließend eine quantitative Bewertung durchzuführen. Evaluiert wurde die Methodik anhand von zwei Beispielen aus dem Bereich Automotive. Im Grundsatz ist die vorgeschlagene Methodik aber domänenunabhängig. So wurde bei der Evaluation des Kosimulationsframeworks gezeigt, dass sich mit X-Plane ein Simulator aus dem Bereich der Luftfahrt in ein vergleichbares Beispiel integrieren lässt und mit der kognitiven Architektur CASCaS lassen sich dazu passend auch Pilotenmodelle entwickeln [FOL10; FOL11]. Der auf Adaptive Importance Sampling (AIS) basierende TUTS-Algorithmus benötigt zur Gewichtung von Simulationsläufen lediglich eine numerisch berechenbare Kritikalitätsfunktion, wobei die daraus resultierenden Werte mittels z-Transformation normiert werden, so dass sich konzeptbedingt sowohl kleine als auch große Maße einer beliebigen Domäne eignen. Wünschenswert wäre an dieser Stelle ein direkter Vergleich des TUTS-Algorithmus mit einer anderen AIS-Implementierung. Aufgrund der notwendigen technischen Integration von TUTS in die Implementierung von CASCaS, um die Vielzahl probabilistischer Prozesse in einer Kosimulation mit einem Realzeitfahrsimulator bewältigen zu können, sowie der Komplexität des Simulationsaufbaus für die vorgestellte Evaluation, war ein direkter Vergleich leider nicht möglich.

Stellvertretend soll an dieser Stelle noch ein artifizielles Beispiel eines zufällig springenden Balls vorgestellt werden, bei dem ein Vergleich von AIS-geführter Simulation mit der Kreuzentropie-Methode und TUTS sowie Monte-Carlo-Simulation durchgeführt wurde. Beide AIS-basierten Ansätze stützen sich auf Varianzreduktion und verwenden zugleich unterschiedliche Optimierungsprinzipien wie beispielsweise Bestärkendes Lernen (*reinforcement learning*) [SB98]. Da es bei einer quantitativen Systemverifikation mit Statistischem Modell Checking, insbesondere im Kontext einer Sicherheitsanalyse, zumeist darum geht, die Wahrscheinlichkeit einer Gefahr gegen ein Restrisiko durch einen Schwellwert θ abzuschätzen,

$$p(\text{Gefahr}) = \text{Restrisiko} \leq \theta \tag{4.16}$$

$$p(\text{Restrisiko}) \in \underbrace{[\ \overbrace{r-\varepsilon_l}^{\text{untere Grenze}}\ ,\ \overbrace{r+\varepsilon_u}^{\text{obere Grenze}}\ ]}_{\text{Konfidenzintervall}} \geq k \quad \text{mit } \varepsilon_l, \varepsilon_u > 0 \tag{4.17}$$

ist eine engere obere Grenze des Konfidenzintervalls zu einem Konfidenzlevel k, mit welcher die Wahrscheinlichkeit des Restrisikos gegen ein Risiko r abgeschätzt wurde, von höherer Relevanz als die Gesamtbreite des Intervalls (vgl. auch vorhergehender Abschnitt).

In Bezug auf Gleichung 4.17 sollte die Eigenschaft eines Schätzers daher das Merkmal $\varepsilon_l > \varepsilon_u$ aufweisen, was im Ergebnis zu einer rechtssteilen Verteilung führt. Grafisch veranschaulichen lässt sich die Eigenschaft einer engeren oberen Intervallgrenze anhand der

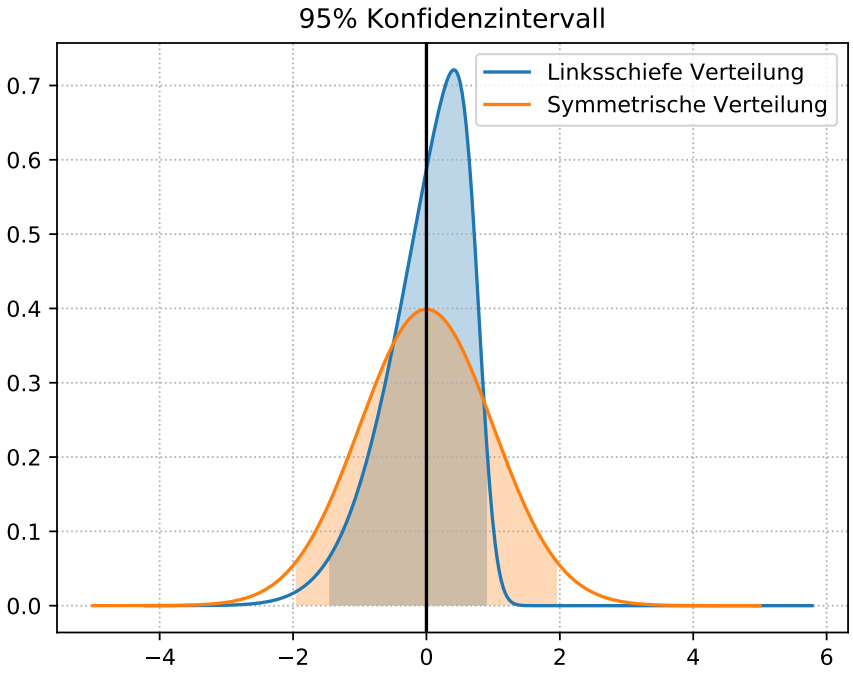

Abbildung 4.20: Vergleich einer linksschiefen mit einer nicht schiefen Verteilung.

schematischen Abbildung 4.20 daran, dass die blaue Verteilung eine steilere rechte Flanke aufweist und damit im Vergleich zur symmetrischen orangen Verteilung um den x-Wert 0 eine gewisse Schieflage hat. Eine negative Schieflage ($\gamma < 0$) wird dabei als „rechtssteil“ oder „linksschief“ bezeichnet und eine positive Schieflage ($\gamma > 0$) analog mit „linkssteil“ oder „rechtschief“. Für eine Schätzung von sehr kleinen Wahrscheinlichkeiten, wie es bei seltenen Ereignissen der Fall ist, liegt der Fokus daher auf dem Erzeugen einer linksschiefen Verteilung, so dass beim Vergleich der beiden geführten Simulationsansätze (AIS mit Cross-Entropy und TUTS) eine Bewertung anhand der Optimierung dieses Sachverhalts möglich ist.

Der zufällig springende Ball

Die Verwendung eines einfacheren Benchmarks, im Vergleich zu den bisherigen realitätsnahen, aber dadurch komplexeren Evaluationsszenarien, hat den Vorteil, dass die „*Ground Truth*“ (objektiv beweisbare Daten) bekannt ist und die Varianz entlang der probabilistischen Ereignisse berechnet werden kann. Der einfache, zufällig springende Ball fällt von einer anfänglichen Höhe aus nach unten auf eine reflektierende Oberfläche. Beim Aufprall auf den Boden, wird der Absprungwinkel des Balls zufällig aus einer festen Menge an Winkeln gezogen. Damit kann der Ball entlang einer Achse in einer festen Richtung, aber mit unterschied-

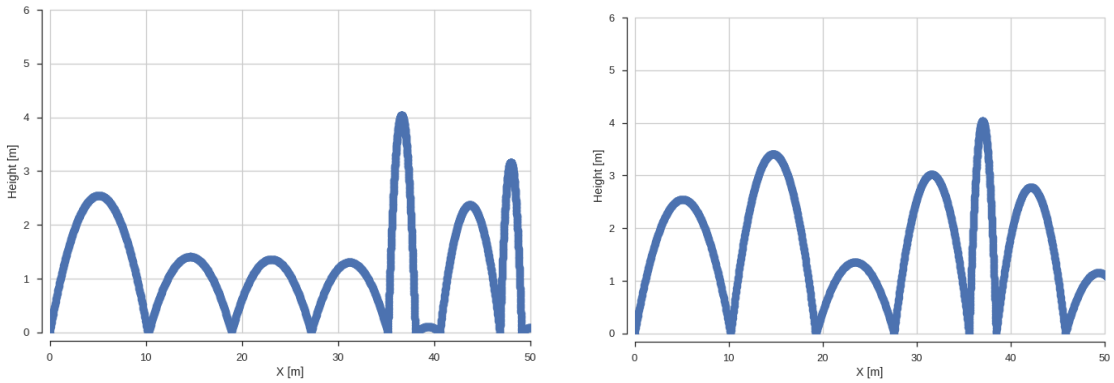

Abbildung 4.21: Zwei zufällige Trajektorien eines springenden Balls [PFG18].

lichen Höhen und horizontalen Geschwindigkeiten abspringen, die sich aus verschiedenen Absprungwinkeln ergeben, vergleiche Abbildung 4.21.

Die ballistische Kurve des Balls wird durch die folgende Gleichung definiert:

$$x(t) = x(0) + vt\cos(\theta) \quad y(t) = y(0) + vt\sin(\theta) - \frac{1}{2}gt^2, \tag{4.18}$$

wobei θ und v durch den initialen Geschwindigkeitsvektor $\vec{v}_0$ wie folgt gegeben sind:

$$\theta = \arctan(\vec{v}_0) \qquad v = ||\vec{v}_0||_2 \tag{4.19}$$

Anhand des initialen Anfangsgeschwindigkeitsvektors und der Position kann berechnet werden, wann der Ball das nächste Mal auf den Boden trifft ($y(t) = 0$). Beim Aufprall auf den Boden wird dann das Vorzeichen der y-Koordinate des Geschwindigkeitsvektors umgekehrt, die Geschwindigkeit selbst wird mit dem Faktor ρ gedämpft und, um eine unregelmäßige Oberfläche zu modellieren, unterliegt der resultierende Absprungwinkel einer zufälligen Störung. Genau genommen ist der Geschwindigkeitsvektor zu einem beliebigen Zeitpunkt wie folgt gegeben:

$$\vec{v}(t) = \frac{\partial(x(t), y(t))}{\partial t} = (v\cos(\theta), v\sin(\theta) - gt) \tag{4.20}$$

Der nächste Zeitpunkt t_{n+1}, zu dem der Ball auf den Boden trifft, lässt sich durch das Setzen der y-Koordinate auf Null berechnen

$$t_{n+1} = \sqrt{\left(\frac{\sin(\theta)v}{g}\right)^2 + \frac{2y(t_n)}{g}} + \frac{v\sin\theta}{g} \tag{4.21}$$

und das Sprungverhalten einschließlich einer Dämpfung sowie die zufällige Störung des Absprungwinkels zum nächsten Zeitpunkt t_{n+1}, wird durch Setzen der Geschwindigkeit und des Winkels folgendermaßen umgesetzt:

$$||v(t_{n+1})||_2 = ||\vec{v}(t_n)||_2\rho \qquad \theta(t_{n+1}) = \eta \tag{4.22}$$

Der Einfachheit halber wird die zufällige Störung η gleichverteilt aus einer vordefinierten Liste gezogen: $\eta \sim \mathscr{U}\{\eta_1, \ldots, \eta_m\}$.

Das seltene Ereignis schließlich, an dessen Wahrscheinlichkeitsschätzung die drei Samplingverfahren verglichen werden, besteht aus einem sehr kleinen Bereich der x-Koordinate und der Höhe 0 in der Oberfläche (stellvertretend für ein Loch), welcher mit dem Ball getroffen werden muss. Der kleine Bereich ist dabei so festgelegt, dass die Wahrscheinlichkeit eines Treffers genau einer einzigen definierten Abfolge von Absprungwinkeln mit einer begrenzten maximalen Anzahl von sieben Sprüngen entspricht. Das Ziel der Samplingverfahren besteht darin, möglichst häufig mit dem springenden Ball das Loch in der Oberfläche zu treffen und dabei eine sichere Abschätzung der Trefferwahrscheinlichkeit zu ermitteln. Im Gegensatz zur MCS haben die beiden AIS-basierten Verfahren die Möglichkeit, die im Modell gleichverteilte Wahrscheinlichkeit der Absprungwinkel sukzessiv zu verändern.

Umsetzung des Benchmarks

Die Umsetzung[3] des zufällig springenden Balls erfolgte in der Programmiersprache Python in der Version 2.7.13 und die Simulation wurde auf einem Rechenserver mit 64 CPUs (Kernen) und 528 Gigabyte Arbeitsspeicher ausgeführt. Für die Funktionsweise des TUTS-Algorithmus sei an dieser Stelle auf die ausführlichen Beschreibungen in Abschnitt 4.2 ab Seite 62 verwiesen, da sich in Unterabschnitt 2.5.2 auf Seite 23 aber lediglich eine abstrakte Beschreibung der Kreuzentropie-Methode befindet, wird an dieser Stelle ihre Anwendung im Rahmen des Benchmarks noch einmal im Detail dargelegt.

Die Cross-Entropy [Rub99] ist eine Methode, die Adaptive Importance Sampling benutzt, um die aktuelle Vorschlagsdichtefunktion (gekennzeichnet durch ihre Dichte oder Wahrscheinlichkeitsfunktion q) so anzupassen, dass sie gegen die optimale Vorschlagsdichtefunktion konvergiert. Optimal bedeutet dabei, dass eine einzige Stichprobe ausreicht, um den Erwartungswert eines Ereignisses genau abzuschätzen. Der daraus resultierende Schätzer weist daher eine Nullvarianz auf. Da diese optimale Vorhersage allerdings nicht verfügbar ist, schätzt die Kreuzentropie-Methode die optimale Vorhersage basierend auf den bereits gezogenen Stichproben. Um die Nähe der aktuellen Vorhersage zur Schätzung zu berechnen, wird der „Kullback-Leibler-Divergenz“ verwendet, die Methode wird als AIS mit Cross-Entropy bezeichnet. Ihre Anwendung auf den zufällig springenden Ball lässt sich wie folgt veranschaulichen.

Es sei p_i die Wahrscheinlichkeit, um im Modell des springenden Balls den i-ten Winkel η_i zu ziehen. In dem stochastischen Modell des springenden Balls sind die Winkel über Zeitpunkte hinweg unabhängig. Daher kann die Wahrscheinlichkeit einer Abfolge von mehreren Winkeln $x_t, t = 1 \ldots, T$ durch das Produkt $\prod_t \sum_i p_i \delta(x_t, \eta_i)$ beschrieben werden, wobei δ das

[3] Der Quellcode ist unter https://uol.de/hs/downloads verfügbar.

Kronecker-Delta bezeichnet, welches zu 1 ausgewertet wird, wenn $x_t = \eta_i$ ist und 0 sonst. Ähnlich wird q ausgewählt, um die Wahrscheinlichkeit darzustellen, dass verschiedene Winkel gezogen werden. Da ein seltenes Ereignis als Verkettung von zufälligen Ereignissen über die Zeit entsteht, wäre es vorteilhaft, zwischenzeitliche Abhängigkeiten innerhalb der Vorschlagsdichtefunktion zu erlauben. Da dies jedoch die Anzahl der Parameter exponentiell erhöht, wird auch eine Unabhängigkeitsannahme für die Vorschlagsdichtefunktion verwendet. Konkret wird zur analytischen Berechnung des Kreuzentropie-Updates die folgende Parametrisierung von q benutzt:

$$q(x) = \prod_i \frac{\exp(\gamma_i \delta(x, \eta_i))}{\sum_k \exp(\gamma_k)} = \frac{\exp(\sum_i \gamma_i \delta(x, \eta_i))}{\sum_k \exp(\gamma_k)} \tag{4.23}$$

Die Wahrscheinlichkeit, einen bestimmten Winkel $x \in \{\eta_1, \ldots \eta_m\}$ zu ziehen, kann über die Auswahl unterschiedlicher Werte für γ reguliert werden. γ kann als natürlicher Parameter C mit $\delta(x, \eta_i)$ als suffiziente Statistik interpretiert werden, die es ermöglicht, auf einfache Weise Updates der Parameter γ zu berechnen, siehe Gleichung 4.25.

Im ersten Schritt werden N_0 Simulationsläufe mit der aktuellen Vorschlagsdichtefunktion q^n durchgeführt. Hier ist N_0 ein freier Parameter des Algorithmus, der als Batch-Size bezeichnet wird. Jedem dieser Simulationsläufe wird dabei ein Kritikalitätswert c_i zugeordnet. Für das Beispiel des zufällig springenden Balls benutzen sowohl TUTS als auch die Cross-Entropy als Kritikalitätsfunktion den euklidischen Abstand zwischen dem Vektor der gezogenen Winkel und dem (bekannten) Vektor derjenigen Winkel, mit denen der Ball in den kleinen Bereich mit dem Loch treffen würde – was das seltene Ereignis darstellt. Unter Verwendung dieser Kritikalität wählt die Kreuzentropie-Methode nun die α-kritischsten Simulationsläufe aus, das heißt den Indexsatz

$$I_\alpha := \{i : |\{j : c_j < c_i\}| \leq \alpha N_0\}. \tag{4.24}$$

Dieser Indexsatz wird wiederum verwendet, um die optimale Vorschlagsdichtefunktion q^* zu schätzen, also diejenige, welche zu einem Null-Varianz-Schätzer[4] führen würde. Aufgrund der exponentiellen Familienform der Darstellung in Gleichung 4.23 müssen lediglich die empirischen Mittel der suffizienten Statistiken berechnet werden (Gleichung 4.25), um neue Parameter γ für eine aktualisierte Vorschlagsdichtefunktion zu erhalten. Dies ist wiederum darauf zurückzuführen, dass ein Momentenabgleich (*moment matching*) gleichbedeutend ist mit der Minimierung der Kullback-Leibler-Divergenz zwischen der empirischen Null-Varianz-Verteilung und der Vorschlagsdichtefunktion über verschiedene Parametereinstellungen der benutzen Vorschlagsdichtefunktion, siehe [Rub99].

[4]Hinweis: Aufgrund der endlichen Anzahl der verwendeten Simulationsläufe ist dies nur eine Annäherung.

Die neuen Parameter γ^{n+1} der Vorschlagsdichtefunktion lassen sich also bestimmen, indem die empirischen Durchschnittswerte der suffizienten Statistiken berechnet werden, wobei die Umgewichtung gemäß der aktuellen Vorschlagsdichtefunktion q^n berücksichtigt werden muss.

$$\gamma_i^{n+1} = \frac{1}{|I_\alpha|} \sum_{k \in I_\alpha} \frac{1}{T_k} \sum_{t}^{T_k} \frac{p(x_t^k)}{q^n(x_t^k)} \delta(x_t^k, \eta_i) \tag{4.25}$$

Dabei bezeichnet x_t^k die (zufällige) Auswahl zum Zeitpunkt t innerhalb des k-ten Simulationslaufes eines Batches. Zu beachten ist dabei, dass die innere Summe $\sum_t$ benutzt werden kann, da Unabhängigkeit angenommen wurde und daher jede Auswahl entlang des k-ten Simulationslaufes gleichbehandelt wird. Mit neuen Parametern und der damit einhergehenden neuen Vorschlagsdichtefunktion q^{n+1} kann ein neuer Batch an Simulationsläufen der Größe N_0 generiert werden.

Durch das Anwenden dieses Updates erfassen die Parameter die Häufigkeiten, mit denen verschiedene Auswahlen eines Absprungwinkels η innerhalb der α kritischsten Simulationsläufe eines Batches auftreten. Diese Informationen werden wiederum verwendet, um die entsprechenden Absprungwinkel innerhalb des nächsten Batches häufiger auszuwählen. Allerdings werden, selbst wenn bestimmte Folgen von Absprungwinkeln bedingt durch die exponentielle Struktur nicht beobachtet werden, die zugehörigen Wahrscheinlichkeiten niemals auf Null gesetzt. Folglich kann mit dieser Parametrisierung durch das Ignorieren bestimmter Absprungwinkel auch keine Konvergenz zu einer optimalen Vorschlagsdichtefunktion erfolgen, es sei denn, es werden unbeschränkt große Batches benutzt.

Ergebnisse des Benchmarks

Bei dem Vergleich der drei Algorithmen MCS, AIS mit Cross-Entropy und TUTS-geführte Simulation wurden jeweils 10.000 Batchläufe mit 1.000 Samples durchgeführt. Metaphorisch dargestellt konnte jeder Algorithmus mit 1.000 Ballwürfen versuchen, das Loch zu treffen und aus dem Resultat zu lernen. Anschließend wurde die Anzahl der Treffer notiert. Jedem Algorithmus standen 10.000 Wiederholungen zur Verfügung, wobei jede Wiederholung mit einem Gedächtnisverlust des zuvor gelernten Verhaltens einherging. In Tabelle 4.1 ist in der zweiten Spalte ersichtlich, dass die Spanne an Batchläufen mit mindestens einem Treffer aus 10.000 Wiederholung von etwa 6,5% bei Monte-Carlo-Simulation über 39,1% für AIS mit Cross-Entropy bis zu 48,4% bei der TUTS-geführten Simulation reicht. Bei einem direkten Vergleich der Gesamtzahl an Treffern in der dritten Spalte wird die Unterlegenheit der Monte-Carlo-Simulation zu den beiden anderen Verfahren deutlich sichtbar. Während erstere mit etwas mehr als einem Treffer pro erfolgreichem Batchlauf nahezu unverändert bleibt, trifft der Ball bei der TUTS-geführten Simulation bei jedem erfolgreichen Batchlauf im Durchschnitt fünfmal das Ziel.

Tabelle 4.1: Anzahl der Treffer des springenden Balls ins Loch bei 10.000 Batchläufen mit jeweils 1.000 Würfen (Samples).

Algorithmus	Batches mit ≥ 1 Treffer	Gesamtzahl der Treffer
Naive Monte-Carlo-Simulation	648	671
AIS mit Cross-Entropy	3.909	5.452
TUTS-geführte Simulation	4.840	25.867

Die Ursache für diese deutliche Verbesserung kann Abbildung 4.22 entnommen werden. Auf der x-Achse sind die möglichen Absprungwinkel η des Balls aufgetragen und deren Häufigkeit kann an der y-Achse abgelesen werden. Wie bei naiver MCS zu erwarten sind die einzelnen Absprungwinkel gleichverteilt. Die beiden AIS-basierten Verfahren hingegen zeigen mit ihrer jeweiligen Strategie eine deutliche Häufung des größten Absprungwinkels, dessen siebenmalige Hintereinanderausführung den einzigen Weg darstellt, das Loch bei einer exakten Wahrscheinlichkeit von 0,000064 ($6,4 \cdot 10^{-5}$) zu treffen.

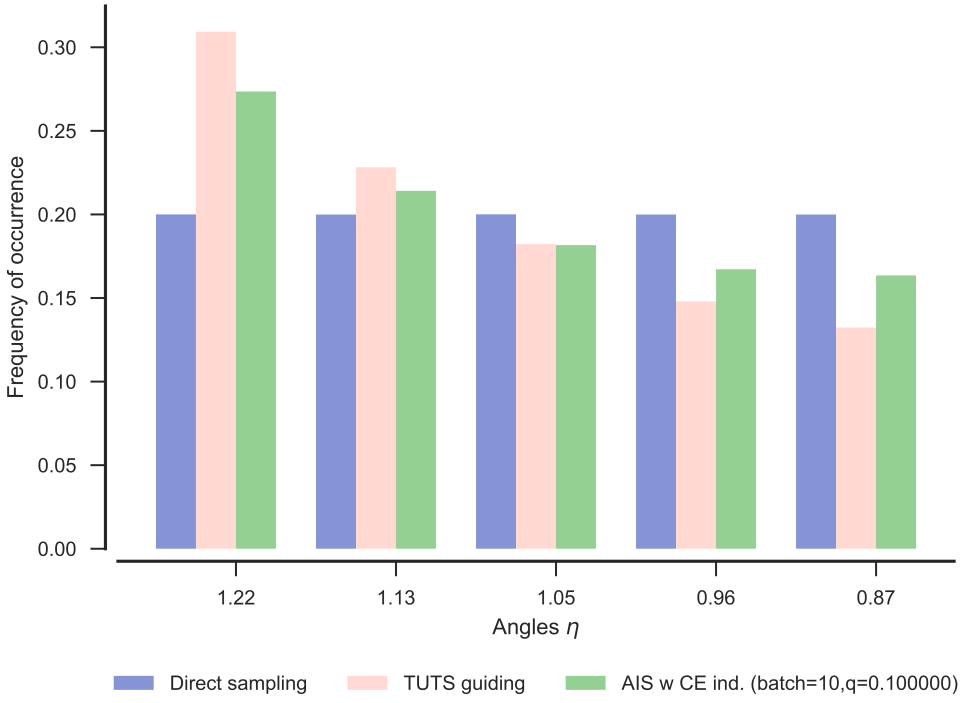

Abbildung 4.22: Histogramm der benutzen Absprungwinkel η bei 1.000 Ballwürfen und 10.000 Batchläufen [PFG18].

Die höhere Trefferquote bei entsprechend geringerem Gewicht jedes einzelnen Treffers erzeugt einen steileren Anstieg der Verteilung der Testergebnisse um die wahre Wahrscheinlichkeit wie in Abbildung 4.23 dargestellt. Diese Grafik zeigt für jeden Algorithmus die Häufigkeit (gezählt über 10.000 unabhängige Batchläufe mit jeweils 1.500 Simulationsläufen), eine Trefferwahrscheinlichkeit des springenden Balls unterhalb des auf der x-Achse angegebenen Schwellenwerts zu vermelden. Bedingt durch die Quantisierung ist MCS unfähig, eine positive Wahrscheinlichkeit von weniger als $\frac{1}{1500}$ zu berechnen. Da die tatsächliche Trefferwahrscheinlichkeit $6,4 \cdot 10^{-5}$ deutlich geringer ist als $\frac{1}{1500}$ (markiert durch die rote senkrechte Linie), ist es glaubhaft, dass naive MCS eine massive Unterschätzung von 0 in ca. $91,2\%$ der Simulationsläufe meldet. Die beiden auf IS-basierten Verfahren hingegen (zur besseren Differenzierung hier kurz AIS bzw. TUTS genannt), können Wahrscheinlichkeitsschätzungen nahe der tatsächlichen Wahrscheinlichkeit liefern und sind somit informativer. Bei einer kleinen Anzahl an Simulationsläufen pro Batch (≤ 2.000 pro Batch), ist es bei TUTS deutlich weniger wahrscheinlich, eine erhebliche Unterschätzung unter 0,000042 zu erzeugen, was die False-Positive-Rate bei Verwendung von Akzeptanzschwellen in diesem Bereich reduziert. TUTS hat auch die höchste Wahrscheinlichkeit, eine relativ genaue Näherung zu erzeugen: Etwa 3.340 Schätzungen von TUTS fallen in den Bereich von $\pm 25\%$ um die wahre Wahrscheinlichkeit, während AIS nur 2.065 innerhalb dieses Bereichs erreicht (und naive Monte-Carlo-Simulation keine).

Wie zu erwarten ist, sinkt die Wahrscheinlichkeit einer massiven Unterschätzung durch AIS, wenn AIS durch die Erhöhung der Simulationsläufe pro Batch deutlich mehr Zeit für Anpassungen der Parameter zu Verfügung gestellt wird. Beim Benchmark des springenden Balls liegt die Grenze bei einer Batchgröße oberhalb von 2.000 Simulationsläufen. In diesem Bereich beginnt AIS, TUTS in Bezug auf die Anzahl der massiven Unterschätzungen zu übertreffen, obwohl TUTS weiterhin die steilste Kurve um die tatsächliche Wahrscheinlichkeit liefert, vergleiche Abbildung 4.24. Da die seltenen Ereignisse in dem realitätsnäheren Szenario mit einem Assistenzsystem (siehe Unterabschnitt 4.3.3 auf Seite 78) um mehrere Größenordnungen seltener sind als beim Benchmark des springenden Balls, bleibt jedoch unklar, ob die entsprechenden Batchgrößen, welche eine Konvergenz von AIS gewährleisten, eine praktikable Option wären. Die anfänglich schnellere Konvergenz von TUTS gegen die tatsächliche Wahrscheinlichkeit im Regime $n \ll \frac{1}{p}$, wobei n die Anzahl der Simulationsläufe pro Batch und p das tatsächlich zu schätzende Risiko darstellt, bietet ein interessantes Merkmal, das in weitergehenden Arbeiten untersucht werden sollte.

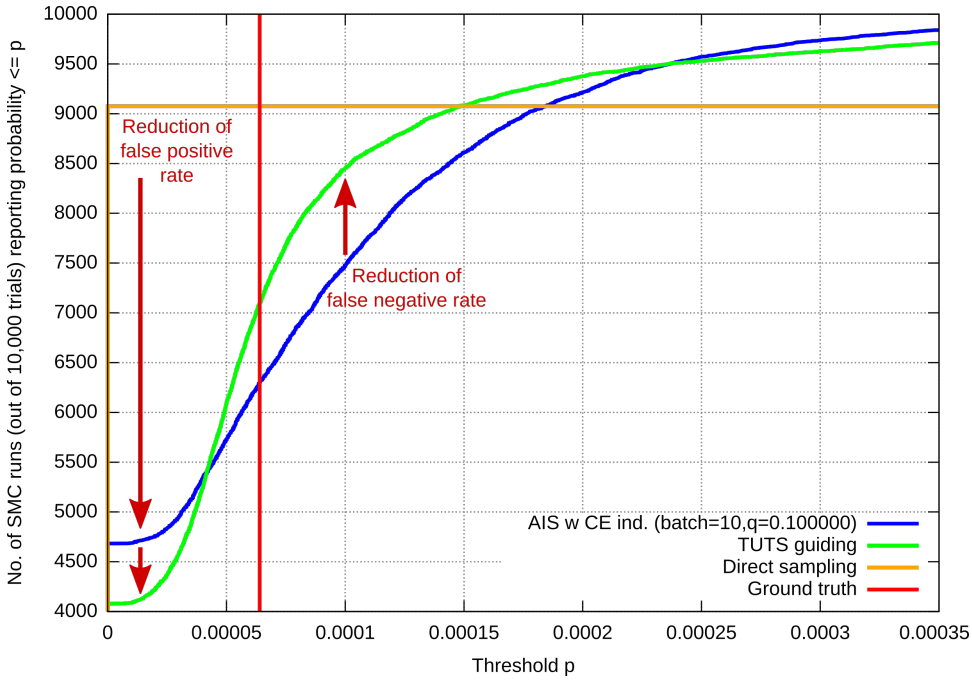

Abbildung 4.23: Die Häufigkeit einer Trefferwahrscheinlichkeit von höchstens p gezählt über 10.000 Batchläufe mit jeweils 1.500 Ballwürfen [PFG18].

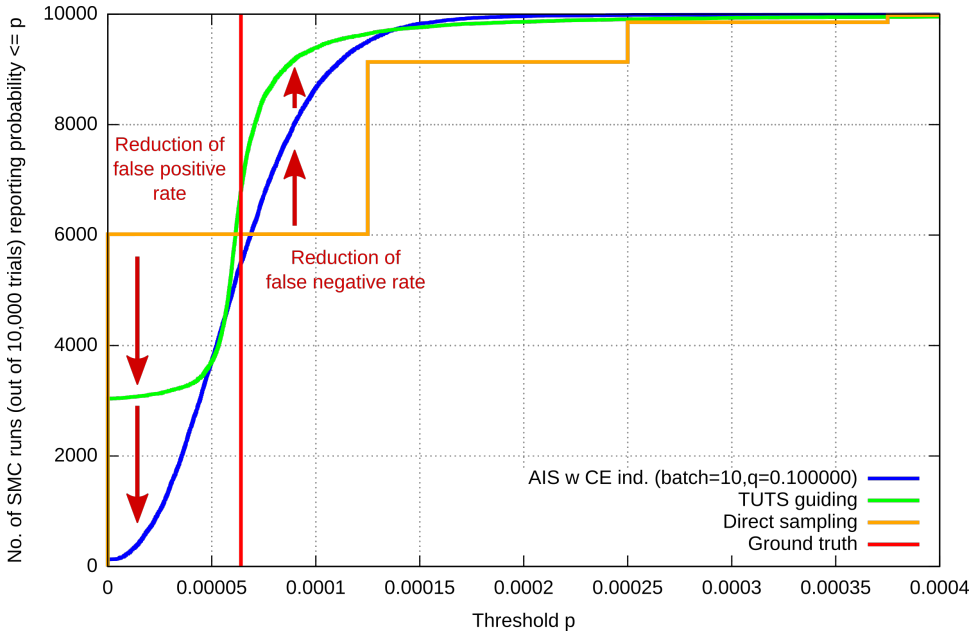

Abbildung 4.24: Die Häufigkeit einer Trefferwahrscheinlichkeit von höchstens p gezählt über 10.000 Batchläufe mit jeweils 8.000 Ballwürfen.

4.5 Einschränkungen und Grenzen

Nach der Evaluation des TUTS-Algorithmus sowie einer Beschreibung der Übertragbarkeit der geführten Simulation auf andere Domänen sollen in diesem Unterabschnitt kritisch die Einschränkungen und Grenzen des Ansatzes herausgestellt werden.

Der in Abschnitt 4.2 beschriebene TUTS-Algorithmus nutzt für die Bewertung eines Simulationslaufes eine Kritikalitätsfunktion, welche genutzt wird, um die Auswahlwahrscheinlichkeiten der probabilistischen Elemente in einer Simulation so zu verändern, dass in nachfolgenden Simulationsläufen diejenigen Elemente bevorzugt werden, die zu kritischerem Verhalten führen. Die Definition einer zur Simulationsführung geeigneten Funktion erfolgt anhand von Expertenwissen. Sie kann für jedes Simulationsszenario angepasst werden und sollte eine möglichst gute Unterscheidung im Hinblick auf das Simulationsziel in wünschenswerte bzw. irrelevante Simulationsläufe ermöglichen. Zwar ist die einzige an die Funktion gestellte Bedingung ihre numerische Berechenbarkeit zur Laufzeit der Simulation, dennoch bedarf es je nach Domäne einer Fachperson, welche die entsprechende Spezifikation übernimmt.

Eine Einschränkung hat die vorgestellte Methode der geführten Simulation hinsichtlich des Verfahrens zur Optimierung der Simulationszeit. Bei der Implementierung des TUTS-Algorithmus wurde eine iterativ anpassbare Form von Importance Sampling verwendet, da etablierte commercial off-the-shelf-Fahrsimulationssoftware wie beispielsweise STSoftware, SILAB und Virtual Test Drive nur einzelne definierte Aufsatzpunkte für den Start einer Simulation zulassen. Dies hat den Nachteil, dass die Simulationszeit, welche vom Start an einem Aufsatzpunkt bis zum Herstellen der zu analysierenden Fahrsituation notwendig ist, bei jedem Simulationslauf verschwendet wird. Bei einer auf Importance Splitting basierten Implementierung in Kombination mit Simulationskomponenten, die sich beliebig instanziieren lassen, könnte dieser Mehraufwand eingespart werden, indem ein Splitting Point (neuer Aufsatzpunkt), direkt hinter die – aus Sicht der geführten Simulation – nutzlose Anfangsphase gesetzt wird. Es existiert in dieser Hinsicht die Notwendigkeit einer Kompromissfindung zwischen voller Flexibilität hinsichtlich verwendbarer Simulationskomponenten und nutzbarer methodischer Verfahren zur Simulationsoptimierung.

Eine Grenze der Methode wurde beim Vergleich einer geführten Simulation mit dem TUTS-Algorithmus und der Cross-Entropy anhand des zufällig springenden Balls aufgezeigt. So lieferte TUTS bei einer kleinen Anzahl an Simulationsläufen pro Batch zwar bessere Ergebnisse, bei einer Anzahl von mehr als 2.000 Simulationsläufen hingegen in Teilen schlechtere. Dieser Übergangspunkt liegt allerdings weit unter der SMC-mäßig notwendigen Anzahl an Simulationsläufen, die eine Abschätzung nach Hoeffding ausweist (vergleiche Gleichung 2.8) und liefert bei viel kleineren Samplezahlen eine brauchbare Schätzung. Das Resultat ist aus Sicht des Verfassers eine Bestätigung des „No Free Lunch Theorems for Opti-

mization“ [WM97], denn es veranschaulicht, dass ein Kompromiss zwischen der Anzahl an Simulationsläufen und benötigter Konfidenz gefunden werden muss.

Zum Schluss sollte bei Anwendung einer geführten Simulation zur Überprüfung eines Fahrerassistenzsystems beachtet werden, dass sich die Methodik hinsichtlich ihrer Reihenfolge nicht brauchbar umkehren lässt: So können die mit dem TUTS-Algorithmus erhobenen Daten eines Batches durch anschließende mathematische Rückrechnung für eine Aussage – in Form einer Schätzung – über das Originalmodell genutzt werden und mittels Statistik (beispielsweise Bootstrapping) ist zusätzlich die Bestimmung einer Konfidenz möglich. Die umgekehrte Abfolge, bei der ausgehend von einem notwendigen Konfidenzniveau bestimmt wird, wie viele geführte Simulationsläufe innerhalb eines Batches durchzuführen sind, scheitert hingegen an der statistischen Unabhängigkeit der einzelnen Simulationsläufe, wie sie beispielsweise für eine Abschätzung mit Hoeffdings Ungleichung gefordert ist (vgl. Abschnitt 2.4 auf Seite 17). Eine Abschätzung mit Hoeffdings Ungleichung erlaubt höchstens eine Abschätzung, wie viele Batches im worst case notwendig sind (vgl. Beispielrechnung in Gleichung 2.8 auf Seite 19). Ein sinnvoller Ablauf in einem aufwändigen Simulationsszenario wäre daher ein iteratives Vorgehen, bei dem die TUTS-geführte Simulation beispielsweise nach 10.000 Simulationsläufen pausiert und nach einer Rückrechnung der Zwischenergebnisse mittels Bootstrapping überprüft wird, ob die erreichte Konfidenz bereits hinreichend ist. Im negativen Fall kann die Simulation ohne Restriktion fortgesetzt werden, bei einem ausreichenden Ergebnis kann die geführte Simulation beendet werden.

4.6 Zusammenfassung

Für das Konzept der geführten Simulation wurde der TUTS-Algorithmus vorgestellt. Seine Implementierung basiert auf Adaptive Importance Sampling und nutzt zur Führung der Simulation eine Kritikalitätsfunktion, welche jeden Simulationslauf im Hinblick auf ein angestrebtes Ereignis bewertet. Damit die Kritikalitätsfunktion sowohl hohe als auch niedrige Kenngrößen verwenden kann, werden ihre Ergebnisse mit Hilfe der z-Transformation normalisiert, bevor sie in die steuernde Gewichtungsfunktion von TUTS einbezogen werden. In einem realistischen Anwendungsszenario konnte gezeigt werden, dass der TUTS-Algorithmus eine Kosimulation – bestehend aus Fahrermodell, Fahrsimulationssoftware und Fahrerassistenzsystem – in seltene kritische Situationen führen kann. Im Gegensatz zu naiver Monte-Carlo-Simulation konnten dabei in 9% von 10.000 Simulationsläufen kritische Situationen mit einer Wahrscheinlichkeit $< 10^{-8}$ aufgezeigt werden. Derart kleine Wahrscheinlichkeiten sind auch im Kontext sicherheitskritischer Fahrerassistenzsysteme anzutreffen, so dass auf diese Weise eine Antwort auf die zweite der Forschungsteilfragen gegeben wurde.

Anschließend wurden die simulativ erreichten Ergebnisse hinsichtlich ihrer Auftretenswahrscheinlichkeit mit dem 99% Konfidenzintervall abgeschätzt. Zur Abschätzung des Resultats

der geführten Simulation wurden auf Resampling basierende Bootstrap Konfidenzintervalle verwendet, wohingegen bei der Monte-Carlo-Simulation das genaue Konfidenzintervall (Binomial Konfidenzintervall) angewendet werden konnte. Während das kritischste Ereignis der MCS – mit einer Wahrscheinlichkeit von 10^{-4} – lediglich mit einer 99%igen Konfidenz unterhalb von $7 \cdot 10^{-4}$ abgesichert werden konnte, weist die geführte Simulation bei gleicher Konfidenz eine engere obere Grenze von $4 \cdot 10^{-4}$ aus. Zwar kann die Monte-Carlo-Simulation bei der Abschätzung der unteren Grenze ein engeres Intervall im Vergleich zur geführten Simulation vorweisen, dieses ist aber sowohl für die Akzeptanz als auch für die Widerlegung quantitativer Sicherheitsziele, wie beispielsweise der Festsetzung eines Restrisikos, nutzlos. Darüber hinaus konnte die TUTS-geführte Simulation das kritischste Ereignis – mit einer Wahrscheinlichkeit von $5 \cdot 10^{-13}$ – immerhin noch mit einem zugehörigen 99% Konfidenzintervall von $[10^{-15}, 10^{-11}]$ abschätzen, womit auch die dritte der Forschungsteilfragen beantwortet ist.

Zur Beantwortung der vierten Forschungsteilfrage wurde gezeigt, dass die Methodik einer geführten Simulation konzeptbedingt unabhängig von der Domäne ist. Anhand eines Benchmarks wurden die Ergebnisse zur Schätzung eines seltenen Ereignisses durch Monte-Carlo-Simulation mit zwei verschiedenen Algorithmen – TUTS und Cross-Entropy – für eine geführte Simulation verglichen. Wie zu erwarten können beide geführten Verfahren eine Wahrscheinlichkeitsschätzung nahe der tatsächlichen Wahrscheinlichkeit von $6,4 \cdot 10^{-5}$ liefern, wohingegen MCS in über 90% der Fälle mit 0 eine massive Unterschätzung und andernfalls mit $6,\overline{6} \cdot 10^{-4}$ eine zu große Wahrscheinlichkeit berichtet.

Kapitel 5

Schlussfolgerungen

Die Arbeit begann mit der Feststellung, dass auch im Jahr 2017 noch durchschnittlich 9 Menschen täglich im deutschen Straßenverkehr ums Leben kommen und damit verbunden ein enormer volkswirtschaftlicher Schaden entsteht. Als eine Ursache wurde der Faktor Mensch identifiziert, der sich in seinem Fahrzeug – bei kontinuierlich zunehmendem Verkehrsaufkommen – in einer hochkomplexen Umgebung befindet, die zu Fehlern führt. Die Automobilindustrie, welche mit Hilfe von Fahrerassistenzsystemen versucht, den Fahrer bei seinen zahlreichen Aufgabe zu unterstützen, steht der Problematik gegenüber, dass ein Unfall mit Personenschaden auf den Einzelfall betrachtet ein extrem seltenes Ereignis darstellt. Die einfache simulative Analyse eines Fahrerassistenzsystems, welches bei den seltenen aber kritischen Situation zum Einsatz kommen soll, in Kombination mit einem Fahrermodell und einer Fahrsimulationssoftware, benötigt zu viel Simulationszeit. So wurde aus dem Bedarf der Automobilindustrie nach speziellen Entwicklungs- und Testabläufen im Bereich der Fahrerassistenzsystementwicklung die Forschungsfrage abgeleitet, ob sich Methoden aus dem Bereich des Statistischen Model Checkings im Rahmen von geführter Simulation nutzbar machen lassen, um die modellbasierte Entwicklung von sicherheitskritischen Fahrerassistenzsystemen, hinsichtlich ihrer angedachten Funktionsweise, in einem möglichst realistischen Anwendungsszenario unterstützen zu können.

5.1 Zusammenfassung

Die in der Arbeit erreichten Ergebnisse wurden im Hinblick auf eine fließende Darstellung und eine gute Lesbarkeit jeweils an den Stellen präsentiert, wo sie am geeignetsten erscheinen. Daher werden im Folgenden die wichtigsten Ergebnisse noch einmal zusammengefasst.

Für die Beantwortung der Forschungsfrage wurden vier relevante Teilfragen identifiziert, anhand derer sich die Arbeit in aufeinander aufbauenden Einzelschritten strukturierte. Um einem Entwicklungsingenieur für Fahrerassistenzsysteme ein modellbasiertes Entwickeln

und simulatives Evaluieren in einem vom ihm präferierten Werkzeug anzubieten, wurde im ersten Schritt ein Kosimulationsframework entwickelt, welches die schnelle Integration domänentypischer Simulatoren in eine Kosimulation ermöglicht. Für die technische Umsetzung wurde mit der High Level Architecture ein Standard verwendet, der mit einer Runtime Infrastruktur nicht nur den Datenaustausch koordiniert, sondern auch einen synchronisierten Simulationszeitfortschritt aller Simulationsteilnehmer ohne Einschränkung der jeweiligen Performanz sicherstellt. Das Framework wurde zunächst mit zwei verschiedenen RTI-Implementierungen hinsichtlich ausreichender Performanz evaluiert und die Anwendbarkeit mit der kognitiven Architektur CASCaS – als Grundlage für ein Fahrer- bzw. Pilotenmodell – in Kombination mit zwei Realzeitsimulatoren – je einmal aus dem Bereich Automotiv und Aeronautik – demonstriert. Anschließend wurde auf Basis des Kosimulationsframeworks die Batchevaluation eines Fahrerassistenzsystems in Kombination mit einem Onlinemonitor, einem kognitiven Fahrermodell und einer Fahrsimulationssoftware durchgeführt.

Der zweite Schritt setzt sich mit der Problemstellung auseinander, seltene Ereignisse in der Größenordnung von 10^{-6} – welche aus den statistischen Zahlen von Unfällen mit Personenschaden abgeleitet wurde – im Rahmen einer Kosimulation simulativ zu erfassen. Auf diese Weise soll ein Fahrerassistenzsystem auch in sehr seltenen aber kritischen Situationen evaluiert werden können. Da der Zustandsraum für eine vollständige simulative Betrachtung zu groß ist, wurde die Methode des Statistischen Model Checkings eingeführt, welche Sampling mit statistischer Analyse kombiniert. Mit Hilfe des TUTS-Algorithmus, welcher mit einem auf Adaptive Importance Sampling basierten Verfahren eine Simulation führen kann, wurde in einem Fahrer-Fahrzeug-Assistenzsystem-Szenario gezeigt, dass in fast 10% der Simulationsläufe (von 10.000) eine kritische Situation dargestellt werden konnte, im Gegensatz zu 0% bei einer naiven Monte-Carlo-Simulation.

Eine nachfolgende quantitative Auswertung der Simulationsdaten zeigte – im dritten Schritt – nicht nur deutlich kritischere Situationen mit erheblich kleinerer Wahrscheinlichkeit bei der TUTS-geführten Simulation als bei der Monte-Carlo-Simulation, sondern auch eine engere Obergrenze des zur Absicherung der Ergebnisse berechneten Konfidenzniveaus von 99%. Letzteres ist insbesondere im Kontext einer Sicherheitsanalyse, bei der es zumeist darum geht, die Wahrscheinlichkeit einer Gefahr gegen ein Restrisiko durch einen Schwellwert abzuschätzen, ein markanter Vorteil.

Der vierte Schritt zeigt die Übertragbarkeit der geführten Simulation auf andere Domänen. Anhand eines Benchmarks – im Beispiel eines zufällig springenden Balls, der ein winziges Loch treffen soll – bei dem die Wahrscheinlichkeit eines seltenen Ereignisses bekannt ist, wurde der TUTS-Algorithmus mit einer Implementierung der Cross-Entropy sowie naiver Monte-Carlo-Simulation verglichen. Die Ergebnisse zeigen, dass der TUTS-Algorithmus zumindest bei einer kleinen Anzahl an Simulationsläufen pro Batch eine deutlich geringere Unterschätzung der tatsächlichen Wahrscheinlichkeit erzeugt, bei einer Anzahl von mehr als

2.000 Läufen setzt sich hingegen die Implementierung der Cross-Entropy durch. Die Monte-Carlo-Simulation ist in keiner Weise konkurrenzfähig.

Insgesamt lässt sich festhalten, dass Statistisches Model Checking mittels geführter Simulation eine geeignete Methode darstellt, um extrem seltene Ereignisse mit einer deutlich reduzierten Anzahl an Simulationsläufen simulativ zu erfassen. In der Kombination mit einem auf Importance Sampling basierten Algorithmus zur Optimierung der Simulationszeit lassen sich die erzielten Ergebnisse im Nachhinein mathematisch auf das Originalmodell zurückrechnen, so dass mit quantitativen Methoden eine Abschätzung berechnet werden kann, mit welcher Wahrscheinlichkeit die Ergebnisse zutreffend sind bzw. wie hoch das verbleibende Restrisiko im Sinne einer Irrtumswahrscheinlichkeit ist. Die vorgeschlagene Methodik ist daher geeignet, einen Assistenzsystementwickler im Rahmen der modellbasierten Entwicklung eines sicherheitskritischen Fahrerassistenzsystems zu unterstützen.

5.2 Ausblick für zukünftige Arbeiten und offene Fragen

„Alle Menschen streben danach, ihr Leben zu verbessern.“ [Tenzin Gyatso[1]] und auch nach Vorstellung der in dieser Arbeit erzielten Ergebnisse verbleibt Handlungsspielraum zur Verbesserung. Daher sollen an dieser Stelle einige Ideen und offene Fragen festgehalten werden, welche Potential haben, die vorgeschlagene Methodik im Rahmen zukünftiger Forschungsarbeiten weiter zu optimieren.

Parallel zu den Forschungstätigkeiten dieser Arbeit, welche für die Entwicklung des Kosimulationsframeworks die High Level Architecture (HLA) verwendet hat, wurde die Version 2.0 des FMI-Standards veröffentlicht und Version 3.0 befindet sich derzeit in der Erprobungsphase. Die HLA bringt konzeptbedingt durch die zentrale Rolle der RTI, welche als eigenständige Softwarekomponente die Koordination aller Kosimulationsteilnehmer einnimmt, einen „Flaschenhals“ hinsichtlich Performanz mit sich. Außerdem sind kommerzielle und als HLA-kompatibel zertifizierte RTI-Implementierungen im Rahmen von Forschungsprojekten mit hohen Lizenzkosten verbunden. Ein interessanter Ansatz wäre daher, ein vergleichbares Kosimulationsframework auf Basis des aktuellen FMI-Standards umzusetzen und zu beurteilen, ob eine Integration heterogener Simulatoren in eine Kosimulation auf dieser Basis weitere Vereinfachungen hinsichtlich erforderlicher Implementierungsarbeiten und Simulationsperformanz bringt.

Der TUTS-Algorithmus, welcher für die intelligente Simulationsführung vorgeschlagen wurde, verwendet für die Bewertung von kritischem Verhalten, vgl. Unterabschnitt 4.2.3 in Gleichung 4.7, eine Funktion zur Feinjustierung der Suchgeschwindigkeit. Die darin enthaltenen

[1] 14. Dalai Lama, geistiges und politisches Oberhaupt der Tibeter, wurde 1989 mit dem Friedensnobelpreis ausgezeichnet.

freien Parameter a und b wurden für die Evaluationsbeispiele allerdings pragmatisch anhand einfacher Benchmarks aus der kognitiven Architektur CASCaS ermittelt. Eine systematischere Evaluation dieser Parameter könnte für die simulative Erfassung seltener Ereignisse noch eine Verbesserung bringen, so dass bei einer gleichen Anzahl an Simulationsläufen im direkten Vergleich mit einer naiven Monte-Carlo-Simulation eine noch größere Anzahl seltener Ereignissen erzeugt werden könnte. Auch das Anwendungsszenario in Unterabschnitt 4.3.3, in dem der TUTS-Algorithmus evaluiert wurde, ist aus Sicht des Fahrermodells noch sehr einfach gehalten. Komplexere Szenarien auf der Autobahn, wo unterschiedliche Manöver wie beispielsweise einem Fahrzeug folgen, Überholen, in den Verkehrsfluss einfädeln, etc. durchgeführt werden müssen, sind derzeit Gegenstand der Forschung und statt einer künstlichen Nebenaufgabe wäre es wünschenswert, dass auch ein fortschrittliches Autobahnassistenzsystem zur Anwendung kommt, mit dem das Fahrermodell interagieren kann.

Beim direkten Vergleich der zwei Algorithmen – TUTS und AIS mit Cross-Entropy – für eine geführte Simulation anhand eines zufällig springenden Balls wurde gezeigt, dass die Methodik der Simulationsführung konzeptbedingt auch in anderen Domänen gelingt. Allerdings war bei diesem Beispiel weder eine Abstraktion des Zustandsraumes für das verwendete physische Modell des Balls notwendig, noch folgte das Beispiel dem gleichen komplexen Kosimulationsaufbau wie das Fahrer-Fahrzeug-Assistenzsystem-Szenario in Unterabschnitt 4.3.3. Da der TUTS-Algorithmus im Vergleich zu AIS mit Cross-Entropy die im quantitativen Verifikationskontext erstrebenswerte scharfe Obergrenze des Konfidenzintervalls, insbesondere bei kleinen Batchgrößen im Verhältnis zur tatsächlichen Wahrscheinlichkeit eines seltenen Ereignisses, sehr gut schätzen konnte, stellt sich zum einen die Frage, ob sich ein derartiges Ergebnis auch auf ein komplexeres Anwendungsszenario übertragen lässt oder ob sich zum anderen mit einer Kombination beider Verfahren noch bessere Ergebnisse mit einer kleineren Anzahl an Simulationsläufen erzielen ließen.

Eine weitere interessante Forschungsfrage, welche sich aus der vorliegenden Arbeit ableiten lässt, ist die Fragestellung nach derjenigen Konfidenz, die zur Absicherung äußerst kleiner Wahrscheinlichkeiten notwendig ist. Die im Kontext von seltenen Ereignissen üblichen Größenordnungen von 10^{-6} (siehe Abschnitt 2.3 auf Seite 13) beziehungsweise 5×10^{-13} (siehe Unterabschnitt 4.3.5 auf Seite 89) wurden zwar mit einem in der Statistik üblichen Konfidenzniveau von 99% abgesichert, allerdings ist die im Vergleich zur Wahrscheinlichkeit des seltenen Ereignisses übrigbleibende Irrtumswahrscheinlichkeit von 1% um Größenordnungen höher. Eine bedingt vergleichbare Problemstellung findet sich im Bereich der Naturkonstanten. So ist die Newtonsche Gravitationskonstante G aktuell mit einem Wert von $6,67408 \cdot 10^{-11} m^3 kg^{-1} s^{-2}$ definiert, die damit einhergehende Standardunsicherheit (*standard uncertainty*) von $0,00031 \cdot 10^{-11} m^3 kg^{-1} s^{-2}$ zeigt allerdings, dass bereits die vierte Stelle nach dem Komma (bei Betrachtung der relativen Unsicherheit die fünfte Stelle) unsicher ist [MNT15]. Die Rydberg-Konstante R_∞ mit einem Wert von $10.973.731,568508 m^{-1}$

hingegen gilt bei Betrachtung der relativen Unsicherheit von $5,9 \cdot 10^{-12}$ als die derzeit am genauesten gemessene Naturkonstante überhaupt [Poh+10]. Das breite Spektrum zeigt hier einerseits, dass es möglich ist, auch äußerst kleine Werte genau zu spezifizieren, andererseits muss für eine direkte Vergleichbarkeit mit Ergebnissen einer Simulation seltener Ereignisse wie im vorliegenden Kontext berücksichtigt werden, wie viel Zeit für eine derart genaue Bestimmung verfügbar ist und welche Präzision hinreichend ist. Für NASA-Ingenieure reichen beispielsweise zur Kalkulation der interplanetaren Navigation mit der mathematischen Konstante pi(π) 15 Nachkommastellen aus, obwohl Japanische Mathematiker im Jahre 2013 mehr als 12 Billionen Stellen darstellen konnten [NAS16; Yee16]. Eine Übertragbarkeit derartiger Erfahrungswerte in den Kontext der Abschätzung von Sicherheitseigenschaften oder Restrisiken wäre für eine zusätzliche Bewertung der vorgestellten Methodik dieser Arbeit wünschenswert.

Abbildungsverzeichnis

Literaturverzeichnis

[ACT19] ACT-R Research Group. *ACT-R 7.6 Reference Manual*. 2019. URL: http://act-r.psy.cmu.edu/actr7.x/reference-manual.pdf. Zugriff am 05.04.2019.

[AFS04] Luca de Alfaro, Marco Faella und Mariëlle Stoelinga. „Linear and Branching Metrics for Quantitative Transition Systems“. In: *Automata, Languages and Programming*. Springer Berlin Heidelberg, 2004, S. 97–109.

[AXA09] AXA Konzern AG. *AXA Verkehrssicherheits-Report 2009*. Nicht mehr online verfügbar. 2009.

[BB12] Stefan Brosig und Arne Bartels (Volkswagen AG - Konzernforschung Wolfsburg). „Automatisches Fahren als große Herausforderung der Zukunft“. Präsentation vom 13. SafeTRANS Industrial Day. Nov. 2012.

[BDH15] Carlos E. Budde, Pedro R. D'Argenio und Holger Hermanns. „Rare Event Simulation with Fully Automated Importance Splitting“. In: *Computer Performance Engineering: 12th European Workshop, EPEW 2015, Madrid, Spain, August 31 - September 1, 2015, Proceedings*. Hrsg. von Marta Beltrán, William Knottenbelt und Jeremy Bradley. Cham: Springer International Publishing, 2015, S. 275–290.

[BDH19] Carlos E. Budde, Pedro R. D'Argenio und Arnd Hartmanns. „Automated compositional importance splitting“. In: *Science of Computer Programming* 174 (Apr. 2019), S. 90–108.

[Ben18] Klaus Bengler. *Focusing on the driver: A Human Factors Approach to Automated Driving*. Online. Juli 2018. URL: https://www.ko-haf.de/fileadmin/user_upload/media/abschlusspraesentation/06_Ko-HAF_Focusing-on-the-Driver.pdf. Zugriff am 05.04.2019.

[BKW10] Herbert Baum, Thomas Kranz und Ulrich Westerkamp. *Volkswirtschaftliche Kosten durch Straßenverkehrsunfälle in Deutschland 2008*. Info: Bundesanstalt für Straßenwesen. BAST, Referat Öffentlichkeitsarbeit, 2010. URL: https://www.bast.de/BASt_2017/DE/Statistik/Unfaelle/volkswirtschaftliche_kosten.pdf?__blob=publicationFile&v=10. Zugriff am 05.04.2019.

[Boe02] Boeing Commercial Airplane – Airplane Safety. *Statistical Summary of Commercial jet Airplane Accidents - Worldwide Operations - 1995 – 2001*. Technischer Bericht. Boeing, Juni 2002.

[Buc04] James A. Bucklew. *Introduction to Rare Event Simulation*. Springer-Verlag New York, 2004.

[Buc88] Christian G. Bucher. „Adaptive sampling — an iterative fast Monte Carlo procedure“. In: *Structural Safety* 5.2 (Juni 1988), S. 119–126.

[BZ15] James L. Beck und Konstantin M. Zuev. „Rare-Event Simulation“. In: *Handbook of Uncertainty Quantification*. Springer International Publishing, 2015, S. 1–26.

[Cel61] Andreas Cellarius. *Harmonia Macrocosmica*. Amstelodami, apud Joannem Janssonium, 1661. URL: http://www.rarebookroom.org/Control/gelmcs/index.html. Zugriff am 05.04.2019.

[CER] CERTI Project Members. *CERTI*. URL: https://savannah.nongnu.org/projects/certi. Zugriff am 05.04.2019.

[Cla+00] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu und Helmut Veith. „Counterexample-Guided Abstraction Refinement“. In: *Proceedings of the 12th International Conference on Computer Aided Verification*. CAV '00. London, UK, UK: Springer-Verlag, 2000, S. 154–169.

[Cla+03] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu und Helmut Veith. „Counterexample-guided abstraction refinement for symbolic model checking“. In: *Journal of the ACM* 50.5 (Sep. 2003), S. 752–794.

[CP34] Charles J Clopper und Egon S Pearson. „The use of confidence or fiducial limits illustrated in the case of the binomial“. In: *Biometrika* 26.4 (1934), S. 404–413.

[DH07] Flavio M. De Paula und Alan J. Hu. „An Effective Guidance Strategy for Abstraction-guided Simulation“. In: *Proceedings of the 44th Annual Design Automation Conference*. DAC '07. San Diego, California: ACM, 2007, S. 63–68.

[DM10] Alexandre Donzé und Oded Maler. „Robust Satisfaction of Temporal Logic over Real-Valued Signals“. In: *Formal Modeling and Analysis of Timed Systems*. 2010, S. 92–106.

[EAS07] EASA. *Certification Specifications for Large Aeroplanes CS-25*. European Aviation Safety Agency. Sep. 2007. URL: https://www.easa.europa.eu/sites/default/files/dfu/CS-25_Amdt%203_19.09.07_Consolidated%20version.pdf. Zugriff am 05.04.2019.

[EB07] Martin Ehmann und Torsten Butz. „Modellbasierte Entwicklung von Fahrerassistenzsystemen“. In: *9. Internationales Forum Nutzfahrzeuge*. VDI Verlag GmbH, Düsseldorf, Juni 2007, S. 481–488.

[Ehm05] Martin Ehmann. „Verkehrssimulation zum Test von Fahrerassistenzsystemen mit Umfeldsensorik.“ In: *AEP - Automotive Engineering Partners* 7/8 (2005), S. 32–35.

[ET93] Bradley Efron und Robert J. Tibshirani. *An Introduction to the Bootstrap*. Monographs on Statistics and Applied Probability 57. Chapman & Hall/CRC, 1993.

[FH05] Martin Fränzle und Michael R. Hansen. „A Robust Interpretation of Duration Calculus“. In: *Theoretical Aspects of Computing - ICTAC 2005, Second International Colloquium, Hanoi, Vietnam, October 17-21, 2005, Proceedings*. Hrsg. von Dang Van Hung und Martin Wirsing. Bd. 3722. LNCS. Springer, 2005, S. 257–271.

[FOL10] Florian Frische, Jan-Patrick Osterloh und Andreas Lüdtke. „Simulating Visual Attention Allocation of Pilots in an Advanced Cockpit Environment“. In: *Proceedings of the MODSIM World Conference and Expo*. Dez. 2010.

[FOL11] Florian Frische, Jan-Patrick Osterloh und Andreas Lüdtke. „Modelling and Validating Pilots Visual Attention Allocation during the Interaction with an Advanced Flight Management System“. In: *Human Modelling in Assisted Transportation*. Hrsg. von P. C. Cacciabue, M. Hjälmdahl, Andreas Lüdtke und C. Riccioli. Springer, Aug. 2011, S. 165–172.

[Frä+10] Martin Fränzle, Tayfun Gezgin, Hardi Hungar, Stefan Puch und Gerald Sauter. „Using Guided Simulation to Assess Driver Assistance Systems“. In: *Proc. FORMS/FORMAT 2010*. Hrsg. von E. Schnieder und G. Tarnai. 2010.

[Frä+11] Martin Fränzle, Tayfun Gezgin, Hardi Hungar, Stefan Puch und Gerald Sauter. „Predicting the Effect of Driver Assistance via Simulation“. In: *Human Modelling in Assisted Transportation*. Hrsg. von P. C. Cacciabue, M. Hjälmdahl, Andreas Lüdtke und C. Riccioli. Springer, 2011, S. 299–306.

[Gag17] Paul A. Gagniuc. *Markov chains : from theory to implementation and experimentation*. Hoboken, NJ : John Wiley & Sons, 2017.

[GDV05] Olaf Gietelink, Bart De Schutter und Michel Verhaegen. „Probabilistic validation of advanced driver assistance systems“. In: *Proceedings of the 16th IFAC World Congress* 19 (2005).

[GDV06] Olaf Gietelink, Bart De Schutter und Michel Verhaegen. „Adaptive importance sampling for probabilistic validation of advanced driver assistance systems“. In: *2006 American Control Conference* 19 (2006), 6 pp.

[Gez09] Tayfun Gezgin. „Observerbasierte on-the-fly Auswertung von QLTL-Formeln innerhalb eines HLA-Simulationsverbundes“. Masterarbeit. Carl von Ossietzky Universität Oldenburg, Sep. 2009.

[Gol16] Volker Gollücke. „Bewertung von Simulationszuständen für eine gezielte Analyse risikoreicher Systeme“. Dissertation. Universität Oldenburg, 2016.

[Hen13] Norbert Henze. *Stochastik für Einsteiger*. Springer Fachmedien Wiesbaden, 2013.

[Hoe63] Wassily Hoeffding. „Probability inequalities for sums of bounded random variables“. In: *Journal of the American Statistical Association* 58.301 (März 1963), S. 13–30.

[Hof12] U. Hoffmann. *Resümee der wissenschaftlichen Begleitung der Aktion „SICHER. FÜR DICH. FÜR MICH.“* BG Verkehr, BGL und KRAVAG. März 2012. URL: https://www.bg-verkehr.de/redaktion/medien-und-downloads/informationen/themen/aktionen-und-kampagnen/fahrer-assistenz-systeme/04_resuemee.pdf. Zugriff am 05.04.2019.

[Hu05] Yunwei Hu. „A Guided Simulation Methodology for Dynamic Probabilistic Risk Assessment of Complex Systems“. Dissertation. University of Maryland, 2005.

[Hül18] Dirk Hülsebusch. *Fahrerassistenzsysteme zur energieeffizienten Längsregelung - Analyse und Optimierung der Fahrsicherheit*. Karlsruher Schriftenreihe Fahrzeugsystemtechnik / Institut fuer Fahrzeugsystemtechnik. Karlsruher Institut für Technologie, 2018. ISBN: 9783731507550.

[Hum+11] Thomas Hummel, Matthias Kühn, Jenö Bende und Antje Lang. *Fahrerassistenzsysteme: Ermittlung des Sicherheitspotenzials auf Basis des Schadengeschehens der deutschen Versicherer*. Forschungsbericht / Gesamtverband der Deutschen Versicherungswirtschaft e.V. GDV, Sep. 2011.

[Hum18] Human Centered Design Group. *Kongnitive Architektur CASCaS*. 2018. URL: https://hcd.offis.de/wordpress/?page_id=16. Zugriff am 05.04.2019.

[Hup97a] Christoph Hupfer. „Computergestützte Videobildverarbeitung zur Verkehrssicherheitsarbeit: am Beispiel von Fußgängerquerungen an städtischen Hauptverkehrsstraßen“. Dissertation. Hochschule Karlsruhe Technik und Wirtschaft, 1997.

[Hup97b] Christoph Hupfer. „Deceleration to safety time (DST) – a useful figure to evaluate traffic safety“. In: *ICTCT Conference Proceedings of Seminar 3*. Lund University, Sweden, 1997.

[HWC06] William J. Horrey, Christopher D. Wickens und Kyle P. Consalus. „Modeling drivers' visual attention allocation while interacting with in-vehicle technologies.“ In: *Journal of Experimental Psychology: Applied* 12.2 (2006), S. 67–78.

[Hyd87] Christer Hydén. „The development of a method for traffic safety evaluation: The Swedish Traffic Conflict Technique.“ Dissertation. Lund University, Sweden, 1987.

[IEE10] IEEE Computer Society. „IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)“. In: *IEEE Std. 1516-2010* (Aug. 2010).

[IEE12] IEEE Computer Society. „IEEE Standard for Distributed Interactive Simulation – Application Protocols“. In: *IEEE Std. 1278.1-2012* (Dez. 2012).

[ISO15] ISO. *Information technology – Vocabulary*. Standard ISO/IEC 2382:2015. International Organization for Standardization, Mai 2015. URL: https://www.iso.org/standard/63598.html. Zugriff am 05.04.2019.

[Jeg+16] Cyrille Jegourel, Kim G. Larsen, Axel Legay, Marius Mikučionis u. a. „Importance Sampling for Stochastic Timed Automata“. In: *Dependable Software Engineering: Theories, Tools, and Applications: Second International Symposium, SETTA 2016, Beijing, China, November 9-11, 2016, Proceedings*. Hrsg. von Martin Fränzle, Deepak Kapur und Naijun Zhan. Cham: Springer International Publishing, 2016, S. 163–178.

[JLS13] Cyrille Jegourel, Axel Legay und Sean Sedwards. „Importance Splitting for Statistical Model Checking Rare Properties“. In: *Computer Aided Verification*. Hrsg. von Natasha Sharygina und Helmut Veith. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, S. 576–591.

[JS06] Sandeep Juneja und Perwez Shahabuddin. „Chapter 11 Rare-Event Simulation Techniques: An Introduction and Recent Advances“. In: *Simulation*. Elsevier, 2006, S. 291–350.

[KEB07] Werner Kober, Martin Ehmann und Torsten Butz. „Model-based development of an integrated vehicle dynamics control“. In: *ATZ worldwide* 109.10 (Okt. 2007), S. 13–16.

[KL51] Solomon Kullback und Richard Arthur Leibler. „On Information and Sufficiency“. In: *The Annals of Mathematical Statistics* 22.1 (März 1951), S. 79–86.

[Lam] Laminar Research. *Flugsimulator X-Plane*. URL: https://www.x-plane.com/. Zugriff am 05.04.2019.

[Läs11] Christoph Läsche. „Evaluation und Optimierung einer Open-Source-RTI zur Kosimulation heterogener Systemklassen“. Masterarbeit. Carl von Ossietzky Universität Oldenburg, Sep. 2011.

[LDB10] Axel Legay, Benoît Delahaye und Saddek Bensalem. „Statistical Model Checking: An Overview“. In: *Runtime Verification*. Springer Berlin Heidelberg, 2010, S. 122–135.

[LDT06] Pierre L'Ecuyer, Valérie Demers und Bruno Tuffin. „Splitting for Rare-event Simulation“. In: *Proceedings of the 38th Conference on Winter Simulation*. WSC '06. Monterey, California: Winter Simulation Conference, 2006, S. 137–148.

[Lüd+09] Andreas Lüdtke, Lars Weber, Jan-Patrick Osterloh und Bertram Wortelen. „Modeling Pilot and Driver Behaviour for Human Error Simulation“. In: *HCI International 2009*. LNCS 5610–56. Springer, Okt. 2009.

[MB01] Michiel M. Minderhoud und Piet H. L. Bovy. „Extended time-to-collision measures for road traffic safety assessment“. In: *Accident Analysis & Prevention* 33.1 (Jan. 2001), S. 89–97.

[Meh94] Horst Mehl. *Methoden verteilter Simulation*. Friedrich Vieweg & Sohn Verlagsgesellschaft mbH, Baunschweig/Wiesbaden, 1994, S. 258.

[MNT15] Peter J. Mohr, David B. Newell und Barry N. Taylor. *Codata Recommended Values Of The Fundamental Physical Constants: 2014*. 2015.

[Mod14] Modelica Association. *Functional Mock-up Interface (FMI) 2.0*. 2014. URL: https://svn.modelica.org/fmi/branches/public/specifications/v2.0/FMI_for_ModelExchange_and_CoSimulation_v2.0.pdf. Zugriff am 05.04.2019.

[NAS16] NASA/JPL Edu. *How Many Decimals of Pi Do We Really Need?* 2016. URL: https://www.jpl.nasa.gov/edu/news/2016/3/16/how-many-decimals-of-pi-do-we-really-need/. Zugriff am 05.04.2019.

[Pag+15] Yves Page, Felix Fahrenkrog, Anita Fiorentino, Martin Fränzle u. a. „A Comprehensive and Harmonized Method for Assessing the Effectiveness of Advanced Driver Assistance Systems by Virtual Simulation: The P.E.A.R.S. Initiative“. In: *The 24th International Technical Conference on the Enhanced Safety of Vehicles (ESV)*. PAPERNO.15-0370. National Highway Traffic Safety Administration. Gothenburg, Sweden, Juni 2015.

[PFG18] Stefan Puch, Martin Fränzle und Sebastian Gerwinn. „Quantitative Risk Assessment of Safety-Critical Systems via Guided Simulation for Rare Events“. In: *Leveraging Applications of Formal Methods, Verification and Validation. Verification*. Springer International Publishing, 2018, S. 305–321.

[Pit] Pitch Company. *Pitch pRTI*. URL: http://pitchtechnologies.com/products/prti/. Zugriff am 05.04.2019.

[Poh+10] Randolf Pohl, Aldo Antognini, François Nez, Fernando D. Amaro u. a. „The size of the proton“. In: *Nature* 466.7303 (Juli 2010), S. 213–216.

[Puc+12a] Stefan Puch, Martin Fränzle, Jan-Patrick Osterloh und Christoph Lässche. „Rapid Virtual-Human-in-the-Loop Simulation with the High Level Architecture“. In: *Proceedings of the 2012 Summer Computer Simulation Conference (SCSC ´12)*. Hrsg. von A. Bruzzone, F. Longo, P. Kropf und M. A. Piera. Bd. 44. 10. The Society for Modeling & Simulation International (SCS). Genua: Curran Associates, Inc., Juli 2012, S. 44–50.

[Puc+12b] Stefan Puch, Bertram Wortelen, Martin Fränzle und Thomas Peikenkamp. „Using Guided Simulation to Improve a Model-Based Design Process of Complex Human Machine Systems“. In: *ESM'2012 - The 2012 European Simulation And Modelling Conference*. Essen: EUROSIS-ETI, Okt. 2012, S. 159–164.

[Puc+13] Stefan Puch, Bertram Wortelen, Martin Fränzle und Thomas Peikenkamp. „Evaluation of Drivers Interaction with Assistant Systems using Criticality Driven Guided Simulation“. In: *Digital Human Modeling and Applications in Health, Safety, Ergonomics, and Risk Management. Healthcare and Safety of the Environment and Transport*. Hrsg. von Vincent G. Duffy. Bd. 8025. Lecture Notes in Computer Science LNCS 8025-8026. Springer Berlin Heidelberg, Juni 2013, S. 108–117.

[Puc07] Stefan Puch. „Kosimulation von Verkehrssystemen und Operateuren“. Diplomarbeit. Carl von Ossietzky Universität Oldenburg, Okt. 2007.

[Rin08] Horst Rinne. *Taschenbuch der Statistik*. 4. Aufl. Wissenschaftlicher Verlag Harri Deutsch, 2008.

[RT09] Gerardo Rubino und Bruno Tuffin, Hrsg. *Rare Event Simulation using Monte Carlo Methods*. John Wiley & Sons, Ltd, März 2009.

[Rub99] Reuven Rubinstein. „The Cross-Entropy Method for Combinatorial and Continuous Optimization“. In: *Methodology And Computing In Applied Probability* 1.2 (1999), S. 127–190.

[San04] Werner Sandmann. „Simulation seltener Ereignisse mittels Importance Sampling unter besonderer Berücksichtigung Markovscher Modelle“. Dissertation. Universität Bonn, 2004.

[Sau+09] Gerald Sauter, Henning Dierks, Martin Fränzle und Michael R. Hansen. „Lightweight hybrid model checking facilitating online prediction of temporal properties“. In: *Proceedings of the 21st Nordic Workshop on Programming Theory, NWPT '09*. Kgs. Lyngby, Denmark: Danmarks Tekniske Universitet, 2009, S. 20–22. URL: http://imost.informatik.uni-oldenburg.de/download/RobustMonitoring-NWPT09.pdf. Zugriff am 05.04.2019.

[SB06] Smitha Shyam und Valeria Bertacco. „Distance-guided Hybrid Verification with GUIDO“. In: *Proceedings of the Conference on Design, Automation and Test in Europe: Proceedings*. DATE '06. Munich, Germany: European Design und Automation Association, 2006, S. 1211–1216.

[SB98] Richard S. Sutton und Andrew G. Barto. *Introduction to Reinforcement Learning*. 1st. Cambridge, MA, USA: MIT Press, 1998.

[SIS15] SISO. *RPR FOM 2.0*. Simulation Interoperability Standards Organization. Aug. 2015. URL: http://www.sisostds.org/ProductsPublications/Standards/SISOStandards.aspx. Zugriff am 05.04.2019.

[SS17] Michael Salins und Konstantinos Spiliopoulos. „Rare event simulation via importance sampling for linear SPDE's“. In: *Stochastics and Partial Differential Equations: Analysis and Computations* 5.4 (Dez. 2017), S. 652–690.

[Sta17a] Statistisches Bundesamt. „Unfallentwicklung auf Deutschen Straßen 2017“. In: 2017. URL: https://www.destatis.de/DE/Presse/Pressekonferenzen/2018/Verkehrsunfaelle-2017/pressebroschuere-unfallentwicklung.pdf?__blob=publicationFile&v=3. Zugriff am 05.04.2019.

[Sta17b] Statistisches Bundesamt. „Verkehrsunfälle 2017“. In: Fachserie 8 Reihe 7. Statistisches Bundesamt, Juli 2017. URL: https://www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Verkehrsunfaelle/Publikationen/Downloads-Verkehrsunfaelle/verkehrsunfaelle-jahr-2080700177004.pdf?__blob=publicationFile&v=4. Zugriff am 05.04.2019.

[STS11] ST Software BV. *ST Software Simulationssysteme*. 2011. URL: https://www.stsoftware.nl/. Zugriff am 05.04.2019.

[SW95] Nadine B. Sarter und David D. Woods. „How in the World Did We Ever Get into That Mode? Mode Error and Awareness in Supervisory Control“. In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 37.1 (März 1995), S. 5–19.

[TA09] Niels Taatgen und John R. Anderson. „The Past, Present, and Future of Cognitive Architectures“. In: *Topics in Cognitive Science* 2.4 (Nov. 2009), S. 693–704.

[VIR19] VIRES Simulationstechnologie GmbH. *VTD - Virtual Test Drive*. 2019. URL: https://vires.com. Zugriff am 05.04.2019.

[Vog03] Katja Vogel. „A comparison of headway and time to collision as safety indicators“. In: *Accident Analysis & Prevention* 35.3 (Mai 2003), S. 427–433.

[Vor10] Ingeborg Vorndran. *Unfallstatistik - Verkehrsmittel im Risikovergleich*. Statistisches Bundesamt, Dez. 2010. URL: https://www.destatis.de/GPStatistik/servlets/MCRFileNodeServlet/DEAusgabe_derivate_00000103/1010200101124.pdf. Zugriff am 05.04.2019.

[VV91] Manuel Villén-Altamirano und José Villén-Altamirano. „RESTART: A Method for Accelerating Rare Event Simulations“. In: *Queueing, Performance and Control in ATM: Proc. Of the 13th Internat. Teletraffic Congress, Copenhagen, Denmark* (Juni 1991). Hrsg. von J.W. Cohen und C.D. Pack, S. 71–76.

[Wan+18] Bernhard Wandtner, Gerald Schmidt, Nandja Schoemig und Wilfried Kunde. „Non-driving related tasks in highly automated driving - Effects of task modalities and cognitive workload on take-over performance“. In: *AmE 2018 - Automotive meets Electronics; 9th GMM-Symposium*. März 2018, S. 1–6.

[WBL13] Bertram Wortelen, Martin Baumann und Andreas Lüdtke. „Dynamic Simulation and Prediction of Drivers' Attention Distribution“. In: *Transportation Research Part F: Traffic Psychology and Behaviour* 21 (2013), S. 278–294.

[Web+09] Lars Weber, Martin Baumann, Andreas Lüdtke und Rike Steenken. „Modellierung von Entscheidungen beim Einfädeln auf die Autobahn“. In: *Der Mensch im Mittelpunkt technischer Systeme*. Hrsg. von A. Lichtenstein, C. Stößel und C. Clemens. Forschritts-Berichte VDI. Zentrum Mensch-Maschine Systeme der TU Berlin. VDI Verlag, Jan. 2009, S. 86–91.

[Web17] Lars Weber. „Driver Modeling and Simulation of Lane Change Situations“. Dissertation. Universität Oldenburg, 2017.

[WH16] Karl-Heinz Waldmann und Werner E. Helm. *Simulation stochastischer Systeme: Eine anwendungsorientierte Einführung*. Berlin: Springer Gabler, 2016.

[Wic+07] Christopher D. Wickens, Michael D. Fleetwood, A. Alexander, Michael Ambinder u. a. *Human Performance Modeling in Aviation*. Hrsg. von David Foyle und Becky Hooey. CRC Press, Dez. 2007.

[WIV19] WIVW GmbH. *Fahrsimulationssoftware SILAB*. 2019. URL: https://wivw.de/de/silab. Zugriff am 05.04.2019.

[WLB13] Bertram Wortelen, Andreas Lüdtke und Martin Baumann. „Integrated Simulation of Attention Distribution and Driving Behavior“. In: *Proceedings of the 22nd Annual Conference on Behavior Representation in Modeling & Simulation*. Ottawa, Canada: BRIMS Society, 2013, S. 69–76.

[WM97] David H. Wolpert und William G. Macready. „No free lunch theorems for optimization“. In: *IEEE Transactions on Evolutionary Computation* 1.1 (Apr. 1997), S. 67–82.

[Wor14] Bertram Wortelen. „Das Adaptive-Information-Expectancy-Modell zur Aufmerksamkeitssimulation eines kognitiven Fahrermodells“. Dissertation. Universität Oldenburg, 2014.

[Yee16] Yee, Alexander J. and Kondo, Shigeru. *12.1 Trillion Digits of Pi*. 2016. URL: http://www.numberworld.org/misc_runs/pi-12t/. Zugriff am 05.04.2019.

[Yil+14] Faruk Yilmaz, Umut Durak, Koray Taylan und Halit Oğuztüzün. „Adapting Functional Mockup Units for HLA-compliant Distributed Simulation“. In: *Proceedings of the 10th International Modelica Conference*. 96. Linköping University Electronic Press, 2014, S. 247–257.

[You+06] Håkan L. S. Younes, Marta Kwiatkowska, Gethin Norman und David Parker. „Numerical vs. statistical probabilistic model checking“. In: *Int. J. Softw. Tools Technol. Transf.* 8.3 (Juni 2006), S. 216–228.

[YS02] Håkan L. S. Younes und Reid G. Simmons. „Probabilistic Verification of Discrete Event Systems Using Acceptance Sampling“. In: *Computer Aided Verification*. Springer Berlin Heidelberg, 2002, S. 223–235.

[YS06] Håkan L. S. Younes und Reid G. Simmons. „Statistical probabilistic model checking with a focus on time-bounded properties“. In: *Information and Computation* 204.9 (2006), S. 1368–1409.

[ZBC12] Paolo Zuliani, Christel Baier und Edmund M. Clarke. „Rare-event verification for stochastic hybrid systems“. In: *HSCC*. 2012, S. 217–226.

[ZBS16] Kathrin Zeeb, Axel Buchner und Michael Schrauf. „Is take-over time all that matters? The impact of visual-cognitive load on driver take-over quality after conditionally automated driving“. In: *Accident Analysis & Prevention* 92 (Juli 2016), S. 230–239.

[ZLL10] Tao Zhang, Tao Lv und Xiaowei Li. „An Abstraction-guided Simulation Approach Using Markov Models for Microprocessor Verification“. In: *Proceedings of the Conference on Design, Automation and Test in Europe*. DATE '10. Dresden, Germany: European Design und Automation Association, 2010, S. 484–489.

Glossar

AIS	*Adaptive Importance Sampling* ist eine adaptive Form des IS, bei dem der Gewichtungsfaktor über die Zeit angepasst werden kann. 23, 57, 70, 84, 86, 89, 92, 94, 95, 104, 120
CEGAR	*Counterexample-Guided Abstraction Refinement* ist eine Methode aus dem Bereich des automatischen Model Checking, bei der zuerst ein abstraktes Modell aus einem gegebenen Programm oder Modell generiert wird. Anschließend wird mit symbolischen Techniken versucht, das abstrakte Modell zu verifizieren. Die abstrakten Modelle können allerdings fehlerhaft sein („spurious“ counterexamples), so dass in fortlaufenden Iterationen der Modellverfeinerung versucht wird, die Fehler zu beheben. 21, 27, 67
FAS	Ein *FahrerAssistenzSystem* ist ein technisches System, welches den Autofahrer bei der Ausübung von Tätigkeiten wie Einparken, Spurhalten, Navigieren, etc. unterstützen soll. Teilweise können einzelne Aufgaben auch komplett automatisiert ausgeführt werden wie z.B. eine Notbremsung. 23, 33, 39, 50, 56
Gesetz der großen Zahlen	Das Gesetz der großen Zahlen sagt, dass bei fortwährender Wiederholung desselben Zufallsexperiments der Durchschnitt des Ergebnisses gegen den Erwartungswert konvergiert. 15, 16

HLA	Die *High Level Architecture* ist im IEEE-Standard 1516 definiert und stellt Rahmenbedingungen für Simulationsentwickler zur Verfügung, mit denen unterschiedliche Simulatoren gemeinsam innerhalb einer Kosimulation ausgeführt werden können. 31, 41–44, 46–48, 50, 52, 59, 103
IS	*Importance Sampling* ist ein stochastisches Verfahren zur Varianzreduktion bei MCS, bei der die relevanten Stichproben bevorzugt werden. 22–24, 26, 27, 29, 30, 59, 61, 68, 70, 71, 74, 85, 86, 96, 119
ISp	*Importance Splitting* ist ein stochastisches Verfahren zur Varianzreduktion bei MCS, welches sich stufenweise einem definierten Ziel annähert. 24–27, 31, 59, 61
MCS	*Monte-Carlo-Simulation* ist ein stochastisches Verfahren, bei dem eine große Anzahl an Zufallsexperimenten (Stichproben) durchgeführt wird. Im Simulationsbereich repräsentiert eine Stichprobe z. B. einem Simulationslauf. 22, 24, 86, 88, 89, 92, 94–96, 100, 102, 120
Simulation	Simulation ist eine Analysemethode, die in vielen Bereichen wie Technik, Wirtschaft und auch der Wissenschaft eingesetzt wird, um Untersuchungen eines realen Systems anhand eines vereinfachten Modells durchzuführen [Meh94]. 10
SMC	*Statistisches Modell Checking* kombiniert Monte-Carlo-Simulation, Model Checking und Statistische Analyse, um stochastische Systeme zu verifizieren. 5, 15, 25
TUTS	*Threshold Uncertainty Tree Search* ist ein auf AIS basierender Algorithmus, welcher mit Hilfe einer Kritikalitätsfunktion eine Simulation zu seltenen Ereignissen führen kann. 62, 68, 70, 74, 77, 83, 88, 98

Index

T

Z

Versicherung

Hiermit versichere ich, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Außerdem versichere ich, dass ich die allgemeinen Prinzipien wissenschaftlicher Arbeit und Veröffentlichung, wie sie in der **Ordnung über die Grundsätze zur Sicherung guter wissenschaftlicher Praxis an der Carl von Ossietzky Universität** Oldenburg festgelegt sind, befolgt habe.

Oldenburg, im Oktober 2019

Stefan Puch