TU Clausthal

Diplomarbeit

Verfahren der dynamischen Gruppenbildung zur Koordination autonomer Fahrzeuge im Straßenverkehr

vorgelegt von

Viet Hung Chu
Wirtschaftsinformatik
Matrikelnummer: 334248

Erstgutachter:
Prof. Dr. Jörg P. Müller
Institut für Informatik

Zweigutachter:
Prof. Dr. Niels Pinkwart
Institut für Informatik

Clausthal-Zellerfeld, den 16. Mai 2011

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

Kürzel	**Beschreibung**
ACC	Adaptive Cruise Control
AI	Artifical Intelligent
AM	Agent Manager
ASM	Aimsun Simulation Model
ATSIM	Agent-based Traffic Simulation System
BA	Behaviour Activator
BBC	Behaviour-based Component
BC	Behaviour Chooser
BCC	Behaviours Control Component
BDI	Belief Desire Intesion
CACC	Cooperative Adaptive Cruise Control
CC	Cooperation Component
CFM	Car Following Model
DAI	Distributed Artifical Intelligent
DC	Data Converter
DLC	Discretionary Lane Changing
DPS	Distributed Problem Solving
DRW	Data Reader and Writer
FIPA	The Foundation for Intelligent Physical Agents
GAM	Gap Acceptance Model
ITS	Intelligent Transport System
JAC	Jade Agent Container
JADE	Java Agent Development Environment
JP	Joints Plan
KI	Künstliche Intelligent
LKW	Lastkraftwagen
MASC	Multi Agent System Connector
MASM	Multi Agent System Manager
MAS	Multi Agent System
MASS	Multi Agent System Service
MLC	Mandatory Lane Changing
PBC	Plan-based Component
PCC	Plan Control Component
PC	Process Controller
PCU	Plan Controll Unit
PE	Plan Extractor

PKW	Personenkraftwagen
PM	Plan Monitor
PoB	Pattern of Behaviour
PRS	Procedural Reasoning System
PSA	Problem Solver Activator
SC	Simulation Controller
SS	Step Synchroniser
TOA	Traffic Object Agent
TOC	Traffic Object Container
WI	World Interface

1 Einleitung

Informationstechnologien und Kommunikationstechnologien werden heutzutage in allen Bereichen verwendet. Die Technologien erleichtern die menschliche Arbeit und erhöhen die Arbeitsleistungen. Auch im Verkehrsbereich spielen die Informationen und Kommunikationstechnologien eine wichtige Rolle. Die Vision der Konstruktion eines intelligenten Transportsystems, in dem alle Verkehrskomponenten (z.B. Fahrzeuge, Straßenampeln) intelligent miteinander arbeiten, ist realisierbar. Solche Systeme werden als „Intelligent Transport System“ (ITS) bezeichnet. Die Erhöhung des Lebensstandards und die Zunahme des Verkehrs sind eine hohe Belastung für die vorhandene Verkehrsinfrastruktur und Herausforderung für das Verkehrsmanagement. Menschen wollen schneller, sicherer, bequemer und effizienter ihre Ziele erreichen. Deshalb kommt einem ITS eine immer größere Bedeutung zu. Der Begriff *Intelligent Transport System* (ITS) wird wie folgt definiert:

ITS can therefore be defined as the application of computing, information and communications technologies to the real-time management of vehicles and neworks involving the movement of people and goods [40]

In einem ITS können alle Verkehrskomponenten (z.B. Fahrzeuge, Straßenampeln, Straßenschilder, usw.) miteinander kommunizieren und Informationen austauschen. Die Integration von Mikrochips in Verkehrskomponenten ermöglicht diesen in einer intelligenten Weise zu arbeiten. Dadurch hilft ITS den Verkehrsteilnehmern sicher, effizient und angenehm zum Ziel zu fahren. Verschiedene Projekte (z.B. PATH [1], AUTO21 [2]) wurden durchgeführt um mögliche Anwendungen des ITS zur Verbesserung der Sicherheit, Geschwindigkeit, Bequemlichkeit von Verkehrsteilnehmern zu untersuchen.

Ein bedeutender Forschungsbereich des ITS ist die Konstruktion von intelligenten autonomen Fahrzeugen, die in der Lage sind miteinander zu kollaborieren um sicher und effizient zum Ziel zu fahren. In den letzten Jahrzehnten wurden zahlreiche Experimente mit autonomen Fahrzeugen durchgeführt und es wurde gezeigt, dass autonome Fahrzeuge das Fahren von Menschen übernehmen könnten. Das beste Beispiel dafür ist der „Defense Advanced Research Projects Agency (DARPA) Grand Challenge“[3] Wettbewerb. In dem Wettbewerb fahren unbemannte und bemannte Fahrzeuge durch verschiedene nachgebildete Straßen in einer Stadt. Ein anderes Beispiel ist das autonome Fahrzeug „Google“[4], welches ungefähr 1600km auf echten Straßen in „San Francisco“ gefahren ist. Obwohl es noch ein langer Weg bis zum wirklichen Einsatz von au-

[1] http://www.path.berkeley.edu/
[2] http://www.auto21.ca/en/news.php
[3] http://www.darpa.mil/grandchallenge/index.asp
[4] http://www.nytimes.com/2010/10/10/science/10google.html

tonomen Fahrzeugen im Alltag ist, wurden einige Technologien in diesem Bereich von luxuriösen Autoherstellern in ihre Produkte integriert, z.B.:

- Die „Adaptive Cruise Control(ACC)"-Technik erlaubt einem Audi-Fahrzeug [5] automatisch seine Geschwindigkeit zu regeln, damit es zu keiner Kollision mit dem Vordermann kommt. Die weiterentwickelte Version der ACC ist „Cooperative Adaptive Cruise Control(CACC)". Die CACC ermöglicht Fahrzeugen miteinander zu kooperieren um mögliche Kollisionen in komplizierten Situationen zu vermeiden.

- Das „Parking-Assist"-System [6] kann ein Fahrzeug autonom einparken.

Eine der wichtigsten Aufgaben der autonomen Fahrzeuge im ITS ist das Kollaborieren. Das Kollaborieren ist die Zusammenarbeit von individuellen Fahrzeugen um ihre Ziele zu erreichen. In vielen Verkehrssituationen ist es einem Fahrzeug nicht möglich sein Ziel effektiv allein zu erreichen. Stau ist ein Beispiel. Die Fahrzeuge im Stau können nichts machen außer warten, bis sie die Stau-Stelle passiert haben. Die Fahrzeuge könnten jedoch schnell eine mögliche neue Route finden ohne durch die Stau-Stelle zu fahren, wenn die Stau-Informationen zwischen ihnen ausgetauscht würden. Diese Arbeit konzentriert sich auf die Anwendung von kollaborierenden autonomen Fahrzeugen zur Verbesserung von Fahrzeuggeschwindigkeiten. Dadurch reduzieren Verkehrsteilnehmer ihre Reisezeit und kommen schneller zu ihren Zielen.

1.1 Problemstellung und Ziel der Arbeit

In Folgenden beschreibe ich ein häufig auftretendes Problem im Straßenverkehr und eine mögliche Anwendung der dynamischen autonomen Fahrzeuggruppen um das Problem zu lösen. Jeder Verkehrsteilnehmer möchte eine bestimmte Geschwindigkeit fahren. Aufgrund von Kollisionsgefahren muss man manchmal seine Geschwindigkeit reduzieren. Die drei Aktionen Beschleunigung, Verzögerung und Spurwechsel müssen zu jedem Zeitpunk passend gewählt werden. Die Auswahl der Aktionen hängt von vielen Faktoren ab (z.B. Verkehrsgesetzgebung, eigene Analyse des wahrgenommenen Verkehrszustandes oder auch aktuelle Mentalität eines Fahrers). Sei der beste Balance-Punkt eines Verkehrszustandes der Punkt, an dem alle Verkehrsteilnehmer mit ihren aktuellen Geschwindigkeiten zufrieden sind. Dies bedeutet, dass der Balance-Punkt nur erreicht werden kann, wenn alle Fahrer ihre gewünschten Geschwindigkeiten fahren. Ohne Kommunikation kommt es oft in realen Verkehrszuständen vor, dass sich die Verkehrsteilnehmer wegen des intensiven Verfolgens ihrer persönlichen gewünschten Geschwindigkeiten untereinander blockieren. Dadurch behindern sie ihre Chance die gewünschte Geschwindigkeit zu erreichen.

[5] http://www.audi.com/com/brand/en/tools/advice/glossary/adaptive_cruise_control .browser.html

[6] http://www.smartpark.net/index.html

1.1.1 Beispiel

Die Abbildung 1.1 zeigt ein Beispiel für solche Fälle.

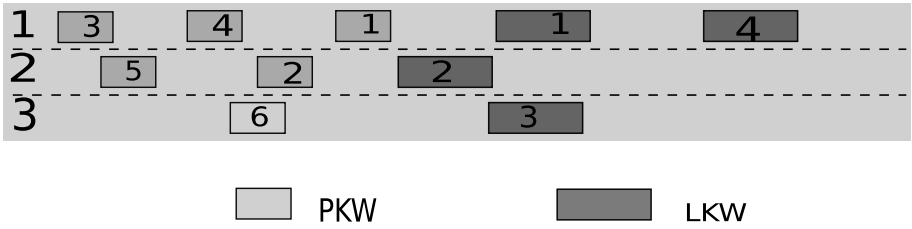


Abbildung 1.1: Die Lastkraftwagen (LKW)-Fahrer blockieren alle Spuren der Personenkraftwagen (PKW)-Fahrer

Angenommen, dass die PKW-Fahrer (kleine Fahrzeuge) hohe Geschwindigkeiten fahren möchten. Wegen ihrer schweren Ladungen sind die LKW-Fahrer (große Fahrzeuge) mit ihren aktuellen Geschwindigkeiten zufrieden. Wie in Abbildung 1.1 zeigt müssen die PKW-Fahrer ihre Geschwindigkeiten verzögern, so dass keine Kollisionen mit den LKW-Fahrern entstehen. Diese Situation wird so lange andauern bis eine Spur von den LKW-Fahrern freigegeben wird. Weil das Rechtsüberholen laut Verkehrsgesetz verboten ist, kann nur die Spur 1 frei gemacht werden. Auf der Spur 2 fahren der LKW-Fahrer 1 und der LKW-Fahrer 4. Wechseln die LKW-Fahrer 1 und 4 auf die Spur 2, so ist die Spur 1 frei und die PKW-Fahrer könnten auf ihre gewünschten Geschwindigkeiten beschleunigen. Der Spurwechsel des LKW-Fahrers 1 führt jedoch zu der Gefahr mit dem LKW-Fahrer 2 zu kollidieren. Ohne Kommunikation mit LKW-Fahrer 2 muss der LKW-Fahrer 1 auf der Spur 1 fahren bis eine sichere Lücke für seinen Spurwechsel entsteht. Angenommen, dass nach langer Zeit der LKW-Fahrer 1 eine ausreichend große Lücke auf der Spur 2 findet und zu dieser Spur wechselt. Dann können die PKW-Fahrer jedoch nicht auf ihre gewünschten Geschwindigkeiten beschleunigen weil sie noch von dem LKW-Fahrer 4 blockiert werden.

Das Beispiel hat gezeigt, dass ohne Kommunikation und ohne Koordination die LKW-Fahrer die PKW-Fahrer stark blockieren. Meine Arbeit beschäftigt sich mit der Lösung des beschriebenen Problems.

1.1.2 Motivation der gruppenorientierten Fahrmethode

In dieser Arbeit wird angenommen, dass die Fahrzeuge sich in einem ITS befinden. Wegen der technischen Beschränkung von Kommunikationstechnologien kann ein Fahrzeug nur mit den Fahrzeugen, die sich in einem bestimmten Umkreis befinden, kommunizieren. Durch Nutzung der Vorteile der Kommunikationstechnologien zwischen Fahrzeugen verwende ich eine Methode, die ich als „gruppenorientierte Fahrmethode“ bezeichne, um die Fahrzeuge zu koordinieren, so dass sie nicht einander blockieren. Dadurch können die Fahrzeuge ihre gewünschten Geschwindigkeiten erreichen. Die Fahrzeuge werden autonom von Computerprogrammen gesteuert.

Die gruppenorientierte Fahrmethode kann informell in zwei Phasen beschrieben werden. In der ersten Phase werden die autonomen Fahrzeuge, die nur kleine Unterschiede in den gewünschten Geschwindigkeiten haben, zu verschiedenen Gruppen zusammengefasst. Wenn eine langsame Gruppe (die Gruppe, deren Mitglieder sich langsame Geschwindigkeit wünschen) eine schnelle Gruppe blockiert, dann wird die zweite Phase durchgeführt. In der zweiten Phase werden die Straßenspuren von Gruppenführern ausgewählt. Die Gruppenmitglieder werden über die gewählten Spuren informiert. Die Mitglieder können entscheiden ob sie auf ihren Gruppenspuren fahren sollen.

Angenommen, dass die PKW- und LKW-Fahrzeuge des obigen Beispiels die gruppenorientierte Fahrmethode verwenden und zwei Gruppen (eine Gruppe von PKW-Fahrzeugen und eine Gruppe LKW-Fahrzeugen) bilden. Und weiterhin angenommen, dass die Gruppenspuren der PKW- und LKW-Fahrzeuge sind $[1, 2]$ bzw $[3]$, dann könnten sich folgende Vorteile ergeben:

Minimierte Blockierungszeit: werden die Fahrzeuge einer Gruppe auf ihre Gruppenspuren gezwungen, dann müssen die Fahrer $1, 2, 4$ ihre Geschwindigkeit umstellen um zur Spur 3 zu wechseln. Dies verzögert zwar den Verkehrsfluss in kurzer Zeit aber so können alle Fahrzeuge ihre gewünschte Geschwindigkeit fahren.

Vermeidung unnötiger Blockierung: Weil der LKW-Fahrer 4 schon so koordiniert wird, das er auf der Spur 3 fährt, blockiert er nicht die PKW-Fahrer auf der Spur 1.

Das Ziel der Arbeit besteht darin herauszufinden ob die autonomen Fahrzeuge, die meine Methode verwenden, schneller als die bemannten Fahrzeugen zum Ziel fahren können.

2 Relevante Arbeiten

2.1 Einleitung

In diesem Kapitel stelle ich die zu meiner Arbeit relevanten Forschungsbereiche vor. Wegen des großen Umfanges der Bereiche werde ich nur einige Arbeiten aus jedem Bereich vorstellen. Der Rest des Kapitels ist wie folgt aufgebaut. Im Abschnitt 2.2 gebe ich einen Überblick über die Anwendungen der Gruppenbildungstechnik im Straßenverkehr. Die Agententechnologie zur Konstruktion der autonomen Fahrzeuge ist ein Bestandteil meiner Arbeit. Aus diesem Grund beschreibe ich im Abschnitt 2.3 die Grundlagen der Agententechnologie und verschiedene Agentenstrukturen. In dieser Arbeit verwende ich ein Multiagentensystem (MAS) zur gleichzeitigen Steuerung mehrerer autonomer Fahrzeuge. Die Grundlagen der MAS und die wichtigsten Werkzeuge zu deren Konstruktion beschreibe ich im Abschnitt 2.4. Die Verhaltensweisen von Verkehrsteilnehmern wie Folgeverhaltensweise und Spurwechselverhaltensweise spielen eine wichtige Rolle bei der Konstruktion des Verhalten von autonomen Fahrzeugen. Im Abschnitt 2.5 werden verschiedene Fahrzeugfolgemodelle, welche die Folgeverhaltensweise von Verkehrsteilnehmern beschreiben, vorgestellt. Die Spurwechselverhaltensweise wird durch die Spurwechselmodelle im Abschnitt 2.6 präsentiert.

Um einen Vergleich zwischen der gruppenorientierten und der normalen Fahrmethode durchzuführen, verwende ich ein Verkehrssimulationssystem. Dieses Verkehrssimulationssystem simuliert das reale Verhalten von bemannten Fahrzeugen. Der Abschnitt 2.7 gibt einen Überblick über die bekannten Verkehrssimulationssysteme.

2.2 Gruppenbildung im Straßenverkehr

Die Bildung von Fahrzeuggruppen zur Verbesserung des Straßenverkehrs ist ein neuer Forschungsbereich des ITS. Ein Experiment mit realen bemannten Fahrzeugen des PATH-Projektes [33] zeigt, dass das Fahrzeuge, die in einer Gruppe fahren bis zu 7 % weniger Benzin verbrauchen als frei fahrende Fahrzeuge. Die Arbeit von Varaiya [45] zeigt, dass die Abstände zwischen Fahrzeugen einer Gruppe reduziert werden können ohne die Kollisionsgefahr zu erhöhen, wenn die Fahrzeuge in Gruppen fahren. Dadurch werden die Kapazitäten von Straßen erhöht. Damit die Fahrzeuge in Gruppen fahren können, sind Methoden zur Bildung und Aufrechterhaltung von Fahrzeuggruppen nötig. In diesem Abschnitt stelle ich einige bekannte dynamische Gruppenbildungstechniken und ihre Anwendungen im Straßenverkehr vor.

2.2.1 Gruppenbildung zur Verringerung des Benzinverbrauches und zur Erhöhung der Straßenkapazität

Die Ergebnisse des Experimentes des PATH-Projektes und der Arbeit von Varaiya motivieren den Forschungsbereich zur Konstruktion von autonomen Fahrzeugen, die in der Lage sind, verschiedene Gruppen zu formieren und zusammen durch Straßen zu navigieren.

Hall und Chin [23] versuchen Fahrzeuggruppen so zu bilden, dass die Fahrzeuge einer Gruppe so lange wie möglich zusammenbleiben. Um ihr Ziel zu erreichen wurde eine Methode entwickelt, die die Fahrzeuge entsprechend ihren Zielen in Gruppen klassifizieren. Die Fahrzeuge mit den entfernteren Zielen sind einer Gruppe und die Fahrzeuge mit den näheren Zielen einer anderen Gruppe zugeordnet. Hall und Chin betrachten die Autobahnen mit Einfahrten und Ausfahrten. Die Ziele der Fahrzeuge sind die Ausfahrten. Jede Ausfahrt wird genau einer Gruppe zugewiesen. Die Lebensdauer einer Gruppe wird durch eine Strecke y definiert. Das heißt, die Fahrzeuge einer Gruppe fahren zusammen bis sie einen Punkt A erreichen. An diesem Punkt werden die Fahrzeuge ihre Gruppe verlassen und zu den Spuren wechseln, auf denen sie ihre Ausfahrten erreichen. Der Bildungsprozess findet an den Einfahrten der Straße statt. Das Ziel der Arbeit von Hall und Chin ist die Fahrzeuge zur Gruppe zu ordnen, so dass die Strecke y maximal wird. Die gebildeten Fahrzeuggruppen erfordern keine Gruppenführer. Hall und Chin betrachten keine unterschiedliche Geschwindigkeiten der Fahrzeuge. In ihrer Arbeit haben alle Fahrzeuge die gleichen Geschwindigkeiten, Verzögerungs- und Beschleunigungs-Kapazitäten.

SARTRE [5, 44, 27] ist ein Projekt der Europa-Kommission FP7, welches darauf abzielt auch die autonomen Fahrzeuggruppen zu konstruieren. Im Gegensatz zur Arbeit von Hall und Chin bilden sich die Fahrzeuge von SARTRE ihre Gruppen während der Fahrt auf Autobahnen. Eine Fahrzeuggruppe von SARTRE besteht aus einem Gruppenführer und den Gruppenmitgliedern. Der Gruppenführer ist ein Lastkraftwagen oder ein Bus, welcher von einem professionellen Fahrer gesteuert wird und fährt immer an der Front der Gruppe. Der Fahrer eines Mitgliedsfahrzeuges hat zwei Fahrmodi. Im manuell gesteuerten Modus, wird das Fahrzeug wie alle normalen Fahrzeuge vom Fahrer gesteuert. Im autonomen Modus übernimmt das Fahrzeug die Steuerung und fährt von selbst. Die Abbildung 2.1 zeigt eine Fahrzeuggruppe von SARTRE. Ein Fahrzeug von SARTRE kann selbst oder mit Hilfe von Infrastrukturen, welche als Back-Office bezeichnet werden, an einer Gruppe teilnehmen. Jedoch wurde nicht beschrieben, wie die Gruppenteilnahme- und Verlassen-Operationen eines Fahrzeuges durchgeführt werden.

Halle [24] entwickelte ein System für die kollaborative Fahrt zwischen Fahrzeugen. Seine Arbeit konzentriert sich auf die Entwicklung von Koordinationsmethoden für die Fahrzeuge innerhalb einer Gruppe (interne Koordination) und für Fahrzeuggruppen untereinander (externe Koordination). Analog zur Fahrzeuggruppenstruktur von SARTRE definiert Halle eine Gruppe mit einem Gruppenführer und den Gruppenmitgliedern. Der Gruppenführer fährt vorne und die Gruppenmitglieder folgen ihm. Die Koordinationsmethoden beschreiben wie die Gruppenoperationen, zum Beispiel Spurwech-

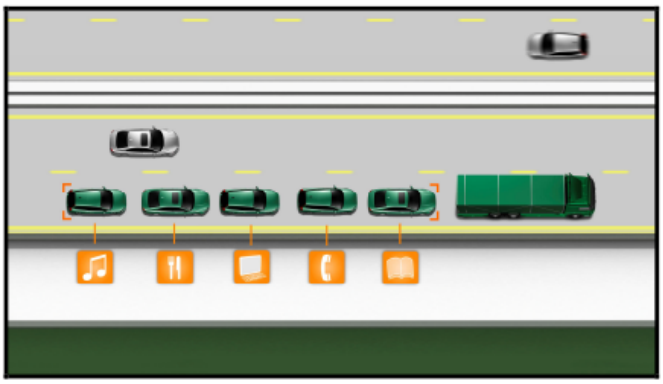


Abbildung 2.1: Eine Fahrzeuggruppe von SARTRE [44]. Der Gruppenführer ist ein LKW und die kleinen Fahrzeuge sind Gruppenmitglieder. Die Fahrer der kleinen Fahrzeuge lassen ihre Fahrzeuge autonom ihrem Vorgänger folgen. Während der autonomen Fahrt haben die Fahrer Zeit für ihre persönliche Arbeit (z.B. Anrufen, Zeitung lesen oder Musik hören)

sel, Gruppenteilnahmen und Gruppen-Verlassen, von Fahrzeugen durchführt werden. Jeder Koordinationstyp (interne Fahrzeugkoordination oder externe Gruppenkoordination) kann zentral oder dezentral durchgeführt werden. Die zentrale Koordination der Fahrzeuggruppen erfolgt durch die Verkehrsinfrastruktur am Straßenrand. Es wird angenommen, dass ein Verkehrskontroll-System am Rand einer Straße existiert. Das System gibt die Kontrollbefehle an geeignete Fahrzeuge zur Durchführung der Gruppenoperationen. Bei der dezentralen Koordination der Fahrzeuggruppen stellt Halle die Blackboard-Methode vor. Diese Koordinationsmethode basiert auf einer sogenannte virtuellen Schwarztafel (Blackboard). Es wird angenommen, dass jede Fahrzeuggruppe eine virtuelle Schwarztafel hat. Jeder Agent einer Gruppe kann mit Hilfe der Tafel ihre eigenen Informationen publizieren und die Tafeln anderer Gruppen lesen um Informationen zu gewinnen. Die Koordination der Fahrzeuggruppen erfolgt durch die Festlegung der Kosten für jede Fahrzeuggruppe. Wenn zwei Gruppen ihre Gruppenoperationen durchführen wollen, jedoch wegen Kollisionsgefahr nur eine Gruppe ihre Operation durchführen darf, dann schreiben die beiden Gruppen ihre Kosten jeweils auf ihre Tafel. Die Gruppe mit dem höheren Kosten darf ihre Operationen zuerst durchführen. Bei der zentralen Koordination der Fahrzeuge einer Gruppe wird der Gruppenführer die Rolle als Koordinator übernehmen. Der Koordinator schickt Kontrollbefehle an geeignete Mitglieder um die Gruppenoperationen durchzuführen. Im Gegensatz dazu wird die dezentrale Koordination der Fahrzeuge innerhalb einer Gruppe von Mitgliedern durchgeführt. Halle nimmt an, dass jedes Mitglied seine auszuführenden Gruppenoperationen kennt. Dieses Wissen wird initialisiert wenn ein Agent an einer Gruppe teilnimmt. Wenn eine Gruppenoperation durchgeführt werden muss, dann reagieren die Mitglieder autonom auf diese Operation und kommunizieren untereinander um sich selbst zu koordinieren. Dadurch wird die Gruppenoperation ausgeführt.

2.2.2 Gruppenbildungstechniken zur Vermeidung der potentiellen Kollision

Zur Vermeidung von potentiellen Kollisionen muss ein autonomes Fahrzeug bestimmen, mit welchen Fahrzeugen es eine Gruppe bilden sollte. Frese und Beyerer [14] präsentieren in ihrer Arbeit eine Methode zur Bildung einer kooperativen Fahrzeuggruppe. Das Ziel dieser kooperativen Gruppe ist das Aufstellen eines gemeinsamen Lagebildes, das alle relevanten Informationen über die Fahrzeuge der Gruppe enthält. Mit Hilfe dieses Lagebildes können die Fahrzeuge ihre gemeinsame Entscheidung treffen um kooperatives Verhalten durchzuführen. Frese und Beyerer konstruieren eine Bewertungsfunktion für die Gruppenbildung. Durch Optimierung dieser Funktion werden die Fahrzeuge in Gruppen klassifiziert, so dass die Summe der Abstände zwischen Mitgliedern einer Gruppe minimal ist.

2.2.3 Andere Anwendungen der Gruppenbildung

Astengo-Noguez und Brena-Pinero [3] stellen eine Methode zur Koordination der Fahrzeuggruppen an Kreuzungen vor. Astengo-Noguez und Brena-Pinero nehmen an, dass es keine Straßenampeln zur Koordination der Fahrzeuge gibt. Nach dieser Methode kommen die Fahrzeuge als Gruppe an eine Kreuzung. Dort hat die größte Fahrzeuggruppe Vorfahrt. Weil die Koordination auf Gruppenebene stattfindet wird behauptet, dass diese Methode die durchschnittliche Geschwindigkeit von Fahrzeugen erhöht. Um die Fahrzeuggruppen zu bilden versuchten Astengo-Noguez und Brena-Pinero die drei folgenden Hauptfragen zu beantworten:

- Wie kann ein Fahrzeug einen geeigneten Partner für die Gruppenbildung finden?
- An welcher Postion sollen die Fahrzeuge ihre Gruppe bilden und an welcher Position soll die Gruppe aufgelöst werden?
- Wie werden die Gruppen koordiniert um Kollisionen zwischen ihnen zu vermeiden?

Für jedes Fahrzeug werden zwei Komfort-Werte berechnet und verglichen. Der erste Wert repräsentiert den Komfort eines Fahrzeuges wenn es allein fährt. Der zweite Wert repräsentiert den Komfort eines Fahrzeuges wenn es gemeinsam mit anderen Fahrzeugen fährt. Ist der zweite Komfort-Wert größer als der erste, dann entscheidet das Fahrzeug mit anderen Fahrzeugen eine Gruppe zu bilden.

2.3 Agententechnologie

Es gibt keine allgemeingültige Definition von Agenten. Wooldridge und Jennings [49] unterscheiden zwei generelle Nutzungen des Begriffes „Agent“. Die erste Nutzung ist relativ unstrittig und betrifft die Hardware oder die Computersysteme mit folgenden

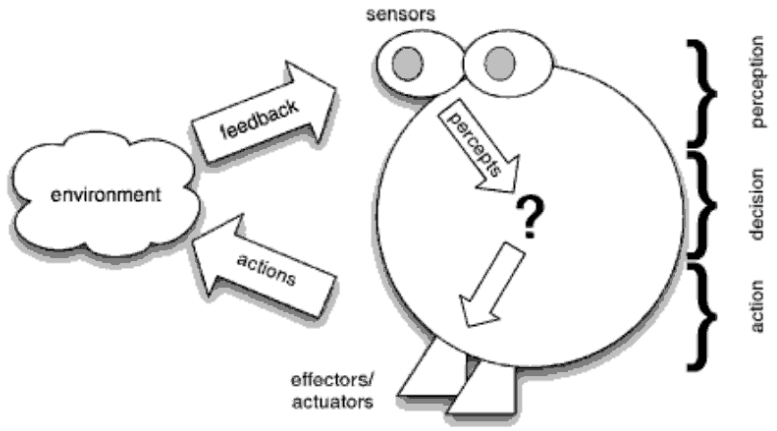


Abbildung 2.2: Ein Agent erkennt die Umgebung durch seinen Sensor und beeinflusst die Umgebung durch seinen Effektor [50]

Eigenschaften: *Autonomie, Sozialkompetenz, Reaktionsfähigkeit, zielgerichtete Aktivität (Proactiveness)*. Die zweite umstrittene Nutzung des Begriffes „Agent" betrifft Computersysteme mit folgenden zusätzlichen Eigenschaften: *Mentalität, Emotionalität* . Im Allgemeinen wird die folgende Agentendefinition von vielen Forschern benutzt:

„Agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its delegated objectives" [50].

Die Abbildung 2.2 beschreibt eine abstrakte Struktur eines Agenten. In den letzten Jahrzehnten wurden verschiedene Agentenarchitekturen vorgestellt. Die Auswahl einer geeigneten Agentenarchitektur ist von den Aufgaben und den Zielen der Agenten abhängig. In diesem Abschnitt stelle ich drei generelle Architekturen vor: reaktive Architektur, deliberative Architektur und Schichtenarchitektur.

2.3.1 Reaktive Architektur:

Agre, Chapman [1], Brooks [7] beschreiben eine Klasse von Agenten, die ihre Entscheidungen mit nur sehr wenigen Informationen treffen. Diese Agenten werden als reaktive Agenten bezeichnet. Ein reaktiver Agent verfügt über keine symbolische Abbildung der Umgebung. Die Durchführung seiner Aktion wird durch simple Regeln definiert. Eine Regel könnte beispielsweise wie folgt lauten:

```
Wenn (Bedingung ist erfüllt) Dann mache etwas
```

Pengi - Agre und Chapman

Agre und Chapman [1] behaupten, dass die Entscheidung von Agenten eine Aktion durchzuführen meistens von dem aktuellen Zustand der Umgebung abhängt. In der Realität dauert es manchmal zu lang und ist es zu kompliziert um eine sofortige Aktion zu planen. In solchen Situationen führt man seine Aktion unbewusst durch. Zum

Beispiel, wenn man auf einen Stein tritt und seine Balance verliert, reagiert man geeignet um seine Balance wieder zu gewinnen.

Agre und Chapman implementieren das Spiel „Pengo“. Darin versuchen die Pinguine vor den Killer-Bienen zu flüchten. Das Spiel wird folgendermaßen geregelt: Wenn eine Biene nah an einen Pinguin herankommt, tötet sie den Pinguin. Auf dem Spielfeld gibt es viele Eisblöcke. Die Bienen und die Pinguine können die Eisblöcke kicken damit sie zur Gegenrichtung rutschen. Wenn ein Block über eine Biene oder einen Pinguin rutscht, stirbt die Biene bzw. der Pinguin. Die Aufgabe der Pinguine im Spiel besteht darin die magischen Blöcke zu sammeln um ein magisches Quadrat zu bauen. Wenn das magisches Quadrat fertig ist, gewinnen die Pinguine das Spiel. Weil sich die Pinguine und die Bienen ständig bewegen, wird der Zustand der Umgebung dynamisch geändert. Das Spiel hat gezeigt, dass es in so einer dynamischen Umgebung für einen Agenten (einen Pinguin) nicht effizient ist seine Aktion nach einem Plan durchzuführen. Zum Beispiel plant ein Pinguin p eine Reihe von Aktionen um von einer Biene b wegzulaufen und führt eine geplante Aktion nach den anderen durch. Jedoch stirbt die Biene b wegen eines rutschenden Eisblockes eines anderen Pinguins. Weil der Pinguin p eine Reihe von Weglauf-Aktionen schon geplant hat, führt er immer noch die geplanten Aktionen durch, obwohl es nicht nötig ist. Zur Konstruktion von reaktiven Pinguinen ließen Agre und Chapman ihre Pinguine mit der Welt durch vordefinierte *Routinen* interagieren. Eine Routine ist die Aktion, die in bestimmten Situationen ausgeführt wird. Zum Beispiel wenn ein Pinguin von einer Biene verfolgt wird, läuft er weg. Mit dieser Methode wird der Pinguin seine Weglauf-Aktion unverzüglich beendet, wenn seine verfolgende Biene stirbt. Dadurch wird keine unnötige Aktion durchgeführt. Jedoch hat diese Methode auch Nachteile. Zum Beispiel kann ein Pinguin einen schnellsten Weg nicht selbst verfolgen um einen magischen Block zu finden [34].

Inkrementale Intelligenz - Brooks

Brooks [7] behauptet, dass ein Agent keine Abbildung der realen Welt braucht. In seiner Arbeit stellt er die sogenannte „inkrementale Intelligenz“-Methode vor. Die Methode ermöglicht die Konstruktion eines komplexen intelligenten Systems durch viele kleinere Subsysteme. Jedes Subsystem ist für eine Aktivität verantwortlich. Die Aktivität wird nur durchgeführt, wenn bestimmte Bedingungen erfüllt werden. Das Subsystem wird als „Ebene“ bezeichnet. Der Vorteil dieser Methode liegt darin, dass beim Einfügen von neuen Ebenen man ein sehr einfaches System zu einem intelligenten komplexen System erweitern kann. Alle Aktivitäten können parallel ausgeführt werden. Zur Demonstration seiner Methode stellte Brooks ein „Kreatur“-Robot vor. Die Kreatur hat die folgenden funktionalen Anforderungen: sie muss die dynamische Änderung der Umgebung bewältigen. Sie muss robust in Bezug auf ihre Umgebung sein. Sie muss in der Lage sein verschiedene Ziele zu verfolgen. Sie muss einen Existenzgrund haben.

Die Kreatur hat drei Ebenen für die folgenden Aktionen: Kollision vermeiden (erste Ebene), wandern(zweite Ebene) und erkunden (dritte Ebene). Die Bewegung der Kreatur wird wie folgt beschrieben: Die Kreatur wandert zu einem Ziel. Die zweite Ebene ist für diese Aktion zuständig. Während der Wanderung wird ein Hindernis von der er-

sten Ebene entdeckt. Die erste Ebene lenkt die Kreatur um das Hindernis zu umgehen. Das heißt, der Motor der Kreatur wird von zwei Ebenen gesteuert. Wenn die Kreatur das Hindernis umgangen hat, hört die erste Ebene auf den Motor zu kontrollieren. Der Motor wird dann nur von der zweiten Ebene gesteuert. Die Entwicklung jeder Ebene ist zeitaufwändig. Es wurde in der Arbeit nicht beschrieben, wie zwei Kreaturen mit einander kooperieren um ein Ziel zu erreichen.

2.3.2 Deliberative Architektur

Wooldridge und Jennings definieren einen deliberativen Agenten wie folgt:

„A deliberative agent or agent architeture to be one that contains an explicity represented, symbolic model of the world, and in which decisions(for example about what actions to perform) are made via logical(or at least pseudo-logical) reasoning, based on pattern matching and symbolic manipulation.“ [49]

Im Gegensatz zu den reaktiven Agenten verfügen beliberative Agenten über die symbolischen Abbildungen der realen Welt. Basierend auf den symbolischen Abbildungen, werden die Entscheidungen (z.B. Entscheidung über ausführende Aktionen) getroffen. Der Prozess der Entscheidungsfindung wird als „Inferenz“ bezeichnet.

BDI-Agent von Bratman

Bratman [6] betrachtet drei mentale Attitüden: „Beliefs(B)“, „Desires(D)“ und „Intensions(I)“ eines rationalen Agenten.

Die erste Attitüde „Beliefs (Ansichten)“ steht für die Ansichten des Agenten bezüglich des aktuellen Zustandes der realen Welt und des nächsten Zustandes der Welt wenn eine bestimmte Aktion ausgeführt wird.

Die zweite Attitüde „Desires(Wünsche)“ steht für die gewünschten künftigen Zustände der Welt oder die gewünschten Ergebnisse der Aktionen eines Agenten. Rao und Georgeff [39] differenzieren „Desires“ und „Goals(Ziele)“ wie folgt: Die Wünsche eines Agenten können inkonsistent sein und die Ziele eines Agenten sind die gewählten konsistenten Wünsche. Ein Beisiel von Müller [34] verdeutlicht diese Differenz. Ein Agent wünscht sich fliegen zu können (Aktion) oder reich zu werden (Zustand). Jedoch glaubt er nicht, dass er fliegen kann oder reich wird. Solche gewünschten Aktionen und Zustände werden als inkonsistente Wünsche bezeichnet. Dagegen wenn ein Agent sich ein Buch wünscht und er das Buch kaufen kann, wird der Wunsch in diesem Fall als Ziel bezeichnet.

Die dritte Attitüde „Intensions(Absichten)“ steht für die gewählten Ziele, die ein Agent letztlich erreichen will. Damit der Agent seine „Intensions“ verwirklichen kann, muss er einen Plan haben. Ein Plan ist eine Reihe von Aktionen, die nacheinander ausgeführt werden um den Agenten von seinem aktuellen Zustand zu den gewünschten Zuständen zu führen. Zum Beispiel hat ein Agent nur dann die Absicht ein Buch zu kaufen, wenn er glaubt, dass er genug Geld (Belief) auf seinem Konto hat. Er erstellt den folgenden Plan für seine Absicht. Zunächst muss er das Geld von Automaten abheben. Danach geht er zu einem Buchladen und kauft sein gewünschtes Buch.

Basierend auf der BDI Theorie von Bratman wurden verschiedene Agentenstrukturen entwickelt. Im folgenden beschreibe ich die BDI Theorie konkreter durch Formalisierung der BDI-Theorie von Rao und Georgeff und die PSR Architektur [17] von Georgeff und Lansky.

PRS - Georgeff und Lansky

Basierend auf der BDI-Theorie konstruieren Georgeff und Lansky [17] das „Procedural Reasoning System(PRS)“ zur Inferenz und Ausführung von komplizierten Aufgaben. Eine Aktion des PRS wird durch drei Komponenten: die Ansicht über die Welt, die Ziele des Systems und die vorher gewählten Absichten begründet. Mit dieser Architektur erlaubt ein PRS nicht nur die geplanten Aktionen sondern auch die reaktiven Aktionen zu argumentieren. Das PRS-System besteht aus einer Datenbank von Prozeduren, Zielen, Weltansichten und einem Interpreter.

Prozedur: [1] Eine Prozedur besteht aus einem Körper und einer Aufrufbedingung. Im Gegensatz zu den meisten künstlichen Intelligenz-Planern die eine mögliche Reihenfolge von primitiven Aktionen planen um ein übergeordnetes Ziel zu erreichen, beschreibt eine Prozedur eine Reihenfolge von unteren Zielen, die das System erreichen muss um das übergeordnete Ziel zu erreichen. Die unteren Ziele können entweder durch andere Prozeduren erreicht werden oder direkt durch Ausführung von primitiven Aktionen. Die Aufrufbedingung der Prozedur spezifiziert die Situationen für die die Prozedur nützlich (anwendbar) ist.

Ziele: Die Ziele eines PRS befinden sich in einem Zielstapel des Systems und in der Repräsentation der Prozedur. Anders als die meisten AI-Planungssysteme, beschreiben die Ziele des PRS eher das gewünschte Verhalten des Systems und weniger die statischen Zuständen, die das System erreichen will. Ein Zielverhalten ist ein Aktionstyp oder eine Reihe von Zuständen des Systems. Zum Beispiel bezeichnet das Ziel *„gehen von A nach B“* eine Reihe von Zuständen des Systems und stellt auch die Aktion *„gehen“* dar.

Interpreter: Der Interpreter spielt die Hauptrolle bei der Auswahl von Zielen und Prozeduren. Basierend auf den aktuellen Ansichten und den aktuellen Zielen wird eine der relevanten Prozeduren vom Interpreter ausgewählt und in einen Prozessstapel eingefügt. Die neuen unteren Ziele (Sub-Goals) der gewählten Prozedur werden extrahiert und in einen Zielstapel eingefügt. Der Interpreter prüft weiter, ob es noch weitere Prozeduren gib, die für die neuen unteren Ziele anwendbar sind. Wenn die Prüfung positiv ist (es existieren Prozeduren für die neuen unteren Ziele), dann wählt der Interpreter wieder eine Prozedur aus und fügt die unteren Ziele der neu gewählten Prozedur in den Zielstapel ein. Der Prozess läuft so lange bis keine Prozedur mehr gefunden wird oder die unteren Ziele direkt durch primitive Aktionen erreicht werden. Die Änderungen der Umgebung verursachen Änderungen der Ansichten des Systems. Nach jeder Ansichten-Änderung führt der Interpreter eine entsprechende Wartung durch um die Konsistenz der Prozeduren zu prüfen und wenn möglich neue relevante Proze-

[1] Aus historischen Gründen wird eine Prozedur meistens als *„Knowledge Area(KA)“* bezeichnet

duren zu aktivieren. Es kann vorkommen, dass die Aktivierung einer neuen Prozedur eine totale Änderung der Ziele des PRS verursacht. In diesem Fall werden die alten Prozeduren von dem Prozessstapel gelöscht und das System verfolgt die neuen Ziele.

Formale BDI Architektur - Rao und Georgeff

Im Jahr 1991 formalisierten Rao und Georgeff [39] die BDI-Theorie von Bratman. Im Gegensatz zu anderen philosophischen Theorien [2] haben die Absichten eines Agenten den gleichen Status wie Ansichten und Wünsche und können nicht auf Ansichten und Wünsche reduziert werden. Rao und Georgeff modellieren die Welten durch einen Zeit-Baum. Eine Welt zum Zeitpunkt wird als *Situation* bezeichnet. Ausgehend von einer Situation werden die weiteren Welten durch eine Menge von Operatoren $\{BEL, GOAL, INTEND\}$ definiert. $BEL, GOAL, INTEND$ stehen für Ansichten, Ziele und Absichten von Agenten. Ein Operator fügt Welten zusammen und wird formal durch die Form $W \times T \times W$ definiert. Wobei W eine Menge von Welten ist und T eine Menge von Zeitpunkten ist. Zum Beispiel hat $BEL : w1 \times t \times w2$ folgende Bedeutung: Ein Agent befindet zum Zeitpunkt t in der Welt $w1 \in W$ und glaubt, dass eine zukünftige Welt $w2$ existiert. Die Welt $w2$ wird als zugängliche Ansichten-Welt bezeichnet. Die weiteren Welten werden wie folgt beschrieben:

Zugängliche Ansichten-Welten (belief-accessible worlds) sind die Welten, die ein Agent als möglich festlegt.

Zugängliche Ziel-Welten(goal-accessible wolds) repräsentieren die Ziele von Agenten. Rao und Georgeff fordern, dass die Ziel-Welten konsistent sind. Das heißt, Agenten müssen glauben, dass die Ziel-Welten für sie erreichbar sind. Diese Eigenschaft wurde von Cohen und Levesque [9] als „Realismus“ bezeichnet. Rao und Georgeff beschreiben einen starken Realismus wie folgt: Für jede zugängliche Ansichten-Welt $b1$ zum Zeitpunkt t gibt es eine zugängliche Ziel-Welt $g1$ zum Zeitpukt t als Sub-Welt von $b1$ und nicht umgekehrt. Das heißt, es gibt auch die Ziel-Welt die keine Ansichten-Welt ist.

Zugängliche Absicht-Welten(intention-accessible worlds) sind die Welten, welche von Agenten als zu realisieren festgelegt werden. Es wird definiert, dass jede zugängliche Ziel-Welt $g1$ eine zugängliche Absicht-Welt $i1$ als ihre Sub-Welt hat, aber eine zugängliche Absicht-Welt zu keiner zugänglichen Ziel-Welt gehören kann. Die Aktion zur Auswahl einer Welt als Absicht-Welt wird als „Commit“ bezeichnet. Rao und Georgeff definieren drei „commit“-Strategien: „blind commit“, „single minded commit“ und „open minded commit“. Ein „blind commit“-Agent verfolgt seine Absicht bis er sie letztlich erreicht. Ein „single minded“-Agent verfolgt seine Absicht solange er sie noch für erreichbar hält. Ein „open minded“-Agent verfolgt seine Absicht solange die Absicht noch sein Ziel ist. Das heißt, er hält an der Absicht fest und glaubt, dass die Absicht für ihn noch erreichbar ist.

Ereignisse verbinden zwei Zeitpunkte. Rao und Georgeff unterschieden zwischen primitiven Ereignissen und nicht-primitiven Ereignissen[3]. Die primitiven Ereignisse

[2] Zum Beisiel der Theorie von Cohen und Levesque [9], werden Absichten als fundamentale Attitüde betrachtet.

[3] Die Aufteilung verweist auf die Arbeit von Cohen und Levesque in [9]

sind die von dem Agenten erzeugenden Ereignisse (z. B. Aktionen durchführen). Die Nicht-primitiven Ereignisse sind die von der Umgebung generierten Ereignisse. Diese Aufteilung erlaubt es die Welt natürlicher zu modellieren. Bei der Auswahl eines Ereignisses kann ein Agent seine künftige Welt bestimmen. Jedoch endet ein Ereignis (Aktion) mit verschiedenen Zuständen. Das heißt, die tatsächlich aus einem Ereignis resultierende Welt wird von der Umgebung definiert. Rao und Georgeff definierten die Zustände eines Ereignisses e durch die Formeln $succeeded(e)$, $failed(e)$, $done(e)$. Wobei $succeeded(e)$, $failed(e)$, und $done(e)$ für erfolgreiche, erfolglose und geschehene Endzustände des Ereignisses e stehen.

2.3.3 Schichtenarchitekturen

Die Schichtenarchitekturen versuchen die Vorteile der reaktive-Aktion-basierten Architektur und der geplante-Aktion-basierten Architektur zu kombinieren. Jede Schicht ist für eine Aktionsart (reaktive Aktion oder geplante Aktion) zuständig. Es gibt zwei typische Designs der Schichten. Beim ersten Design werden die Schichten horizontal organisiert. Die horizontale Organisation lässt alle Schichte ihre Aktionen generieren. Die tatsächlich ausgeführte Aktion wird von einer Hauptkontrolle entschieden. Die typische horizontal Schichtenarchitektur verweist auf die Arbeit von Ferguson [13]. Das zweite Design ist die vertikale Organisation von Schichten. Bei der vertikalen Organisation wird die Suche nach einer geeigneten Aktion von einer Schicht zu einer anderen Schicht delegiert. Die Delegation wird beendet, wenn eine Aktion gefunden wird. Eine typische vertikal Schichtenarchitektur wurde von Müller und Pischel im Jahr 1993 vorgestellt. Im folgenden stelle ich die InteRRaP -Architektur von Müler und Pischel vor.

InteRRaP Agent - Müller und Pischel

Müller und Pischel [35] stellen die „InteRRaP " Architektur vor. InteRRaP ist eine Hybrid-Agentenarchitektur. Sie kombiniert die reaktive Architektur und die deliberative Architektur in einer Schichtenarchitektur. Deshalb kann ein InteRRaP -Agent flexibel auf die Änderungen der Umgebung reagieren und komplexe geplante Aufgaben durchführen. Eine zusätzliche Funktion der InteRRaP ist die Fähigkeit mit anderen Agenten zu kooperieren. Die InteRRaP -Struktur besteht aus vier Schichten (siehe Abbildung 2.3).

Die erste Schicht „*World Interface(WI)*" ist eine Schnittstelle zwischen dem Agenten (der die InteRRaP Struktur benutzt) und der Umgebung. Die WI übernimmt verschiedene Rollen.

1. Rolle als Sensors: Die WI erkennt die Änderungen der Umgebung und speichert sie in einer „*World Model*"-Datenbank.

2. Rolle als Kommunikator: Alle Kommunikationsnachrichten zwischen Agenten werden über die WI transferiert.

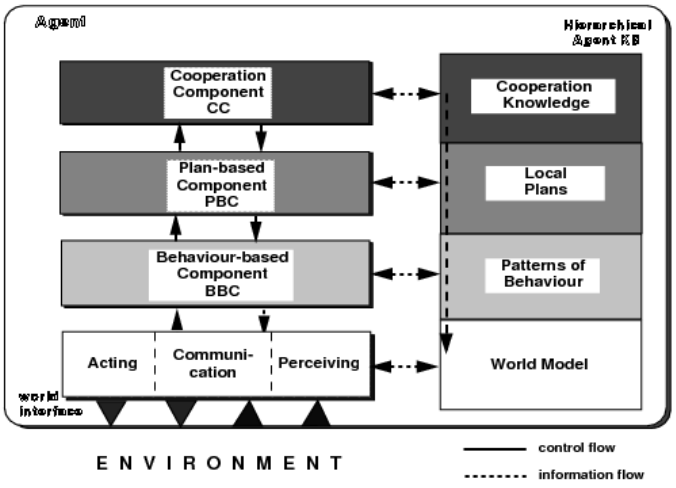


Abbildung 2.3: Das InteRRaP Agentenmodell [35]

3. Rolle als Aktionsexekutor: Ein InteRRaP -Agent benutzt die WI um die physikalische Umgebung zu beeinflussen.

Die zweite Schicht wird als „Behaviour-based Component(BBC)“ bezeichnet. BBC implementiert das Routineverhalten des Agenten, entscheidet über das auszuführenden Verhalten und führt es aus. Das Routineverhalten umfasst nicht nur die reaktiven Aktionen des Agenten selbst, sondern auch die Interaktionen zwischen Agenten. Um das Verhalten des Agenten zu beschreiben, stellen Müller und Pischel das „Pattern of Behaviour(PoB)“-Modell vor. Eine PoB beschreibt ein Routineverhalten des Agenten (z.B eine Aktion oder Interaktion) und kann von dem BBC oder unter bestimmten externen Bedingungen aktiviert werden. Das heißt, in bestimmten Situationen können zwei oder mehrere PoB aktiviert werden. Die wichtigste Komponente des BBC ist die BBC-Kontrolle. Die BBC-Kontrolle koordiniert die PoB und entscheidet welche PoB an einem bestimmten Zeitpunkt ausgeführt werden sollen. Der Agent, welcher die InteRRaP Struktur implementiert, verfolgt immer ein Ziel. Für den Fall, dass kein PoB von der Situation aktiviert wird, leitet die BBC das Ziel zu der obereren Schicht(PBC) weiter um einen Plan zu generieren.

Die dritte Schicht ist die „Plan-based Component (PBC)“. Die PBC hat folgende Funktionen: einen Plan für ein Ziel erstellen aber nicht ausführen, verschiedene Pläne bewerten, einen gegebenen Plan interpretieren. Die PBC besteht aus eine PBC-Kontrolle, einem Plan-Generator, einem Plan-Bewerter und einem Ressource-Handler. Die PBC-Kontrolle ist der „Kopf“ der PBC. Die Ziele, die die PBC von der unteren Schicht (BBC) bekommt, werden bei der PBC-Kontrolle verarbeitet. Die PBC-Kontrolle benutzt den Plan-Generator um verschiedene Pläne für den Agenten selbst (es ist zu unterscheiden zwischen den sozialen Plänen für mehrere Agenten und den Plänen für nur einen Agenten) zu generieren. Die generierten Pläne werden von dem Plan-Bewerter bewertet um den günstigsten Plan zu finden. Sobald ein Plan gefunden wird, wird er zurück

an die BBC gegeben. Falls der Plan-Generator kein Plan generieren kann, wird das Ziel weiter an die obere Schicht(CC) geleitet. Ein Plan hat eine hierarchische Struktur, deren Knoten wieder untere Pläne oder ausführbare PoB oder primitive Aktionen sind.

Die vierte Schicht ist die „Cooperation Component(CC)“. Die CC ist für Planung der gemeinsamen Pläne (joint Plan (JP)), die mehrere Agenten als Teilnehmer enthalten, zuständig. Wie die PBC hat auch die CC die unteren Komponenten: CC-Kontrolle, JP-Bewerter, JP-Übersetzer, Ressource-Handler. Die CC-Kontrolle spielt die selbe Rolle wie die PBC-Kontrolle der dritten Schicht. Sie ist nicht nur eine Schnittstelle für die Kommunikationen mit der unteren Schicht (PBC) sondern auch ein Koordinator der anderen Komponenten. Die Ziele, die der Agent allein nicht erreichen kann, werden an die CC-Kontrolle geleitet. Die CC-Kontrolle benutzt den JP-Generator und den JP-Bewerter um die JP-Pläne zu generieren und zu bewerten. Der beste JP-Plan wird gewählt. Weil ein JP-Plan dieser Schicht verschiedene Agenten als Teilnehmer hat, werden die unteren Pläne für den einzelnen Plan-Teilnehmer aus dem JP-Plan extrahiert. Der untere Plan des Agenten wird an die PBC geleitet.

Die InteRRaP ist eine flexible Struktur und kann in verschiedenen Bereichen verwendet werden. Sie ermöglicht die automatischen Problemlösungen mit Agententechnologien zu testen. Die Kooperationskomponente der InteRRaP erlaubt den Agenten die dezentralen Problemen gemeinsam zu lösen.

2.4 Multiagentensystem

Der Abschnitt 2.3 hat einen Überblick über verschiedene Agentenstrukturen gegeben. Die Forschung im Bereich „Artifical Intelligent (AI)“ konzentriert auf die Konstruktion des autonomen Agenten, welcher durch seine Verhaltensweise die Umgebung beeinflusst. Anhand der wahrgenommen Informationen aus der Umgebung führt der Agent eine geeignete Verhaltensweise durch, um ein bestimmtes Problem zu lösen oder ein Ziel zu erreichen. In vielen Problembereichen ist die Anwendung eines einzigen Agenten für die Problemlösungssuche wegen mangelnder Rechenleistung, dezentraler Speicherung von nötigen Informationen oder Beschränkung von Technologien jedoch nicht möglich.

Als Beispiel soll das folgende Szenario [46] betrachtet werden. Angenommen, ein Unternehmen X Reifen. Zur Produktion der Reifen wird die Radmutter eines anderen Unternehmens Y benötigt. Um ein einziges System, welches den automatischen Reifen-Produktionsprozess durchführt zu konstruieren, müssen die internen Produktionsprozesse von beiden Unternehmen X und Y modelliert werden. Jedoch will keines der beiden Unternehmen die Informationen und die Kontrolle über ihre Produktionsprozesse an das andere Unternehmen abgeben. Deswegen ist die Konstruktion eines automatischen Produktionsprozesses nicht möglich.

Ein anderes Beispiel ist der Straßenverkehr. Angenommen, die Fahrzeuge auf Straßen werden autonom von einem einzigen Agenten gesteuert. Das heißt alle Fahrzeuge werden zentral von einem Agenten koordiniert, so dass sie nicht miteinander kollidieren. Es ist erkennbar, dass die Konstruktion eines Agenten als Koordinator aller anderen

Agenten in der realen Welt nicht möglich ist. Der Grund liegt in der Beschränkung von Rechenleistung und Technologien. Für ein kleines Verkehrsszenario könnte die zentrale Koordination sinnvoll sein. Jedoch gibt es keinen Rechner der stark genug ist, um ein großes Szenario mit vielen Fahrzeugen zu koordinieren.

Um die obigen Beschränkungen zu überwinden, wird an einer Erweiterung von AI unter dem Namen „Distributed Artifical Intelligent (DAI)“ geforscht. Im Gegensatz zu dem Forschungsbereich der traditionellen künstlichen Intelligenz (AI), konzentriert sich DAI auf die Nutzung mehrerer Agenten für die Problemlösungssuche. Weiss [47] definiert DAI in seinem Buch wie folgt:

DAI ist the study, construction, and application of multiagent systems, that is, systems in which several interacting, intelligent agents pursue some set of goals or perform some set of tasks. [47]

Laut der Definition von Weiss, ist ein Multiagentensysem (MAS) ein System, in dem die Agenten mit einander interagieren um eine Menge von Zielen zu erreichen oder eine Menge von Aufgaben zu erfüllen.

Traditionell gibt es zwei Typen von DAI-Systemen: Multiagentensysteme und „Distributed Problem Solving (DPS)“-Systeme. Während sich ein DPS-System auf die Lösung eines bestimmten Problems durch Dekomposition und Synthese konzentriert, konzentriert sich ein MAS stärker auf die Interaktionen zwischen Agenten und die Koordination des Agentenverhaltens. Im Allgemeinen gibt es jedoch keine klare Trennung zwischen einem DPS-System und einem Multiagentensystem. Die Konstruktion eines Multiagentensystems erlaubt einem System-Designer bestimmte Annahmen für sein System und seine Agenten festzulegen. Durfee und Rosenschein [11] zeigen, dass ein Multiagentensystem als ein DPS-System betrachtet werden kann, wenn es die folgenden Annahmen erfüllt:

- Wohlwollen-Annahme: Es wird angenommen, dass alle Agenten des Multiagentensystems hilfsbereit sind. Sie helfen den anderen Agenten ohne sich zu fragen „Warum muss ich den anderen Agenten helfen ?“

- Gleiche-Ziele-Annahme: Es wird angenommen, dass alle Agenten die gleichen Ziele haben.

- Zentraler-Designer-Annahme: Diese Annahme ist eine Zusammenfassung der beiden obigen Annahmen. Weil das Ziel, das der zentrale Designer erreichen will, in allen Agenten eingesetzt wird, haben die Agenten ein gemeinsames Ziel. Weil sich der zentrale Designer auf das Endergebnis seines Zieles konzentriert, nimmt er tendenziell an, dass die Agenten Wohlwollen haben.

Die Eigenschaften eines Multiagentensystems bestimmen wie sich das System verhält und wie die Agenten miteinander interagieren. Im folgenden Abschnitt stelle ich die wichtigen Eigenschaften eines Multiagentensystems vor. Die Konstruktion eines Multiagentensystems erfolgt durch Bestimmung der Eigenschaften des Multiagentensystems und der Agenten.

2.4.1 Eigenschaften von Multiagentensystemen

Die allgemeinen Haupteigenschaften von MAS wurden von Weiss [47] wie folgt notiert:

- Jeder Agent hat nur beschränkte Informationen und limitierte Fähigkeiten.
- Die Kontrolle des Multiagentensysems in dem der Agent existiert, ist dezentral.
- Die Daten sind dezentral.
- Die Berechnung ist asynchron.

Zwei MAS unterschieden sich durch die Eigenschaften von Agenten, der Interaktion zwischen Agenten und der Umgebung, in der sich die Agenten befinden. Die Tabelle 2.1 gibt einen Überblick über einige Attribute von MAS und deren Ränge. Um ein Multi-

	Attribute	**Ränge**
	Anzahl	ab zwei
	Einheitlichkeit	einheitlich...heterogen
Agenten	Ziele	widersprechend...ergänzend
	Architektur	reaktiv...bewusst
	Fähigkeit	einfach...fortschrittlich
	Frequenz	niedrig...hoch
	Persistenz	kurzfristig...langfristig
	Niveau	Signal-Austausch...wissensintensiv
Interaktion	Muster (Informationsfluss und Kontrollfluss)	dezentral...hierarchisch
	Variabilität	fix...veränderlich
	Zweck	konkurrierend...kooperativ
	Vorhersehbarkeit	vorhersehbar...nicht vorhersehbar
	Zugänglichkeit und Wissen	beschränkt...nicht beschränkt
Umgebung	Dynamik	fix...veränderlich
	Vielfalt	arm...reich
	Verfügbarkeit von Ressourcen	eingeschränkt...reichlich

Tabelle 2.1: Attribute von Multiagentensystemen und ihre Ränge [47]

agentensystem zu konstruieren, müssen die Eigenschaften des Systems, der Agenten und der Interaktionen zwischen Agenten zunächst definiert werden. Weiss [47] schlägt die folgenden Fragen als besondere Herausforderungen der Konstruktion eines Multiagentensystems vor.

- Wie zerlegen die Agenten ihre Ziele und Aufgaben und verteilen untere Ziele und untere Aufgaben an anderen Agenten? Wie synthetisieren die Agenten die Ergebnisse und die Lösungen von anderen Agenten?
- Wie kommunizieren die Agenten miteinander? Welche Protokolle benutzen sie?
- Wie repräsentieren und argumentieren die Agenten die Aktionen, Pläne und das Wissen von anderen Agenten?

- Wie repräsentieren und argumentieren die Agenten die Zustände von Interaktion-Prozessen? Wie können die Agenten wissen, ob es eine Verbesserung der Versuche ihrer Koordination gibt und wie können die Agenten die Zustände ihrer Koordination verbessern und zusammenarbeiten?
- Wie erkennen und schlichten die Agenten Konflikte?
- Wie wird das Multiagentensystem ausgeführt und eingeschränkt?
- Wie werden die lokale Berechnung und die Kommunikation balanciert?
- Wie wird schädliches System-Verhalten (z.B. Chaos) vermieden oder gemildert?
- Wie verhandeln die Agenten miteinander?
- Wie werden die Gruppen oder die Organisationen von Agenten formiert um Ziele von Agenten zu erreichen oder Aufgaben von Agenten zu erledigen?
- Wie werden das Multiagentensystem und die Interaktionen zwischen Agenten formell beschrieben?
- Wie werden die intelligenten Prozesse wie Planung, Lösung eines Problems oder Treffen einer Entscheidung im MAS erkannt? Wie interagieren die Agenten um solche Prozesse durchzuführen?

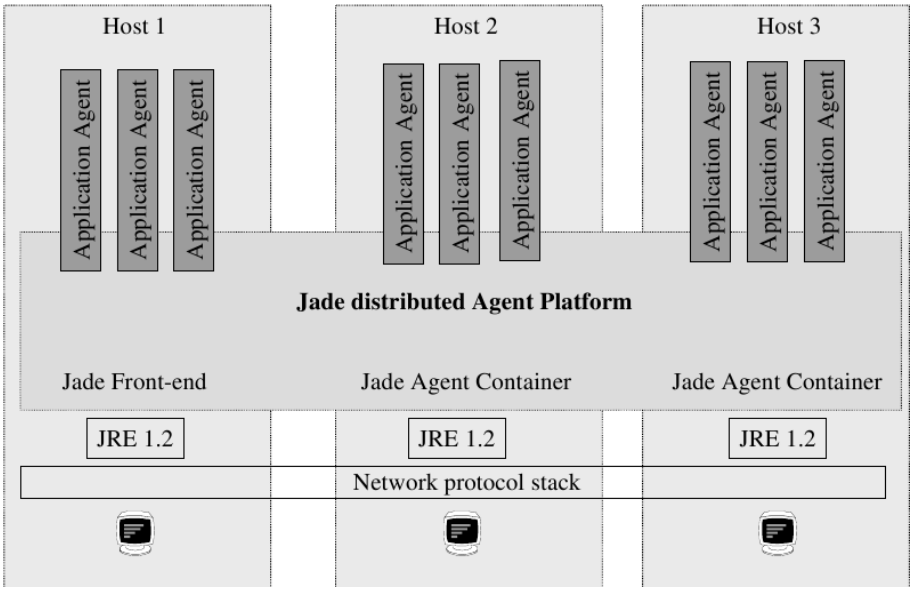
Viele Werkzeuge wurden entwickelt um die Konstruktion eines Multiagentensystems zu erleichtern. Im nächsten Abschnitt stelle ich einige wichtige Werkzeuge für den Aufbau eines Multiagentensystems vor.

Jade

JADE (Java Agent Development Environment) ist ein Framework von Tilab [4] [4]. Das Ziel des Framework besteht darin die Implementierung von standardisierten Agenten-basierten Applikationen zu erleichtern. JADE wurde mit der „Java“ Programmiersprache implementiert und befolgt die Spezifikationen der Organisation „The Foundation for Intelligent Physical Agents (FIPA)“. Das JADE-Framework kann aus zwei Sichten betrachtet werden.

Aus Sicht eines Laufzeitsystems bietet JADE verschiedene Dienste für die Agenten-basierten Applikationen. Ein von JADE konstruiertes Multiagentensystem ist ein verteiltes System und kann sich über verschiedene Hosts aufteilen. Auf jedem Host existiert ein Agenten-Container, der Teil des Systems ist. Ein Agenten-Container ist für die Verwaltung von Agenten zuständig. Beim Erzeugen eines neuen Agenten wird der Agent in einem Agenten-Container speichert. Wenn ein Agent von dem System entfernt werden soll, wird er von seinem Agenten-Container gelöscht. Jeder Container existiert in einer eigenen Java-virtuellen Maschine. Die Abbildung 2.4 zeigt die Architektur eines von JADE konstruiertes Multiagentensystems. JADE bietet die Mobilität-Funktion für

[4] http://jade.tilab.com/



382

Abbildung 2.4: die Architektur eines von JADE konstruierten Multiagentensysems [4]

die Agenten. Das heißt, die Agenten eines Containers können zu anderen Containern „gehen“. Die Kommunikation zwischen Agenten erfolgt durch den Austausch von ACL-standardisierten Nachrichten. Zur Überwachung des Multiagentensysems stehen verschiedene Werkzeuge wie „Remoting Manager Agent“ und „Sniffer Agent“ zur Verfügung.

Aus Sicht der Agenten-Entwicklung bietet JADE umfangreiche Bibliotheken zur Erleichterung der Implementierung und Fehlersuche von FIPA-konformen Agenten. Der Aufbau eines neuen Agenten erfolgt durch Erweiterung einer abstrakten Java-Agent-Klasse. Jeder Agent ist ein eigener Prozess. Dies erlaubt verschiedene Agenten parallel laufen zu lassen. Die von JADE vordefinierten Verhaltensweisen vereinfachen die Konstruktion des Agentenverhaltens. JADE unterstützt asynchrone Kommunikation, hat keine interne Agentenstrukturen und bietet keine Konzepte für gruppenorientierte Programmierung.

Jack

JACK [26] ist eine kommerzielle Plattform von AOS [5] zur Entwicklung autonomer Systeme. Die Plattform ist eine Implementierung des BDI-Paradigmas. Die JACK-Agenten werden durch die Komponenten `Ansichten`, `Wünsche` und `Intentionen` definiert [30]. Ereignisse sind zentrale Motivation-Faktoren zur Auslösung von Agentenaktionen. Ein Ereignis kann durch interne Berechnungen des Agenten oder durch die externe Umgebung erzeugt werden. Wenn ein Agent ein Ereignis bekommt, verwendet er einen Plan um auf das Ereignis zu reagieren. JACK bietet die JAL (JACK Agent Languague)-Sprache

[5] http://www.agent-software.com.au/products/jack/

zur Definition von Ereignissen, Plänen und Ansichten von Agenten. Es werden keine Standardspezifikationen wie FIPA oder KQML von JACK implementiert. Die Entwickler definieren selbst die Protokolle für die Kommunikation zwischen Agenten. Mit Hilfe von graphischen Designwerkzeugen wird die Entwicklung von Agenten erleichtert.

JACK bietet eine „JACK-Team"-Erweiterung für Team-orientierte Programmierung. JACK-Team [31] erlaubt vielen Agenten eine soziale Organisation zu formieren, um bestimmte Aufgaben zu erledigen. So wie ein JACK-Agent hat auch ein Team eigene Team-Komponenten: `Ansichten`, `Wünsche` und `Intentionen`. Das JACK-Team definiert die Konzepte für die weiteren Komponenten: `Team`, `Rolle`, `Team-Daten` und `Team-Plan`. Ein Team hat eine Aufgabe, die von verschiedenen Agenten erledigt wird. In einem Team spielt ein Agent eine bestimmte Rolle. Die Verbreitung von `Team-Ansichten` zwischen Agenten erfolgt durch den Austausch von Team-Daten. Ein `Team-Plan` beschreibt wie eine Aufgabe von verschiedenen Rollen erledigt wird.

MADKIT

MADKIT (Multi Agent Development Kit) ist eine Multiagentenplattform, die auf einem sogenannten „AaLaaDIN"-organisatorischen Modell basiert [22]. Wie das JACK-Team so hat auch das AaLaaDIN-Modell drei primitive Konzepte: `Agent`, `Gruppe` und `Rolle`. Eine Gruppe ist eine Menge von Agenten. MADKIT definiert keine interne Struktur für Agenten. Ein Agent kann nur mit anderen Agenten einer Gruppe kommunizieren wenn er an der Gruppe teilnimmt. Bei der Teilnahme einer Gruppe wird dem Agenten eine Rolle zugewiesen. Die Rolle beschreibt die Funktionen oder die Dienste des Agenten in der Gruppe. Die MADKIT-Plattform besteht aus drei Komponenten: Micro-Kernel, Agent und graphische Komponente. Die Abbildung 2.5 zeigt die organisatorische Struktur der Komponenten.

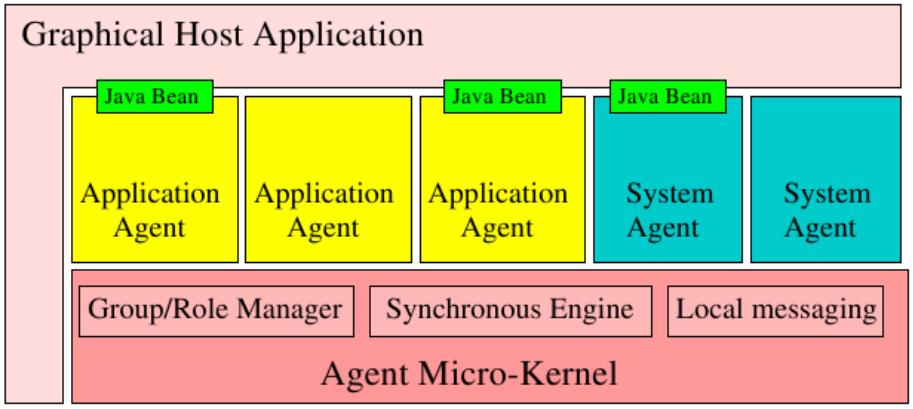


Abbildung 2.5: Architektur von MADKIT-Plattform [22]

Die Mircro-Kernel-Komponente spielt die Hauptrolle bei der Verwaltung der Agentengruppen und Rollen, dem Austausch von Nachrichten zwischen lokalen Agenten und der Verwaltung des Agenten-Lebenszyklus. Die Agent-Komponente von MAD-

KIT definiert den Agent-Lebenszyklus. Jeder Agent wird durch die `AgentAdress`-Klasse identifiziert. Diese Klasse beschreibt die Adresse für den Nachrichtenaustausch zwischen Agenten. MADKIT implementiert keine Standardspezifikation für die Agentenkommunikation sondern bietet eine abstrakte `Message`-Klasse, die der Konstruktion von zwischen Agenten ausgetauschten Nachrichten dient.

2.5 Fahrzeugfolgemodelle

Eine der wichtigsten Verhaltensweisen von Verkehrsteilnehmern ist die Folge-Verhaltensweise. Auf einer Spur versucht ein Verkehrsteilnehmer immer seine Geschwindigkeit so zu regulieren, dass es zur keiner Kollision mit dem Vorgänger kommt. Ein mathematisches Modell zur korrekten Darstellung der Folge-Verhaltensweise spielt eine wichtige Rolle im Aufbau von Mikro-Verkehrssimulationssystemen. Um diese Verhaltensweise darzustellen wurden verschiedene mathematische Modelle unter dem Namen „Car Following Model(CFM)“ entwickelt. Die Abbildung 2.6 zeigt die Folge-Verhaltensweise des Fahrzeuges V_a. Die Hauptaufgabe eines CFM-Modells ist Beantwortung der Frage

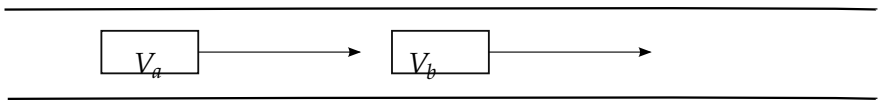


Abbildung 2.6: V_a folgt V_b

„Welche Geschwindigkeit fährt der Fahrer V_a in wirklichen Straßenverkehr sodass es keine Kollision mit dem Fahrer V_b entsteht ?“.

2.5.1 Das Modell von Chandler

Das erste einfache Modell wurde von Chandler et al. [8] im Jahr 1958 vorgestellt:

$$a_n(t) = c\Delta_v(t - T) \tag{2.1}$$

Wobei $a_n(t)$ die Beschleunigung des Fahrzeugs V_a zum Zeitpunkt t ist. $\Delta_v(t - T)$ ist die relative Geschwindigkeit zwischen V_a und V_b zum Zeitpunkt $t - T$. Der Sensitivitätsfaktor c bezeichnet die Intensität der Reaktion des Fahrzeuges V_a auf die Änderung der Geschwindigkeit von V_b. Wenn es keine Änderung der Geschwindigkeit von V_b gibt, dann fährt V_a die gleiche Geschwindigkeit wie V_b.

2.5.2 Das Modell von Gazis, Herman und Potts

Im Jahr 1959 verbesserten Gazis, Herman und Potts das Modell von Chandler et al [16]. Das neue Modell berücksichtigt den Abstand zwischen dem nachfolgenden Fahrzeug V_a und seinem Vorgänger V_b. Das Modell wird wie folgt geändert:

$$a_n(t) = c\frac{\Delta_v(t - T)}{\Delta_x(t - T)} \tag{2.2}$$

Wobei $\Delta_x(t-T)$ der Abstand zwischen V_a und V_b ist. Beim Ersetzen c durch $\frac{c}{\Delta_x(t-T)}$ hängt die Beschleunigung von V_a nicht nur vom Sensitivitätsfaktor sondern auch vom Abstand $\Delta_x(t-T)$ ab. Dabei wird mit steigendem Abstand $\Delta_x(t-T)$ der Einfluss der Änderung der Geschwindigkeit von V_b auf die Geschwindigkeit von V_a kleiner und umgekehrt.

2.5.3 Das Modell von Edie

Im Jahr 1961 verallgemeinerte Edie [12] das Modell von Gazis et al. beim Einfügen des Geschwindigkeitsfaktors s_a des Fahrzeuges V_a:

$$a_n(t) = c\frac{s_a^m}{\Delta_x^l(t-T)}\Delta_v(t-T) \tag{2.3}$$

Die beiden Konstanten m und l müssen durch empirische Untersuchungen definiert werden. Verschiedene Experimente wurden durchgeführt um eine beste Kombinationen zwischen m und l zu finden. Jedoch gibt es kein einheitliches Ergebnis. Die Abbildung 2.7 zeigt die Ergebnisse der durchgeführten Experimente

Table 1 Valuable combinations of the parameters m and l in the GHR model

Source	m	l	Note
Chandler et al. (1958)	0	0	
Herman & Potts (1959)	0	1	
Hoefs (1972)	1.5/0.2/0.6	0.9/0.9/3.2	Deceleration without the use of brakes/ Deceleration with the use of brakes/Acceleration
Treiterer&Myers (1974)	0.7/0.2	2.5/1.6	Deceleration/Acceleration
Ceder&May(1976)	0/0	3/0~1	Uncongested/Congested
Ozaki (1993)	0.9/−0.2	1/0.2	Deceleration/Acceleration

Abbildung 2.7: Empirische Untersuchung zur Bestimmung der Werte von m und l [52]

2.5.4 Das Modell von Gipps

Im Jahr 1981 stellte Gipps eine neue Annäherung des Fahrzeugfolgemodells vor. Es wurde angenommen, dass jedes Fahrzeug V die folgenden Eigenschaften hat:

1. A_V ist die maximale Beschleunigung von V.
2. D_V ist die maximale Verzögerung von V.
3. L_V ist die physikalische Länge von V.
4. M_V ist der Abstand zwischen der Halteposition von V und der Halteposition seines Vorgängers.
5. $V_V(t)$ ist die Geschwindigkeit von V zum dem Zeitpunkt t.

6. V_V ist die gewünschte Geschwindigkeit von V.
7. T ist die Zeit, die ein Fahrer von V braucht um auf eine Änderung der Geschwindigkeit seines Vorgängers zu reagieren. Gipps nahm an, dass alle Fahrzeuge die gleiche Zeit T haben.
8. $X_V(t)$ ist die Position von V zum Zeitpunkt t.

Im Gegensatz zu den vorstehenden Modellen berechnet das Modell von Gipps nicht die Beschleunigung $a_n(t)$ von V_a sondern die maximale Geschwindigkeit $V_{V_a}(t+T)$ zum Zeitpunkt $t+T$, die das Fahrzeug V_a unter allen Umständen erreichen kann. Die Geschwindigkeit $V_{V_a}(t+T)$ wird von zwei Bedingungen beschränkt. Die erste Bedingung ist die Kapazitätsbedingung. Es wird angenommen, dass das Fahrzeug V_a seine gewünschte Geschwindigkeit am besten durch maximale Beschleunigung A_{V_a} in der Zeit T erreichen will. Auf dieser Weise wird die maximal erreichbare Geschwindigkeit $V_{max,V_a}(t+T)$ von V_a durch die folgende Funktion berechnet:

$$V_{max,V_a}(t+T) = V_{V_a}(t) + 2.5A_{V_a}T(1 - \frac{V_{V_a}(t)}{V_{V_a}})\sqrt{0.025 + \frac{V_{V_a}(t)}{V_{V_a}}}$$

Die zwei Konstanten 2.5 und 0.025 wurden von Gipps vorgeschlagen. Die zweite Bedingung beschreibt die Beschränkung des Vorgängers V_b auf die Geschwindigkeit $V_{V_a}(t+T)$ des Fahrzeuges V_a. Das heißt, das Fahrzeug V_a muss seine Geschwindigkeit so regeln, dass im Fall einer Notbremsung von V_b immer noch mit Hilfe der maximalen Verzögerung D_{V_a} vor der Halteposition von V_b anhalten kann. Unter der Annahme, dass V_b zum Zeitpunkt t mit Stärke $D^*_{V_b}$ abbremst, berechnet V_a die Halteposition von V_b wie folgt:

$$X_{V_b} = X_{V_b}(t) - \frac{V^2_{V_b}(t)}{2D^*_{V_b}}$$

Nach T Zeiteinheiten reagiert das Fahrzeug V_a bei Reduzierung seiner Geschwindigkeit durch maximale Verzögerung D_{V_a}. V_a nimmt an, dass es zum Zeitpunkt $t+T$ die Geschwindigkeit $V_{pre,V_a}(t+T)$ fährt. Die Halteposition von V_a wird wie folgt berechnet:

$$X_{V_a} = X_{V_a}(t) + \frac{[V_{V_a}(t) + V_{pre,V_a}(t+T)]T}{2} - \frac{V^2_{V_a}(t+T)}{2D_{V_a}}$$

Damit nicht zur Kollision zwischen V_a und V_b kommt, muss die folgende Bedingung erfüllt werden:

$$X_{V_a} \leq X_{V_b} - L_{V_b} - M_{V_a} \tag{2.4}$$

Aus der Funktion 2.4 wird die Bedingung für $V_{pre,V_a}(t+T)$ abgeleitet:

$$V_{pre,V_a}(t+T) \leq D_{V_a}T + \sqrt{D^2_{V_a}T^2 - D_{V_a}[2[X_{V_b}(t) - X_{V_a}(t) - L_{V_b} - M_{V_a}] - V_{V_a}(t)T - V^2_{V_b}(t)0.5]} \tag{2.5}$$

Die sichere Geschwindigkeit für V_a zum Zeitpunk $t + T$ ist:

$$V_{V_a}(t + T) = min(V_{max,V_a}, V_{pre,V_a}) \tag{2.6}$$

2.6 Spurwechselmodelle

Der Spurwechsel ist die zweite Verhaltensweise von Verkehrsteilnehmern. In diesem Abschnitt stelle ich die wichtigsten Spurwechselmodelle vor. Im Allgemeinen kann eine Spurwechsel-Aktion in zwei Phasen eingeteilt werden: die Motivationsphase und die Durchführungsphase. Die Motivationsphase beschreibt die Gründe für einen Spurwechsel (z.B. gewünschte Geschwindigkeit). Die Gründe werden bewertet um für die Frage *„Soll der Fahrer jetzt seine Spur wechseln ?“* zu beantworten. Die zweite Phase wird nur weiter betrachtet, wenn die Spurwechselentscheidung positiv ausfällt. In der zweiten Phase wird zunächst die Durchführbarkeit eines Spurwechsels geprüft. Zur Prüfung der Durchführbarkeit wird ein „Gap Acceptance Model(GAM)“ Modell benutzt. Ein Fahrer wird letztlich seine Spur wechseln, wenn die Prüfung ein positives Ergebnis bringt.

2.6.1 Motivationen eines Spurwechsels

In seinem betrachtet Modell Gipps [19] drei zeitabhängige Gründe für einen Spurwechsel: gewünschte Geschwindigkeit erreichen, Sicherheit, persönlichen Pfad verfolgen. Ein Fahrer versucht zu jedem Zeitpunkt t durch seine Spurwechselentscheidung die drei obigen Gründe bestens zu befriedigen. Zum Beispiel, ein Fahrer überlegt zu einer anderen Spur zu wechseln wenn er auf dieser Spur schneller fahren kann und noch eine lange Fahrstrecke zwischen seiner aktuellen Position und der nächsten Ausfahrt liegt. Gipps betrachtet Fahrer in drei Zonen (siehe Abbildung 2.8) Die Motiva-

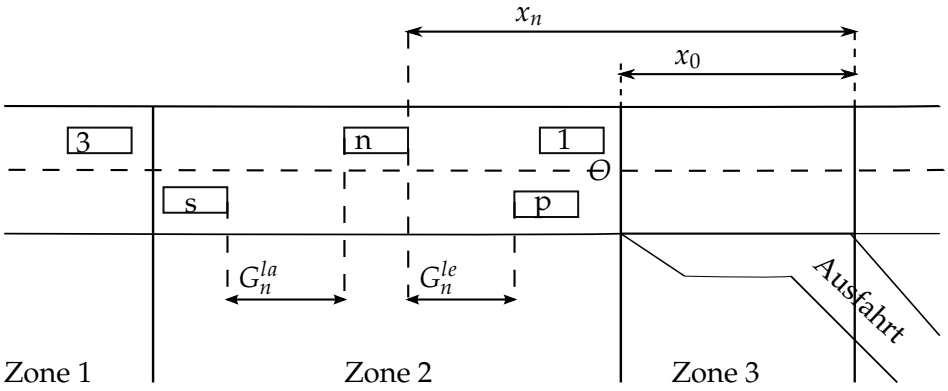


Abbildung 2.8: Die Fahrer, die ihre Ausfahrt erreichen wollen, müssen ihre Spurwechsel ab der kritische Position „O“ durchführen [41]

tionen eines Spurwechsels hängen von der Position eines Verkehrsteilnehmers in der

entsprechenden Zone ab. In jeder Zone werden die drei Spurwechselgründe nach ihrer Dringlichkeit bewertet. Die Abbildung2.8 zeigt ein Beispiel der Bewertung der Spurwechselgründe. Es wird angenommen, dass alle betrachteten Fahrer in den Zonen 1,2,3 letztlich zur Ausfahrt(*next turning*) fahren wollen. Die Motivation eines Spurwechsels wird dann wie folgt definiert:

Zone 1: Die Spurwechselentscheidung eines Fahrers in dieser Zone wird durch den Grad der Zufriedenheit über seine aktuelle Geschwindigkeit motiviert. Wenn ein Fahrer langsamer als seine gewünschte Geschwindigkeit fährt wird er versuchen, zu einer schnelleren Spur zu wechseln. Fährt er schnell genug, wird er wieder zu einer langsameren Spur wechseln.

Zone 2: Ein Fahrer in dieser Zone wechselt tendenziell zu der Spur, die ihm erlaubt seine Ausfahrt schneller zu erreichen. Jedoch berücksichtigt er immer noch den Grund „gewünschte Geschwindigkeit erreichen“. Dabei wird ein Spurwechsel durchgeführt, wenn er dadurch seine Ausfahrt schneller erreicht, seine Geschwindigkeit nicht zu stark reduzieren muss und keinen anderen Fahrer blockiert.

Zone 3: Die Zone 3 hat den kürzesten Abstand zur Ausfahrt. In dieser Zone zwingt ein Fahrer sich zum Wechsel auf die Spur, die ihm erlaubt seine Ausfahrt zu erreichen. Der Spurwechselgrund „Sicherheit“ spielt eine untergeordnete Rolle und der Spurwechselgrund „gewünschte Geschwindigkeit erreichen“ wird nicht mehr betrachtet. Wenn es nötig ist, wird der Fahrer seine Geschwindigkeit verzögern und anhalten um seine Spur zu wechseln.

Basierend auf unterschiedlichen Gründen eines Spurwechsels unterscheidet CORSIM [21] zwei Spurwechsel-Typen: „Mandatory Lane Changing(MLC)“ (obligatorischer Spurwechsel),„Discretionary Lane Changing(DLC)“ (willkürlicher Spurwechsel). Ein MLC wird durchgeführt wenn ein Fahrer zu einem Spurwechsel gezwungen ist (z.B Bauarbeiten auf einer Spur ausweichen). DLC beschreibt die Spurwechselentscheidung als persönlichen Willen eines Fahrers (z.B. um schneller zu fahren). Es gibt keine klare Festlegung dafür, wann ein Spurwechsel eines Fahrers vom Typ MLC ist. Allgemein wird der Abstand zwischen Fahrer und dem Objekt, das einen Spurwechsel vom Typ MLC verursacht, als ein Parameter einer stochastischen Funktion benutzt. Die Funktion wird so konstruiert, dass je kürzer der Abstand zwischen Fahrer und dem Objekt ist, desto größer ist die Wahrscheinlichkeit für den Spurwechsel vom Typ MLC.

Yang und Koutsopouslos(1996) [53] definierten ein regelbasiertes(rule-based) Spurwechselmodell für ihren MITSIM Verkehrssimulator. Sie klassifizierten drei Situationen, in denen ein Fahrer einen MLC durchführen muss:

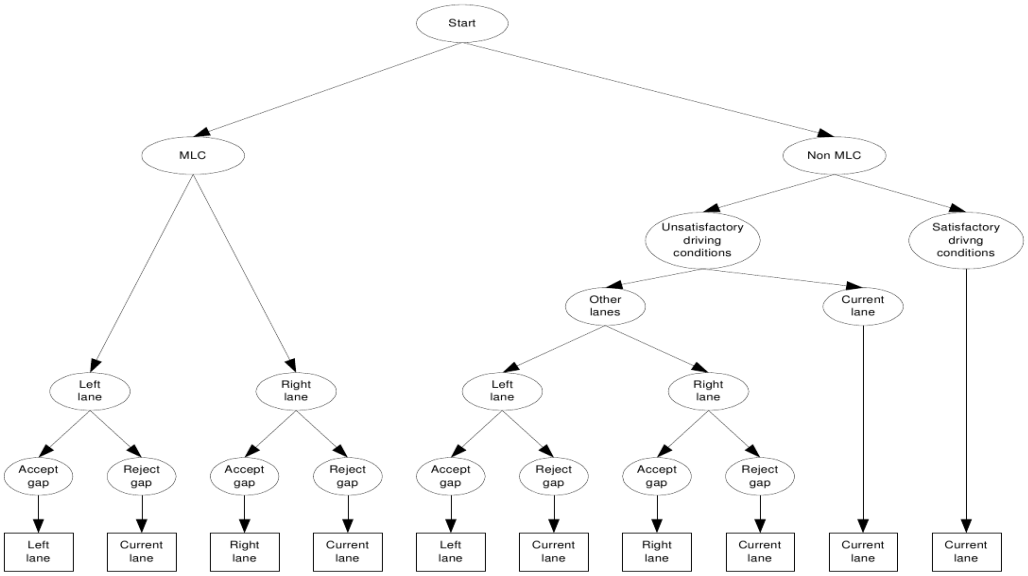
1. Der Fahrer will die Ausfahrt einer anderen Straße in seiner Route erreichen
2. Der Fahrer will blockierte Spuren vermeiden
3. Der Fahrer muss die Nutzungsvorschrift einer bestimmten Spur einhalten, oder auf Verkehrsschilder einer Spur (z.B. Hinweisschild Bauarbeiten) reagieren.

Ein Spurwechsel vom Typ DLC wird dann betrachtet, wenn ein Fahrer langsamer als seine gewünschte Geschwindigkeit fährt. Yan und Koutsopouslos benutzten eine stochastische Funktion um die Auswahl eines MLC darzustellen. In ihrer Arbeit beschreiben sie ein Beispielszenario (siehe Abbildung 2.8), in dem ein Fahrer n seine Ausfahrt erreichen will. Die Wahrscheinlichkeit dafür, dass der Fahrer n einen MLC zum Zeitpunkt t durchführt, wird durch die Funktion 2.7 berechnet:

$$f_n(t) = \begin{cases} exp\left(\dfrac{-(x_n - x_0)^2}{\alpha_0(1 + \alpha_1 m_n + \alpha_2 K)}\right) & x_n > x_0 \\ 1 & x_n \leq x_0 \end{cases} \tag{2.7}$$

Wobei $f_n(t)$ die Wahrscheinlichkeit dafür ist, dass der Fahrer n einen MLC durchführen muss. α_i sind vordefinierte Parameter. Der Parameter K ist der Indikator der Verkehrsüberlastung. Der Parameter m_n ist die Anzahl der Spuren, die der Fahrer n wechseln muss um seine Ausfahrt zu erreichen. Der Parameter x_n ist der Abstand zwischen n und der letzten Position der Ausfahrt. Der Parameter x_0 ist der Abstand zwischen kritischer Position O und der letzten Postion der Ausfahrt. Die kritische Position O ist die Position, an der der Fahrer n am stärksten motiviert wird seine Spur zu wechseln. Die Funktion 2.7 zeigt, dass er gezwungen ist seine Spur zu wechseln, wenn n über die kritische Position O fährt.

Ahmed(1999) [2] stellt in seiner konzeptionellen Rahmenstruktur ein generelles Spurwechselmodell vor. Die Abbildung 2.9 zeigt die Struktur des Modells als ein Entscheidungsbaum Auf der höchsten Ebene beschreibt die MLC-Branche eine Situation in der



583

Abbildung 2.9: Spurwechselmodell von Ahmed

ein Fahrer auf einen MLC reagieren muss. Die Wahrscheinlichkeit eines Fahrers auf

MLC zu reagieren zu müssen, basiert auf einem Vektor von folgenden Spurwechselgründen:

1. Abstand zwischen Fahrer und Objekt, das den MLC verursacht
2. Anzahl der Spuren, die der Fahrer wechseln muss um seine gewünschte Spur zu erreichen
3. Typ des Fahrzeuges, das der Fahrer fährt. Zum Beispiel reagiert der Fahrer früher auf den MLC, wenn er einen schweren Lastwagen fährt.

Ahmed berechnete diese Wahrscheinlichkeit mit Hilfe der stochastischen Funktion 2.8 [6].

$$P(MLC|n) = \frac{1}{1 + exp(-X_n^{MLC}(t)\beta^{MLC} - \alpha^{MLC}v_n)} \tag{2.8}$$

Wobei $-X_n^{MLC}(t)$ ein Vektor von Spurwechselgründen ist und β^{MLC} ein Vektor von entsprechenden Parametern ist. Der Parameter v_n ist ein Wert, der die persönlichen Eigenschaften des Fahrers n beschreibt und wird zufällig generiert. Der Parameter α^{MLC} ist ein Parameter von v_n. Im Unterschied zum Modell von Yang und Koutsopouslos [53] erlaubt das Modell von Ahmed die Berücksichtigung mehrerer Spurwechselgründe in einer MLC Entscheidung . Die Nicht-MLC Branche stellt die Überlegung des Fahrers einen DLC durchzuführen graphisch dar. Wenn der Fahrer nicht mit seinem aktuellen Zustand zufrieden ist und er seinen Zustand auf einer anderen Spur verbessern kann, versucht er einen DLC durchzuführen.

Toledo [43] unterscheidet nicht, wann ein Fahrer ein MLC oder DLC durchführt. Er integriert die Motivationen eines MLC und eines DLC in einer einzigen Nutzenfunktion 2.9:

$$U_n^d(t) = X_n^d(t)\beta^d + \alpha^d v_n + \varepsilon_n^d(t) \tag{2.9}$$

Wobei $U_n^d(t)$ der Nutzwert der Spur d ist. $X_n^d(t)$ ist ein Vektor der Spurwechselgründe der Spur d und β^d ist entsprechende Parametervektor. $\varepsilon_n^d(t)$ ist zufälliger Term. v_n ist ein Parameter, welcher die nicht beobachtbaren Eigenschaften des Fahrers n (z.B Aggressivität, Fahrterfahrung) beschreibt. Für jede Spur (Links, Rechts, Aktuell) wird ein Nutzwert berechnet. Der Fahrer n wird motiviert zu der Spur zu wechseln, die ihm den maximalen Nutzwert bringt. Toledo betrachtet die folgenden Spurwechselgründe: Nachbarschaft, Pfad-Plan, Kenntnisse vom Straßennetz, Angewohnheiten des Fahrers und Fahrfähigkeit.

2.6.2 GAM-Modell

Ein „Gapp Acceptance Model (GAM)“ Modell beschreibt die Durchführbarkeit eines Spurwechsels. Wenn ein Fahrer n seine Spur wechseln will, vergleicht er eine von ihm abgeschätzte kritische Lücke $G^c r_n$ mit der verfügbaren Lücke G_n auf der Zielspur. Wenn

[6] Für den Fall, dass es kein Objekt gibt, welches einen MLC verursacht(z.B Warnung vor Bauarbeiten), wird die Wahrscheinlichkeit gleich 0 gesetzt

die kritische Lücke größer als die verfügbare Lücke ist, wechselt er seine Spur. Das folgende binäre Wahlmodell stellt die Entscheidung des Fahrer n dar:

$$X_n = \begin{cases} 1 & if G_n \geq G^c r_n \\ 0 & if G_n < G^c r_n \end{cases} \tag{2.10}$$

Wobei n seine Spur wechseln wird, wenn das Ergebnis $X_n = 1$ ist. Die kritische Lücke $G^c r_n$ ist nicht sichtbar und stellt die minimale akzeptable Lücke, die von n abgeschätzt wird, dar. Wenn die kritische Lücke gleich oder kleiner als die vorhandene Lücke auf der Zielspur ist, wird n seine Spur wechseln ($X_n = 1$). Umgekehrt, wenn die kritische Lücke größer als die vorhandene Lücke auf der Zielspur ist, bleibt n auf seiner aktuellen Spur ($X_n = 0$). In vielen Verkehrssimulatoren wird die kritische Lücke durch eine stochastische Funktion generiert.

Yang und Koutsopouslos [53] unterscheidet Frontlücke (lead Gap) G_n^{le} und Hecklücke (lag Gap) G_n^{la} (siehe Abbildung 2.8). Die Frontlücke ist der freie Abstand zwischen dem Fahrzeug n und seinem Vorgänger auf der Zielspur. Die Hecklücke ist der freie Abstand zwischen dem Fahrzeug n und seinem Nachfolger auf der Zielspur. Der Fahrer n wird seine Spur wechseln wenn die beiden Lücken akzeptabel sind. Dafür werden zwei kritische Lücken durch die Gleichungen 2.11 berechnet. Es hängt vom Typ der Spurwechselentscheidung (MLC oder DLC) ab, wie die entsprechenden kritischen Lücken generiert werden:

$$\begin{aligned} G_{Dn}^{cr,la} &= max(G_n^{la}, G_n^{la} + \beta_{Dn1}^{la} + \beta_{Dn2}^{la}(v_n - v_s) + \epsilon_{Dns}) \\ G_{Dn}^{cr,le} &= max(G_n^{le}, G_n^{le} + \beta_{Dn1}^{le} + \beta_{Dn2}^{le}(v_n - v_s) + \epsilon_{Dnp}) \\ G_{Mn}^{cr,la} &= max(G_n^{la}, G_n^{la} + [\beta_{Mn1}^{la} v_n + \beta_{Mn2}^{la}(v_n - v_s)][1 - exp(-\gamma x_n^2)] + \epsilon_{ns}) \\ G_{Mn}^{cr,le} &= max(G_n^{le}, G_n^{le} + [\beta_{Mn1}^{le} + \beta_{Mn2}^{le}(v_n - v_s)][1 - exp(-\gamma x_n^2)] + \epsilon_{np}) \end{aligned} \tag{2.11}$$

Die Werte $G_{Dn}^{cr,la}, G_{Dn}^{cr,le}$ sind die Längen der kritischen Hecklücke und kritischen Frontlücke für den DLC. Die $G_{Mn}^{cr,la}, G_{Mn}^{cr,le}$ sind die Längen der kritischen Hecklücke und kritischen Frontlücke für den MLC. Die Parameter β und γ sind vordefinierte Modellparameter. Die Parameter ϵ_{np} und ϵ_{ns} sind maximale Berechnungsfehler. Der Parameter v_n ist die aktuelle Geschwindigkeit des Fahrers n. Bei einem MLC akzeptiert der Fahrer n auch die kleinere Lücke, wenn der Abstand zwischen seiner Position und der Ausfahrt kürzer wird.

Ahmed [2] benutzt das selbe Konzept der exponentiellen Funktion 2.8, um eine Funktion zur Generierung der kritischen Frontlücke und der kritischen Hecklücke zu konstruieren:

$$G_n^{cr,g} = exp(X_n^g \beta^g + \alpha^g v_n + \varepsilon_n^g) \qquad g = lead, lag \tag{2.12}$$

Wobei $G_n^{cr,g}$ die kritische Frontlücke oder Hecklücke für $g = lead$ bzw. $g = lag$ ist. X_n^g ist ein Vektor von Spurwechselgründen (z.B. Fahrzeugtyp, Anzahl der zu wechselnden Spuren um die gewünschte Spur zu erreichen), der die Spurwechselentscheidung beeinflusst. β^g ist der Vektor der Parameter der Spurwechselgründe. α^g ist der Parameter

von v_n. Der Parameter ε_n^g ist ein zufälliger Wert, der die persönlichen Eigenschaften des Fahrers n repräsentiert.

Gipps [19, 20] präsentiert sein eigenes GAM-Modell durch einen Algorithmus. Die Grundidee seines GAM-Modells kann wie folgt beschrieben werden. Der Fahrer n wechselt zur Spur i, wenn die folgende Bedingungen gelten:

- Auf der Spur i gibt es eine Lücke für den Spurwechsel.
- Auf der Spur i muss der Fahrer n sicherstellen, dass sein künftiger Nachfolger s ihm *folgen* kann
- Auf der Spur i muss der Fahrer n sicherstellen, dass er dem künftigen Vorgänger p *folgen* kann

Das *Folgeverhalten* wird durch ein CFM-Modell beschrieben (siehe das CFM-Modell des Abschnittes 2.5.4). Der Algorithmus 1 beschreibt das Spurwechselmodell eines Fahrers n (vergleiche mit Abbildung 2.8).

Algorithm 1 GAM von Gipps [19, 20]

```
Der Fahrer n möchte zur Spur i wechseln
s, p sind das nachfolgende Fahrzeug (upstream vehicle) bzw. das vorherfahrende
Farzeug (downstream vehicle) auf der Spur i
TargetGap(Leerstelle): ist der Abstand zwischen s und p
if TargetGap > Länge von n then
  berechne die Geschwindigkeit G_m von n via CFM-Modell mit p als Vorgänger
  if n kann seine Geschwindigkeit zu G_m reduzieren then
    berechne die Geschwindigkeit G_s von s via CFM-Modell mit n als Vorgänger
    if s kann seine Geschwindigkeit auf G_s reduzieren then
      n kann zur Spur i wechseln
    end if
  end if
end if
```

2.7 Verkehrssimulationssysteme

Die Forschung auf dem ITS-Gebiet beschäftigt sich mit der Effizienz verschiedener Verkehrsszenarien. Dafür werden die Verkehrsdaten verschiedener Verkehrsszenarien zu Vergleichszwecken benötigt. Wegen des hohen Aufwandes für die Datenerhebung und um Verkehrsteilnehmer nicht zu gefährden können viele Verkehrsszenarien nicht mit echten Verkehrsobjekten (z.B. Fahrzeuge, Straßen) durchgeführt werden. Ein Verkehrssimulationssystem bietet die Möglichkeit, verschiedene Verkehrsszenarios mit dem Computer zu entwerfen und zu simulieren. In einem Verkehrssimulationssystem werden die Verhaltensweisen von Verkehrsobjekten (z.B. Straßenampel, Verkehrsteilnehmer) durch verschiedene mathematische Modelle (z.B. CFM-Modelle) abgebildet. Heutzutage spielen Verkehrssimulationssysteme nicht nur im Verkehrsforschungsbereich

eine wichtige Rolle, sondern auch im Verkehrsmanagement. In dieser Arbeit benutze ich ein Verkehrssimulationssystem um das Fahrverhalten von Verkehrsteilnehmern zu simulieren. Die Daten des normalen Fahrverhaltens von Verkehrsteilnehmern werden mit den Daten meiner gruppenorientierten Fahrmethode (siehe Kapitel 1) verglichen.

Im Zentrum eines Verkehrssimulationssystems stehen das Fahrzeugfolgemodell und das Spurwechselmodell. Es existieren verschiedene Verkehrssimulationssysteme. Jedes System implementiert seine eigenen Modelle. In diesem Abschnitt gebe ich einen Überblick über drei führende kommerzielle Verkehrssimulationssysteme: AIMSUN , VISSIM und PARAMICS .

2.7.1 AIMSUN

AIMSUN ist ein mikroskopisches, mesoskopisches Verkehrssimulationssystem von TSS [7]. Das mikroskopische Simulationsniveau dient dazu kleine Verkehrsszenarien zu simulieren. AIMSUN benutzt das Fahrzeugfolgemodell und das Spurwechselmodell von Gipps [18, 19] für die Simulation des Fahrverhaltens von Verkehrsteilnehmern. Das mesoskopische Simulationsniveau ermöglicht AIMSUN große Verkehrsszenarien zu simulieren. Im mesoskopischen Simulationsniveau werden das Fahrzeugfolgemodell und das Spurwechselmodell dahingehend optimiert möglichst wenig Rechenleistung zu verbrauchen. Ein Verkehrsszenario kann automatisch von einer GIS-Datei [8] generiert werden. AIMSUN bietet dabei auch ein graphisches Werkzeug zur Modellierung verschiedener Verkehrsszenarien. Ein simuliertes Verkehrsszenario kann von AIMSUN durch

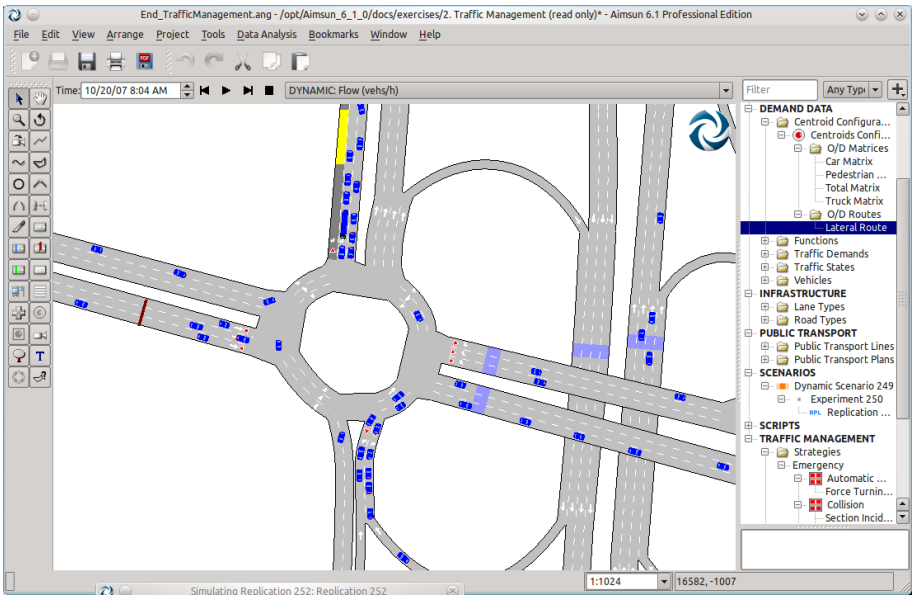


Abbildung 2.10: AIMSUN simuliert ein Verkehrsszenario

[7] http://www.aimsun.com/site/content/category/1/32/53/

[8] GIS ist die Abkürzung des englischen Begriffs „Geographic Information System“. Weitere Informationen über GIS können unter http://www.gis.com/ gefunden werden

zweidimensionale oder dreidimensionale Animation dargestellt werden. Nach einer Simulation können die Verkehrsdaten in der eigenen Datenbank oder in der von AIMSUN mitgelieferten Datenbank gespeichert werden.

Externe Applikationen können auf die Verkehrsobjekte mit Hilfe der von AIMSUN bereitgestellten Programmierschnittstellen zugreifen. Der Zugriff erfolgt mit Hilfe der Python- oder C-Programmiersprache. Weil AIMSUN nicht nur auf Windows- Betriebssystemen sondern auch auf Linux- und MAC-OS-Betriebssystemen installiert werden kann, ist es in der Lage mit den für Linux, MAC OS entwickelten Applikationen zu kommunizieren.

2.7.2 VISSIM

VISSIM ist ein mikroskopisches Verkehrssimulationssystem [38]. Das System wurde im Jahr 1970 von der Universität Karlruhe entwickelt. Im Jahr 1993 wurde es von PTV [9] als kommerzielles Software verbreitet. VISSIM benutzt das Fahrzeugfolgemodell von Wiedemann [48] zur Simulation des Verhaltens von Verkehrsteilnehmern. Für Benutzer bietet VISSIM eine graphische Schnittstelle. Diese Schnittstelle erlaubt den schnellen Entwurf verschiedener Verkehrsszenarios und eine einfache Kontrolle der Simulation. Während der Simulation können die Verhaltensweisen der simulierten Verkehrsobjekte als zweidimensionale oder dreidimensionale Animation dargestellt werden. Die Abbildung 2.11 zeigt eine laufende Simulation von VISSIM . So wie AIMSUN sammelt auch

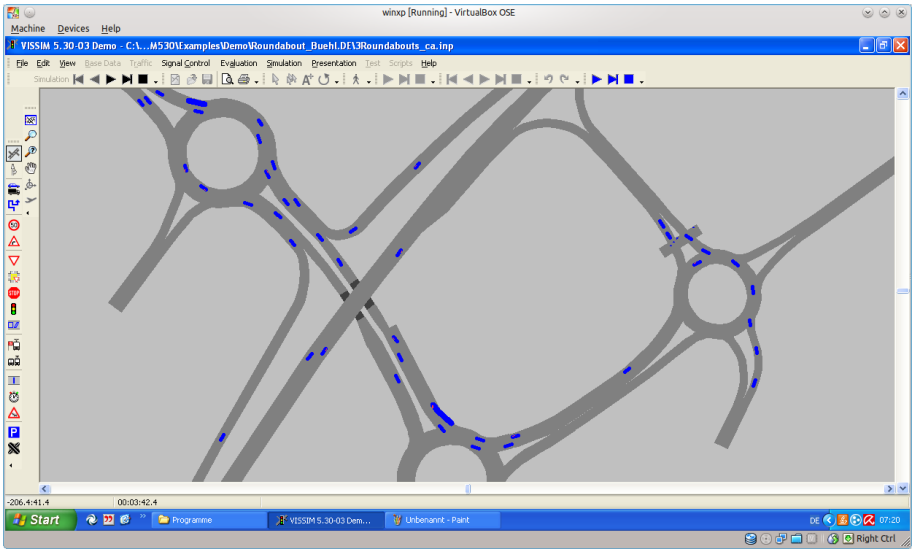


Abbildung 2.11: VISSIM simuliert ein Verkehrsszenario

VISSIM die Daten der simulierten Verkehrsobjekte. Diese Daten können in einer Datenbank oder als Textdateien abgespeichert werden.

[9] http://www.vissim.de/index.php?id=1801

Für Entwickler bietet VISSIM verschiedene Programmierschnittstellen. Ein externes Programm kann über die Programmierumgebungen (z.B. VisualBasic, Visual C++, Visual J++ oder Python) auf VISSIM -Objekte zugreifen. Weil VISSIM auf keinem Linux-Betriebssystem installiert werden kann, ist es nicht möglich für eine auf Linux laufende Applikation mit den Verkehrsobjekten von VISSIM zu interagieren.

2.7.3 PARAMICS

PARAMICS ist ein mikroskopisches Verkehrssimulationssystem von QuadstoneParamics [10]. Das Fahrzeugfolgemodell von PARAMICS basiert grundsätzlich auf dem Folgemodell von Fritzsche [15]. PARAMICS liefert verschiedene Werkzeuge für normale Benutzer und Entwickler zur Entwurf und Simulation von Verkehrsszenarien. Wie AIMSUN und VISSIM kann PRAMICS ein simuliertes Szenario als zweidimensionale oder dreidimensionale Animation darstellen.

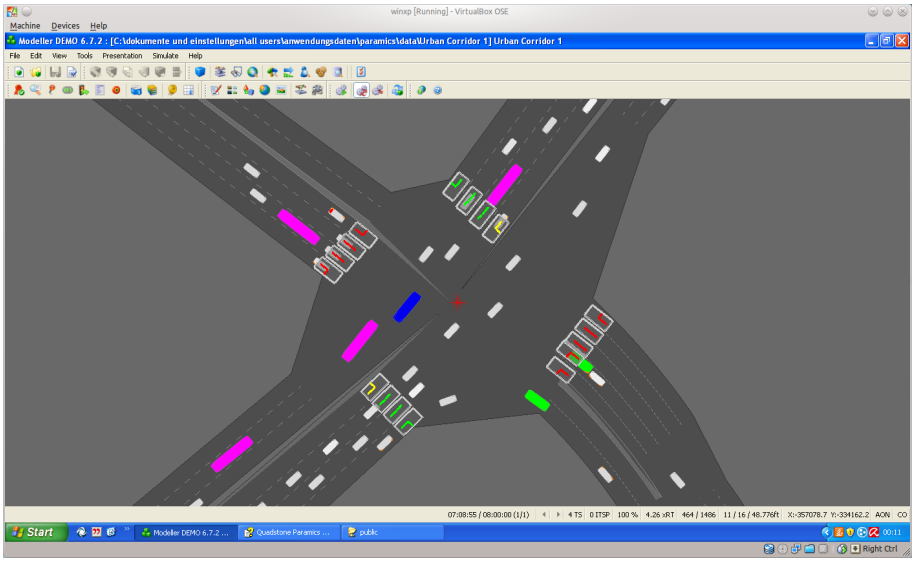


Abbildung 2.12: PARAMICS simuliert ein Verkehrsszenario

Eine Besonderheit an PRAMICS ist die sogenannte „Netzwerk-Simulation“-Funktion. Jeder Rechner wird als ein Prozessor-Knoten betrachtet und ist für eine Simulation zuständig. Die Rechner sind miteinander verbunden. Ein Rechner fungiert als Prozessor-Manager. Alle Simulationen können gleichzeitig durchgeführt werden. Die Ergebnisdaten werden zentral vom Prozessor-Manager formatiert, zusammengefasst und miteinander verglichen. Diese Funktion ist nützlich, wenn ein besonders großes Szenario simuliert werden soll. Die Verarbeitung und Analyse der simulierten Daten erfolgt mit Hilfe des „Analyser“-Werkzeugs. Das Werkzeug bietet verschiedene Möglichkeiten um die Daten darzustellen und zu verarbeiten. Zum Beispiel können die Daten

[10] http://www.paramics-online.com/index.php

von Verkehrsobjekten direkt auf der graphischen Schnittstelle eines Verkehrsszenarios dargestellt werden oder in verschiedenen Formaten (z.B. Excel) gespeichert werden.

Für Entwickler bietet PARAMICS die Möglichkeit auf die Verkehrsobjekte über die Programmierschnittstelle zuzugreifen. Im Gegensatz zu AIMSUN und VISSIM , erfolgt der Zugriff nur durch die „Visual C++"-Programmiersprache. Eine externe Applikation wird als ein Plugin betrachtet und wird beim Laden eines Verkehrsszenarios initialisiert.

2.7.4 Zusammenfassung

Im Allgemeinen bieten AIMSUN , VISSIM und PARAMICS alle grundsätzlichen Funktionen (graphische Darstellung, Werkzeug zur Modellierung, automatische Datensammlung und Bewertung) eines Verkehrssimulationssystems. Es gibt jedoch bei jedem System Vorteile und Nachteile. In diesem Abschnitt werden die Stärken und Schwächen der drei obigen Systeme zusammengefasst. Dabei betrachte ich die Stärken und Schwächen nur unter den folgenden Aspekten: Benutzerfreundlichkeit, Erweiterbarkeit, Funkionen für Datensammlung und Datenanalyse, zusätzliche Funktionen.

Benutzerfreundlichkeit: Im Vergleich mit VISSIM und PARAMICS bietet AIMSUN eine benutzerfreundlichere Grafikschnittstelle. Seine Integration von makroskopischen, mesoskopischen und mikroskopischen Modellierung-Werkzeugen in eine einzige Applikation erlaubt AIMSUN Benutzern einfacher ein großes Verkehrsnetzwerk zu modifizieren und zu simulieren. Im Gegensatz dazu erfordert die Konstruktion eines großen Verkehrsnetzwerks des VISSIM eine externe VISUM-Applikation. Die Demo-Version des PARAMICS bietet nur eine beschränkte Möglichkeit zur Modellierung eines einfachen vordefinierten Straßennetzes. Aus diesem Grund kann ich keine Aussage zur Modellierungs-Möglichkeit dieser Software machen. In einer anderen Studie von Hidas [25] wird gezeigt, dass die Eingabeprozedur des AIMSUN einfacher und schneller als die des PARAMICS ist. Die Konfiguration eines Modells mit AIMSUN führt bei einigen Benutzern zur Zeiteinsparung von 30 bis 50 % im Vergleich mit der Konfiguration mit PARAMICS .

Erweiterbarkeit: Die Erweiterbarkeit ist für Entwickler wichtig um die zusätzlichen Funktionen eines Systems zu entwickeln. Obwohl alle drei obige Systeme die API für den Zugriff auf ihre Komponenten anbieten, hat jedes System jedoch seine Beschränkung der Zugriffsmöglichkeit. Die Beschränkungen des AIMSUN und VISSIM liegen darin, dass sie keine API für den tiefen Zugriff auf graphische Darstellung der simulierten Verkehrskomponente bieten. Zum Beispiel können AIMSUN und VISSIM keinen vom Benutzer definierten Text direkt auf einem Verkehrsobjekt darstellen. AIMSUN bietet keine API um die Farbe eines Fahrzeuges dynamisch zu verändern. Im Gegensatz dazu bietet PARAMICS mehr Zugriff auf seine Komponenten und auf den Kern seines Simulators. Benutzer können beispielsweise zusätzliche Informationen eines Fahrzeuges definieren und sie während einer Simulation dynamisch darstellen. PARAMICS stellt auch ein SDK (Software Development Kit)-Werkzeug zur Entwicklung von zusätzlichen

Funktionen zur Verfügung. Die Nachteile des PARAMICS bestehen darin, dass es nur eine Programmierumgebung (Visual C++) unterstützt und nur auf dem Windows-Betriebssystem installiert werden kann.

Datensammlung und Datenanalyse: Bei der Datensammlung und Datenanalyse zeigt jedes System seine Stärken und Schwächen. Die Stärken des VISSIM liegen darin, dass es seinen Benutzern selbst ihre Datenformat zu konfigurieren erlaubt, ein Analyser-Werkzeug zur Verfügung stellt und verschiedene Möglichkeiten bietet um Daten in einer Datenbank abzuspeichern. Die Stärke des AIMSUN ist die Fähigkeit die Daten in verschiedenen Datenbanken abspeichern zu lassen. Die Schwäche des AIMSUN ist, dass es kein gutes Werkzeug zur Datenanalyse bietet. Das PARAMICS hat die folgenden Stärken: Funktion zur Runtime-Datenanalyse, dezentrale Datenverarbeitungsfunktion und Analayser-Werkzeug. Die Funktion der Runtime-Datenanalyse ermöglicht die Daten während des Laufes einer Simulation graphisch darzustellen. Mit der dezentralen Datenverarbeitungsfunktion können die Daten, die auf verschiedenen Computern liegen, verarbeitet werden. Diese Funktion ist für die dezentrale Simulation nützlich. Der einzige Nachtei von PARAMICS ist, dass es die Daten nur in .csv Dateien abspeichern kann. Die Speicherung in einer Datenbank erfordert einen zusätzlichen Plugin.

Simulationsfehler: Laut der Arbeit von Panwai und Dia [36] simuliert AIMSUN das Fahrzeugfolgeverhalten genauer als die PARAMICS und VISSIM . Eine andere Arbeit von Xiao et al. [51] zeigt, dass die Genauigkeiten der beiden VISSIM und AIMSUN Systeme gleich sind.

Zusätzliche Funktionen: Eine Stärke des PARAMICS gegenüber VISSIM und AIMSUN ist die „Netzwerk-Simulation“-Funktion (siehe Abschnitt 2.7.3). PARAMICS kann viele Simulationen auf verschiedenen Computern gleichzeitig starten. Diese Funktion wurde von einigen Forschern [29, 28] benutzt um ein verteiltes Verkehrssimulationssystem zu konstruieren.

Die obigen Beschreibungen haben einen Überblick über Stärken und Schwächen der drei Systeme gegeben. Um ein System auszuwählen sollten die Stärken und Schwächen im Bezug auf die Anforderungen eines Projektes geeignet bewertet werden. Ein Leitfaden für die Auswahl eines Verkehrssimulationssystems wird von Downling et al. in ihrer Arbeit [10] beschrieben. Diese Arbeit erfordert ein Verkehrssimulationssystem, welches so genau wie möglich das menschliche Fahrverhalten simuliert. Aus diesem Grund wähle ich das AIMSUN System.

3 Gruppenorientierte Fahrmethode

3.1 Einleitung

Wie ich im Abschnitt 1.1 dargestellt habe, hat jeder Verkehrsteilnehmer eine bestimmte gewünschte Geschwindigkeit zu erreichen. Diese Geschwindigkeit spielt die Hauptrolle bei der Entscheidung über die Fahraktionen des Verkehrsteilnehmers. Wenn die gewünschten Geschwindigkeiten von Verkehrsteilnehmern unterschiedlich sind, passiert es sehr oft, dass die schnell Fahrenden von den langsamer Fahrenden blockiert werden. Solch eine Situation wird als *Konfliktsituation* bezeichnet. In dieser Arbeit werden die autonomen Fahrzeuge verschiedene Gruppen bilden. Somit entstehen die Konfliktsituationen auf Gruppenebene. Das heißt, zwei Fahrzeuggruppen sind im Konflikt, wenn die schnelle Fahrzeuggruppe von der langsamen Fahrzeuggruppe blockiert wird. Die gruppenorientierte Fahrmethode beschäftigt sich mit dieser Konfliktsituation. Mit Hilfe der gruppenorientierten Fahrmethode können die autonomen Fahrzeuge das Auftreten von Konfliktsituationen reduzieren. Im Fall einer Konfliktsituation kooperieren die autonomen Fahrzeuge mit einander um die Konfliktsituation zu lösen. Dadurch erhöht sich die durchschnittliche Geschwindigkeit der Fahrzeuge.

Bevor ich die gruppenorientierte Fahrmethode in diesem Kapitel vorstelle, werden die folgenden zentralen Fragen informell beantwortet:

1. Was ist die gruppenorientierte Fahrmethode?
2. Wann bilden Fahrzeuge eine Gruppe ?
3. Bei welchen Verkehrszuständen ist die gruppenorientierte Fahrmethode verwendbar?

Die gruppenorientierte Fahrmethode lässt sich von Naturphänomenen inspirieren. Seit Jahren sind Naturphänomene wie Fischschwärme oder Vogelschwärme Forschungsthemen für Biologen. Es wird allgemein akzeptiert, dass Fische wie Sardinen im Schwarm schwimmen um sich vor anderen Raubfischen zu schützen. Dies bedeutet, dass das kollektive Verhalten ein natürliches Verhalten der Sardinen ist. Sie sammeln sich auch dann im Schwarm, wenn es keine Gefahr gibt. Jede Sardine kontrolliert ihre Bewegung unter Berücksichtigung von Aktionen anderer Sardinen. Wenn eine Sardine eine Gefahrenquelle entdeckt, bewegt sie sich schneller als normal in eine Richtung um sich weit weg von der Gefahr zu bringen. Diese Aktion löst Ängste bei den anderen Sardinen in unmittelbarer Nähe aus. Als Konsequenz schwimmen die anderen in die selbe Richtung der ersten. Mit dieser Erkenntnis können die Antworten für die zentralen Fragen informell wie folgt formuliert werden:

1. Die gruppenorientierte Fahrmethode wird wie folgt definiert:

   Die gruppenorientierte Fahrmethode ist eine kooperative Fahrmethode, die die Fahrzeuggruppen als Hilfselemente nutzt, um die autonomen Fahrzeuge global zu koordinieren und ihre Fahraktionsentscheidungen zu unterstützen. Dadurch erreichen die autonomen Fahrzeuge effizient und sicher ihre Ziele.

   Die gruppenorientierte Fahrmethode beginnt mit der Zuordnung der autonomen Fahrzeuge zu den geeigneten Gruppen. Eine nützliche Analogie aus der Biologie stellt die Betrachtung einer Gruppe von Fahrzeugen als ein Schwarm von Sardinen dar.

   So wie die Sardine eine Gefahr erkennt, hat jedes Mitglied einer Gruppe die Rolle eines Konfliktentdeckers. Wenn eine Konfliktsituation entdeckt wird, wird sie an den Gruppenführer gemeldet. Im Gegensatz zu den Sardinen, die ihre Aktionen in einer Gefahrensituation reaktiv durchführen, werden die Aktionen der Gruppenmitglieder in einer Konfliktsituation von dem Gruppenführer global koordiniert. Dabei erfolgt die Koordination in zwei Phasen. Die erste Phase ist die globale Koordinationsphase. In dieser Phase wählt der Gruppenführer die Fahrspuren für seine Gruppe aus (Gruppenfahrspuren). Die gewählten Spuren sollen sicherstellen, dass alle Gruppenmitglieder, die auf solchen Spuren fahren, andere Gruppen überholen können (oder von anderen Gruppen überholt werden). Die zweite Phase ist die lokale Entscheidung jedes Fahrzeugmitgliedes. Ein Gruppenmitglied kann entschieden, ob es auf der Gruppenspur fahren will oder nicht. Die Entscheidung hängt von dem aktuellen Zustand (siehe Abschnitt 3.5.3) des Mitgliedes ab.

2. Die Gruppenbildung der Sardine ist ein instinktives Verhalten. Bei den autonomen Fahrzeugen ist das Gruppenbildungsverhalten auch *instinktiv* motiviert. Dies bedeutet, ein Fahrzeug fragt sich nicht „*Warum muss ich an einer Gruppe teilnehmen ?*“ sondern es versucht zu jedem Zeitpunkt instinktiv die passende Gruppe zu finden. Damit ist die Gruppenbildung unabhängig von Konflikten zwischen Fahrzeuggruppen.

3. Die gruppenorientierte Fahrmethode erfordert dass sich verschiedene Fahrzeuge zu einzelnen Gruppen zusammenfassen lassen. Dabei ist die Gruppenbildung abhängig von der mentalen Motivation und den physikalischen Eigenschaften der individuellen. Zum Einen bilden beispielsweise Sardinen einen Schwarm, weil sie sicher vor Raubfischen sein wollen (instinktive mentale Motivation). Zum anderen haben Sardinen fast gleiche physikalische Eigenschaften (z.B. Größe, maximale Schwimmgeschwindigkeit, Farbe usw.). Damit die gruppenorientierte Fahrmethode verwendet werden kann, müssen Fahrzeuge unterschiedlicher Konfliktgruppen signifikant unterschiedliche Eigenschaften besitzen. Die Mitglieder einer Gruppe haben gleiche oder fast gleiche physikalische Eigenschaften. Zum Beispiel unterscheidet sich eine Gruppe von PKW-Fahrzeugen (Personenkraftwaren) von einer Gruppe von LKW-Fahrzeugen (Lastkraftwagen) bezüglich Beschleunigungs- und Verzögerungs-Kapazität.

Dieses Kapitel wird wie folgt konstruiert. Zunächst wird die Gruppenbildungsmethode im Abschnitt 3.2 beschrieben. Der Abschnitt 3.3 beschreibt kurz wie die Konfliktsituationen zwischen Fahrzeuggruppen entdeckt werden. Im Abschnitt 3.4 wird die Methode zur globalen Koordination von Fahrzeuggruppen beschrieben. Die Methode für die lokalen Fahraktionsentscheidungen eines Fahrzeuges wird im Abschnitt 3.5 präsentiert. Der Abschnitt 3.6 fasst die wichtigen Aspekte dieses Kapitels zusammen.

3.2 Gruppenbildung

Das Ziel der Gruppenbildung besteht darin, Objekte mit gleichen Zielen oder gleichen Eigenschaften (oder fast gleiche Eigenschaften) in eine Gruppe zu klassifizieren. Die Konstruktion einer Gruppenbildungsmethode für Fahrzeuge sollte die Stabilität der Gruppe beachten. Dabei beschreibt die Stabilität einer Fahrzeuggruppe die Änderungen der relativen Abstände zwischen Fahrzeugmitgliedern. Eine Fahrzeuggruppe ist nicht stabil, wenn in kurzer Zeit die relativen Abstände zwischen Mitgliedern signifikant geändert werden. Neben dem aktuellen Verkehrszustand haben die Eigenschaften eines Fahrzeuges einen großen Einfluss auf die Stabilität der Fahrzeuggruppe. Bildet ein schnelles Fahrzeug mit einem langsamen Fahrzeug eine Gruppe, dann steigt der Abstand zwischen den Fahrzeugen rasant an. Die Gruppe wird instabil und bricht zusammen. Aus diesem Grund spielen die statischen Eigenschaften eines Fahrzeuges eine wichtige Rolle bei der Bildung der Fahrzeuggruppe. Ich betrachte drei statische Eigenschaften: *gewünschte Geschwindigkeit*, *maximale Verzögerung*, *maximale Beschleunigung*. Diese Eigenschaften sind die Hauptelemente für die Konstruktion einer Fahrzeuggruppe.

Gewünschte Geschwindigkeit: Wie schon im ersten Kapitel erwähnt, erhöht ein Fahrzeug seine Geschwindigkeit bis es seine gewünschte Geschwindigkeit erreicht. Wenn sich die gewünschten Geschwindigkeiten zu stark unterscheiden, dann trennen sich die Mitglieder schon nach kurzer Zeit. Deshalb beeinflusst die gewünschte Geschwindigkeit stark die Stabilität der Gruppe.

Maximale Beschleunigung: Die maximale Beschleunigung ist die Kapazität eines Fahrzeuges seine Geschwindigkeit in einem bestimmten Zeitabstand maximal zu beschleunigen. Die Beschleunigung a wird durch die folgende mathematische Funktion berechnet:

$$a = \frac{v_2 - v_1}{t_2 - t_1} \tag{3.1}$$

wobei v_1 die Geschwindigkeit des Fahrzeuges zum Zeitpunk t_1 ist und v_2 die Geschwindigkeit des Fahrzeuges zum Zeitpunkt t_2 ist. Dabei soll die Beschleunigung immer einen positiven Wert annehmen. Das heißt, die Bedingung $v_2 > v_1$ musst immer erfüllt sein.

Maximale Verzögerung: Die maximale Verzögerung bezeichnet die Kapazität eines Fahrzeuges seine Geschwindigkeit in einem bestimmten Zeitabstand maximal zu

reduzieren. Die Verzögerung d wird durch die folgende mathematische Funktion berechnet:

$$d = \frac{v_2 - v_1}{t_2 - t_1} \tag{3.2}$$

wobei v_1 die Geschwindigkeit des Fahrzeuges an dem Zeitpunk t_1 ist und v_2 die Geschwindigkeit des Fahrzeuges an dem Zeitpunkt t_2 ist. Dabei soll die Beschleunigung immer einen negativen Wert annehmen. Das heißt, die Bedingung $v_2 < v_1$ musst immer erfüllt sein.

Der Grund für die Auswahl der maximalen Beschleunigung und maximalen Verzögerung als die Hauptelemente für die Konstruktion einer Fahrzeuggruppe liegt darin, dass die Fahrzeuge beim Fahren ständig ihre Geschwindigkeiten umstellen müssen. Wenn zwei Fahrzeuge große Unterschiede in Beschleunigungen und Verzögerungen aufweisen, dann wird der Abstand zwischen ihnen durch ständige Geschwindigkeitsumstellungen immer größer. Als Konsequenz bricht die Gruppe schneller zusammen. Eine Fahrzeuggruppe wird dann stabiler, wenn die Fahrzeugmitglieder nur kleine Unterschiede in den gewünschten Geschwindigkeiten, maximalen Beschleunigungen und maximalen Verzögerungen aufweisen.

3.2.1 Definition einer Fahrzeuggruppe

Eine Fahrzeuggruppe G^X wird wie folgt definiert:

Definition 3.2.1. Es seien $f : X \times Y \rightarrow R$ eine Funktion, die die Unähnlichkeit der zwei Fahrzeuge X und Y berechnet. G^X ist eine Fahrzeuggruppe und X ist der Gruppenführer dann und nur dann wenn:

$$\forall Y \in G \; f(X, Y) \leq \alpha$$

Die obige Definition beschreibt ein Fahrzeug X als den Gruppenführer einer Fahrzeuggruppe G^X. Ein beliebiges Fahrzeug wird zur Gruppe G^X zugeordnet, wenn die Unähnlichkeit zwischen deren Eigenschaften und den Eigenschaften des Gruppenführers X kleiner oder gleich α ist. Die Abbildung 3.1 zeigt eine graphische Darstellung einer Gruppe G^X. Wenn Fahrzeuge wegen bestimmter gemeinsamer Eigenschaften eine Gruppe bilden, dann weisen die Fahrzeuge in zwei Gruppen unterschiedliche Eigenschaften auf. Sei ein Vertreter einer Gruppe ein virtuelles Fahrzeug, dessen Eigenschaften den durchschnittlichen Eigenschaften aller Mitglieder der Gruppe entsprechen, dann sind die Eigenschaften der Gruppe auch die Eigenschaften des Vertreters der Gruppe.

Es ist wichtig, dass die bestimmten Eigenschaften von zwei Gruppen miteinander verglichen werden können. Zum Beispiel, wird der Vergleich zwischen der gewünschten Geschwindigkeit der Gruppe A und der gewünschten Geschwindigkeit der Gruppe B benutzt, um unterscheiden zu können, welche Gruppe schneller ist. Theoretisch kann der Vergleich der Eigenschaften zwischen zwei Gruppen durch den Vergleich der Eigenschaften zwischen den Vertretern der entsprechenden Gruppen erfolgen. Es gibt zwei Methoden zur Definition der Eigenschaften des Gruppenvertreters.

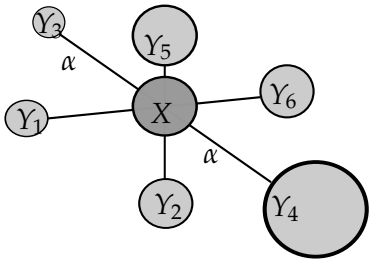


Abbildung 3.1: Darstellung einer Gruppe G^X mit sieben Mitgliedern und X als Gruppenführer. Jedes Mitglied wird einem Kreis zugeordnet. Die Größe eines Kreises stellt die Eigenschaften dessen Besitzers dar. Die Längen der Linien zwischen X und anderen Mitgliedern Y_i stellen die Unähnlichkeit-Werte der $f(X, Y_i)$ Funktion dar. Das Paar (Y_3, Y_4) hat die größte Unähnlichkeit 2α.

- Die erste Methode definiert einen virtuellen Vertreter. Die Eigenschaften des virtuellen Vertreters entsprechen den durchschnittlichen Eigenschaften aller Gruppenmitglieder.
- Die zweite Methode bestimmt den Gruppenführer als Gruppenvertreter.

In einer dynamischen Verkehrsumgebung verändert sich die Anzahl der Mitglieder ständig. Wenn die erste Methode eingesetzt würde, müssten die Eigenschaften des Vertreters ständig aktualisiert werden. Dies verursacht einen großen Aufwand und ist für eine dynamische Umgebung wie Straßenverkehr, in der die autonomen Fahrzeuge schnell ihre Entscheidungen treffen müssen, nicht geeignet. Deswegen benutze ich in dieser Arbeit die zweite Methode. Sei der Ungenauigkeit-Wert der zweiten Methode die Unähnlichkeit zwischen dem Gruppenführer und dem virtuellen Gruppenvertreter dann liegt der Ungenauigkeit-Wert der zweiten Methode im Bereich $[0, \alpha)$.

3.2.2 Bewertung der Unähnlichkeit von zwei Fahrzeugen

Die Bewertung der Unähnlichkeit von zwei Fahrzeugen spielt die zentrale Rolle in dem Gruppenbildungsalgorithmus, den ich im Abschnitt 3.2.3 beschreibe. Die Bewertung erfolgt durch Nutzung einer Bewertungsfunktion $f(X, Y)$. Die $f(X, Y)$ Funktion wird sowohl von Gruppenteilnehmern als auch von Gruppenführern benutzt. Ein Gruppenführer X benutzt die Bewertungsfunktion um einen Teilnehmer zu bewerten, ob dieser an seiner Gruppe teilnehmen darf. Ein Gruppenteilnehmer Y benutzt die $f(X, Y)$ Funktion um seine gewünschte Gruppe herauszufinden und an ihr teilzunehmen.

Wie im Abschnitt 3.2.1 definiert, darf ein Fahrzeug Y an einer Gruppe G^X nur teilnehmen, wenn die Unähnlichkeit $f(X, Y)$ kleiner oder gleich α ist. Jedoch wurde bisher keine Definition für die Bewertungsfunktion $f(X, Y)$ und den α-Wert vorgestellt. In diesem Abschnitt stelle ich die Konstruktion der Bewertungsfunktion $f(X, Y)$ und eine geeignete Wahl des α-Wertes vor.

Die Konstruktion einer Bewertungsfunktion $f(X, Y)$ musst die folgenden Eigenschaften enthalten:

Robustheit: Die Bewertungsfunktion soll robust gegen Änderungen der Umgebung sein. D.h., wenn ein Fahrzeug Y zu einem Zeitpunkt t ein potentielles Mitglied einer Gruppe G von X ist, dann ist es auch zum Zeitpunkt $t + n$ ein potentielles Mitglied der Gruppe G von X.

Flexibilität: Die Bewertungsfunktion soll flexibel für den Bedarf des Fahrers (oder des autonomen Fahrzeuges) sein. D.h., das Fahrzeug X muss zu jedem Zeitpunkt die Bewertungsfunktion $f(X, Y)$ beeinflussen können, um die gewünschten Mitgliedskandidaten zu finden.

Eine bekannte Methode zur Berechnung der Unähnlichkeit von zwei Objekten ist, die Objekte in einem mehrdimensionalen Raum darzustellen und den Abstand zwischen ihnen zu berechnen. Je größer ist der Abstand, desto unähnlicher sind die beiden Objekte. In dieser Arbeit benutze ich die gewichtete Manhattan-Abstandsfunktion zur Berechnung der Unähnlichkeit von Fahrzeugen. $V(ds, acc, dcc)$ sei ein Fahrzeug. Die Parameter ds, acc, dcc sind die gewünschte Geschwindigkeit, maximale Beschleunigung und maximale Verzögerung von V. Der Manhattan-Abstand $f_1(X, Y)$ zwischen zwei Fahrzeugen $X(ds_x, acc_x, dcc_x)$ und $Y(ds_y, acc_y, dcc_y)$ wird wie folgt definiert:

$$f_1(X, Y) = \frac{|ds_x - ds_y|}{s_{ds}} + \frac{|acc_x - acc_y|}{s_{acc}} + \frac{|dcc_x - dcc_y|}{s_{dcc}} \tag{3.3}$$

Wobei s_{ds}, s_{acc}, s_{dcc} die akzeptablen Abweichungen der Eigenschaften *gewünschte Geschwindigkeit, maximale Beschleunigung, maximale Verzögerung* sind. Der Grund für die Einführung der akzeptablen Abweichungen liegt darin, dass im Straßenverkehr zwei Fahrzeuge nur selten die gleichen Eigenschaften besitzen. Die akzeptablen Abweichungen erlauben einem Fahrzeug X mit einem anderen Fahrzeug Y, dessen Eingenschaften nur wenig von den Eigenschaften des Fahrzeuges X abweichen, eine Gruppe zu bilden. Laut der Gruppendefinition 3.2.1 akzeptiert das Fahrzeug X das Fahrzeug Y als sein Gruppenmitglied wenn die Unähnlichkeit $f_1(X, Y)$ kleiner oder gleich einem vordefinierten α-Wert ist. Es ist erkennbar, dass je kleiner das Fahrzeug X die akzeptablen Abweichungen s_{ds}, s_{acc}, s_{dcc} setzt, desto schwieriger wird es für X eine Gruppe zu bilden. Bei kleinen akzeptablen Abweichungen erhöht sich die Stabilität der gebildeten Gruppe von X. An diesem Punkt stellt sich die Frage:
Wie werden die Werte der akzeptablen Abweichungen s_{ds}, s_{acc} und s_{dcc} berechnet, so dass die autonomen Fahrzeuge möglichst stabile Gruppen bilden?
Im Rahmen dieser Arbeit werde ich diese Frage nicht näher betrachten und als zukünftiges Forschungsthema offen lassen. In meinem Test im Kapitel 6 werden die akzeptablen Abweichungen für einzelne Testszenarien vordefiniert.

Die Gruppendefinition 3.2.1 legt fest, dass die Unähnlichkeit zwischen einem zufälligen Mitglied und dem Gruppenführer kleiner oder gleich einem α-Wert ist. Neben der Festlegung der akzeptablen Abweichungen s_{ds}, s_{acc}, s_{dcc}, hat die Festlegung des α-Wertes eine entscheidende Rolle bei der Bildung von Gruppen. Im folgenden beschreibe ich eine Möglichkeit zur Festlegung des α-Wertes.

Sei G ist eine Menge von Fahrzeugen Y, deren Eigenschaften ds_y, acc_y, dcc_y in den entsprechenden Bereichen $[ds_x - s_{ds}, ds_x + s_{ds}]$, $[acc_x - s_{acc}, acc_y + s_{acc}]$, $[dcc_x + s_{dcc}, dcc_x -$

$s_{dcc}]$ liegen. Bei der Festlegung der akzeptablen Abweichungen s_{ds}, s_{acc}, s_{dcc}, bildet das Fahrzeug X *tendenziell* mit den Fahrzeugen der Menge G eine Gruppe. Es ist zu bemerken, dass die Fahrzeuge $Y_1(ds_x - s_{ds}, acc_x - s_{acc}, dcc_x + s_{dcc})$ und $Y_2(ds_x + s_{ds}, acc_x + s_{acc}, dcc_x - s_{dcc})$ das *stärkste* und das *schwächste* Fahrzeug von G sind. Die Manhattan-Abstände $f_1(X, Y_1)$ und $f_1(X, Y_2)$ können mit Hilfe der Funktion 3.3 wie folgt ermittelt werden:

$$\begin{aligned} f_1(X, Y_1) &= \frac{|ds_x - (ds_x - s_{ds})|}{s_{ds}} + \frac{|acc_x - (acc_x - s_{acc})|}{s_{acc}} + \frac{|dcc_x - (dcc_x + s_{dcc})|}{s_{dcc}} \\ &= 3 \\ f_1(X, Y_2) &= \frac{|ds_x - (ds_x + s_{ds})|}{s_{ds}} + \frac{|acc_x - (acc_x + s_{acc})|}{s_{acc}} + \frac{|dcc_x - (dcc_x - s_{dcc})|}{s_{dcc}} \\ &= 3 \end{aligned}$$

Es ist zu beachten, dass beim Auswählen von $\alpha > 3$ das Fahrzeug X auch die Fahrzeuge, deren Eigenschaften außerhalb der akzeptablen Abweichungen liegen, als Gruppenmitglieder akzeptiert. Aus diesem Grund soll der Wert von α kleiner oder gleich 3 gewählt werden (im Allgemein soll α kleiner oder gleich der Anzahl an Eigenschaften sein). In dieser Arbeit nehme ich $\alpha = 3$. Diese Auswahl basiert auf der Annahme, dass jede Eigenschaft einen gleichen Beitrag in Höhe von einer Einheit an dem Ergebnis der Funktion $f_1(X, Y)$ bringt.

Im Abschnitt 3.2 wurden die Einflüsse der Eigenschaften ds, acc, dcc an die Stabilität der Gruppe informell geklärt. Um die Stärke der Einflüsse von ds, acc, dcc formell darzustellen, benutze ich drei entsprechende Faktoren $\alpha_1, \alpha_2, \alpha_3$. Nach dem Einfügen der neuen Faktoren kann die Gleichung 3.3 wie folgt umgeschrieben werden:

$$f(X, Y) = \alpha_1 \frac{|ds_x - ds_y|}{s_{ds}} + \alpha_2 \frac{|acc_x - acc_y|}{s_{acc}} + \alpha_3 \frac{|dcc_x - dcc_y|}{s_{dcc}} \tag{3.4}$$

Damit die neu eingefügten Faktoren nicht die Bedeutung des α-Wertes verändern, setze ich eine Bedingung für $\alpha_1, \alpha_2, \alpha_3$:

$$\alpha_1 + \alpha_2 + \alpha_3 = \alpha \tag{3.5}$$

Die Bewertungsfunktion 3.4 hat, wie erwähnt, zwei wichtige Eigenschaften: Robustheit und Flexibilität. Die Robustheit der Bewertungsfunktion besteht darin, dass die Entscheidung von X darüber, ob das Fahrzeug Y an seiner Gruppe teilnehmen darf, nicht von den Änderungen der Verkehrsumgebung sondern von den statischen Eigenschaften des Fahrzeuges Y und von den Festlegungen der akzeptablen Abweichungen abhängt. Beim Einfügen der neuen $\alpha_1, \alpha_2, \alpha_3$ Faktoren, wird die Funktion $f(X, Y)$ flexibler. Das Fahrzeug X kann durch Umstellung der passenden $\alpha_1, \alpha_2, \alpha_3$ Werte selbst die Eigenschaften seiner Gruppe und damit die Menge der potentiellen Mitglieder dynamisch verändern.

3.2.3 Algorithmus für die Gruppenbildung

Dieser Abschnitt beschreibt die Nutzung der Bewertungsfunktion 3.4 zur Bildung von Fahrzeuggruppen. Es wird angenommen, dass jedes Fahrzeug die Änderungen der Umgebung nur innerhalb eines bestimmten Bereiches bemerken kann (z.B. die Änderungen der Positionen von Nachbarfahrzeugen). Diese Annahme liegt an der Beschränkung der heutigen Technologien, die ein autonomes Fahrzeug benutzt um die Umgebung wahrzunehmen (z.B. Wi-fi, Bluetooth, WiMax, Radiowelle usw.). Im Abschnitt 3.1 wurde beschrieben, dass die Gruppenbildungsaktion ein instinktives Verhalten jedes Fahrzeuges ist. Dieses Verhalten motiviert das Fahrzeug zu jedem Zeitpunkt einen passenden Nachbar für die Gruppenbildung zu finden. Der Algorithmus 2 beschreibt die Gruppenbildung eines Fahrzeuges Y. Es ist anzumerken, dass der Begriff „*direkte Nachbarn*“ in der Zeile 2 für die Fahrzeuge, die in der Nähe von Y sind, steht. Der Algorithmus 2 kann in zwei Phasen betrachtet werden. In der ersten Phase sucht das Fahrzeug Y beim Fragen der Nachbarn in seiner Nähe nach Informationen von Gruppenführern. Es ist zu beachten, dass jedes Mitglied einer Gruppe die statischen Eigenschaften seines Gruppenführers hat: wenn ein Nachbar von Y zu einer Gruppe gehört, dann kann der Nachbar immer die Informationen seines Gruppenführers an Y schicken.

Algorithm 2 Der Algorithmus für die Teilnahme an einer existierenden Gruppe und Begründung einer neuen Gruppe

1: Y ist in keine Gruppe (freies Fahrzeug).
2: Y fragt die *direkten* Nachbarn nach Gruppenführerinformationen
3: Für jeden gefundenen Gruppenführer X, berechnet Y den Wert $f(Y,X)$ und fügt $f(Y,X) \leq \alpha_Y$ in die Liste L_1 und den entsprechenden X in die Liste L_2 ein.
4: **while** L_1 ist nicht leer **do**
5: Y fragt einen Gruppenführer $X \in L2$, dessen Unähnlichkeitswert $f(X,Y) \in L_1$ am kleinsten ist, für die Teilnahme an dessen Gruppe.
6: X antwortet „JA“ wenn $f(X,Y) < \alpha_X$ sonst „Nein“
7: **if** Die Antwort des Gruppenführers X ist „Ja“ **then**
8: Y ist in der Gruppe G^X und bricht den Gruppenbildungsprozess ab.
9: **else**
10: entfernt X aus L_2 und den entsprechenden Wert $f(Y,X)$ aus L_1
11: **end if**
12: **end while**
13: **if** Y ist noch in keiner Gruppe und es existiert ein direkter Nachbar Z_i (oder mehrere direkte Nachbarn), sodass $f(Y,Z_i) \leq \alpha_Y$ und Y die kleinste ID-Nummer hat und Z_i zu keiner Gruppe gehört **then**
14: Y begründet eine Gruppe G^Y. Y ist in der Gruppe G^Y
15: **end if**

Das Fahrzeug Y benutzt die Bewertungsfunktion 3.4 zur Filterung der Gruppenführer, mit denen sich Y verbinden will. Das Fahrzeug Y wählt einen ihm möglichst

ähnlichen Gruppenführer X und sendet die Teilnahme-Anfrage an X. Der Führer X benutzt die Bewertungsfunktion 3.4 um zu entscheiden ob Y an seiner Gruppe teilnehmen darf. Wenn die Antwort von X „ja“ ist, beendet der Gruppenbildungsprozess erfolgreich. Die weite Phase wird durchgeführt wenn Y an keiner Gruppe teilnehmen kann. In dieser Phase überlegt Y seine eigene Gruppe zu begründen. Zunächst bewertet Y seine Nachbarfahrzeugen. Wenn Y einen Nachbar Z_i, mit dem es eine Gruppe bilden kann, findet, erstellt Y eine Gruppe G^Y und wartet auf die Teilnahme-Anfrage von Z_i. Allerdings kann Y nicht sicherstellen, dass Z_i an seiner Gruppe G^Y teilnimmt. Ist die Gruppe G^Y nach einer bestimmten Zeit noch leer, dann wird sie von Y gelöscht.

Wie im Abschnitt 3.2.2 erwähnt, hat jedes Fahrzeug nur einen beschränkten sichtbaren Bereich. Als Konsequenz kann sich Y manchmal nicht mit dem gefundenen Gruppenführer direkt verbinden. Die Zeile 5 beschreibt eine indirekte Anfrage von Y an den gefundenen Gruppenführer X. Diese erfolgt unter der Annahme, dass jedes Mitglied einer Gruppe die Rolle des Routers spielen kann. Deswegen kann jedes Mitglied mit allen anderen Mitgliedern seiner Gruppe kommunizieren (auch mit dem Gruppenführer). Weil Y mit einem Mitglied einer Gruppe G^X verbunden ist, kann Y durch das Mitglied auch indirekt mit X kommunizieren. Für die Einfachheit der späteren Implementation, nehme ich an, dass Y direkt eine Anfrage an den Gruppenführer X schicken kann.

3.3 Gruppenkonflikterkennung

Wie im Abschnitt 3.1 informell beschrieben, befinden sich zwei Gruppen $G_i^{V_i}$ und $G_j^{V_j}$ in einem Konflikt, wenn $G_j^{V_j}$ vor $G_i^{V_i}$ fährt und die Gruppe $G_j^{V_j}$ *schneller* als die Gruppe $G_i^{V_i}$ ist. Es wurde aber keine relative Position zwischen zwei Fahrzeuggruppen formell definiert. Im folgenden definiere ich die Konfliktsituation und die relativen Positionen zwischen zwei Fahrzeuggruppen.

Definition 3.3.1. Es sei $X_V(t)$ die Position des Fahrzeuges X zum Zeitpunk t. Die Fahrzeuge V_i und V_j sind Gruppenführer der Gruppe $G_i^{V_i}$ bzw. $G_j^{V_j}$. Die Konfliktsituation und die relativen Positionen zwischen der Gruppe G^{V_i} und der Gruppe G^{V_j} werden wie folgt definiert:

- $G_i^{V_i}$ ist hinter $G_j^{V_j}$ ($h(G_i, G_j)$) wenn:
  für alle $V_e \in G_i$ und für alle $V_f \in G_j$ gilt $X_{V_e}(t) < X_{V_f}(t)$

- $G_i^{V_i}$ ist vor $G_j^{V_j}$ ($v(G_i, G_j)$) wenn:
  für alle $V_e \in G_i$ und für alle $\in G_f$ gilt $X_{V_e}(t) > X_{V_f}(t)$

- $G_i^{V_i}$ und $G_j^{V_j}$ überlappen sich ($u(G_i, G_j)$) wenn:
  $\exists V_e \in G_i, \exists V_f \in G_j$ sodass $X_{V_e}(t) \geq X_{V_f}(t)$ und $\exists V_n \in G_i, \exists V_m \in G_j$ sodass $X_{V_n}(t) \leq X_{V_m}(t)$

- $G_i^{V_i}$ und $G_j^{V_j}$ sind in einem Konflikt $Conf(G_i, G_j)$ wenn:
  $h(G_i, G_j) \vee u(G_i, G_j)$ und $\exists V_i \in G_i, \exists V_j \in G_j$ sodass $ds_i - ds_j$ größer als s_{ds} von V_i ist . ds_i und ds_j sind die gewünschten Geschwindigkeiten von V_i bzw. V_j.

Die Definition 3.2.1 des Abschnittes 3.2 legte fest, dass der Gruppenführer auch der Vertreter seiner Gruppe ist. Beim Nutzen dieser Definition wird die Gruppenkonflikterkennung wie folgt beschrieben. Jedes Mitglied einer Gruppe ist ein Konfliktdetektor und enthält die Eigenschaften des Gruppenführers. Sobald es ein Mitglied einer anderen Gruppe in seinem sichtbaren Bereich entdeckt, fragt es das andere Mitglied nach dessen Gruppenführereingenschaften. Wenn es die Antwort des anderen Mitgliedes bekommt, dann leitet es die Antwort an seinen Gruppenführer weiter. Anhand der erhaltenen Antwort weiß der Gruppenführer ob sich seine Gruppe in einem Konflikt mit der anderen Gruppe befindet oder nicht.

3.4 Globale Koordination

Globale Koordination zielt darauf ab Konflikte zwischen Fahrzeuggruppen zu lösen. Die globale Koordination besteht aus Vereinbarungen zwischen Führern von Konfliktgruppen über die zu nutzenden Straßenspuren. Diese Gruppenspuren werden von den Gruppenführern an ihre Mitglieder übermittelt. Beim Fahren auf den Gruppenspuren können die Mitglieder einer Gruppe vermeiden von Mitgliedern einer anderen Gruppe blockiert zu werden.

Die Vereinbarungen der Gruppenspuren zwischen Gruppenführern werden wie folgt beschrieben. Jedem Gruppenführer wird eine Priorität zugeordnet, die sich nach seiner gewünschten Geschwindigkeit richtet: je größer die gewünschte Geschwindigkeit eines Gruppenführers ist, desto höher ist sein Priorität. Der Gruppenführer mit der höchsten Priorität darf zunächst die Spuren für seine Gruppe auswählen. Die Spurenauswahl erfolgt durch die Dominanz-Methode. Danach kalkuliert der Gruppenführer für jede Spur seiner Gruppe einen Dominanz-Wert. Ich definiere einen Dominanz-Wert für eine Spur x einer Gruppe G durch die folgende Funktion:

$$Do_x(G, S) = Num_x(G) - \sum_{G_a \in S} Num_x(G_a) \tag{3.6}$$

Wobei S die Konfliktgruppen der Gruppe G sind und die Gruppenführer von S langsamer als der Gruppenführer von G sind. Wenn eine Spur von keinem der Gruppenführer gewählt wird, dann wird sie als *frei* markiert. Ein Gruppenführer X_i von G wählt alle freie Spuren mit den Dominanz-Werten $Do_x(G, S) \geq 0$ als seine Gruppenspuren und markiert sie als *besetzt*. Falls keine Spur frei ist, oder es keine Spur mit dem Dominanz-Wert $Do_x(G, S) \geq 0$ gibt, dann wählt X_i eine Spur mit dem größten Dominanz-Wert. Die Abbildung 3.2 zeigt ein Beispiel mit drei Konfliktgruppen G_7, G_1 und G_9. Es wird angenommen, dass die Konflikte von den Gruppenführern V_7, V_1, V_9 wie folgt erkannt werden: $Conf(G_7, G_9)$, $Conf(G_7, G_1)$,$Conf(G_9, G_7)$. Laut der Definition über Konfliktgruppen werden die Ordnungen der Gruppenführerprioritäten wie folgt geschrieben:

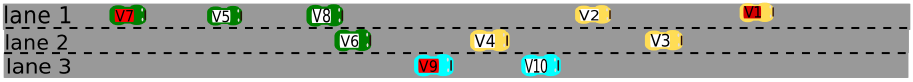


Abbildung 3.2: Die Fahrzeuge V_7, V_9, V_1 sind die Gruppenführer der Gruppen G_7, G_9, G_1. Die Konflikte zwischen G_7, G_9, G_1 sind $Conf(G_7, G_9)$, $Conf(G_7, G_1)$,$Conf(G_9, G_7)$

$V_7 > V9 > V_1$. Der Gruppenführer V_7 muss zunächst seine Gruppenspuren auswählen. Die Dominanz-Werte von $G7$ auf Spuren $1, 2, 3$ sind:

$$Do_1(G_7, [G_9, G1]) = 3 - 2 = 1$$
$$Do_2(G_7, [G_9, G1]) = 1 - 2 = -1$$
$$Do_3(G_7, [G_9, G1]) = 0 - 2 = -2$$

Weil $Do_1(G_7, [G_9, G1]) = 1$ positiv ist, wählt V_7 die Spur 1 als seine Gruppenspur. Weil die Spur 1 schon von V_7 gewählt wurde, bleiben für V_9 und V_7 nur die Spuren 2 und 3 übrig. Die Dominanz-Werte von G_9 sind:

$$Do_2(G_9, [G_1]) = 0 - 2 = -2$$
$$Do_3(G_9, [G_1]) = 2 - 0 = 2$$

Es ist erkennbar, dass der Gruppenführer V_9 die Spur 3 für seine Gruppe wählt. Die letzte Spur 2 wird von V_1 genommen.

3.5 Lokale Entscheidung des Fahrmanövers

Der Gruppenführer einer Gruppe benutzt Gruppenspuren zur globalen Kooperation seiner Mitglieder. Jedoch entscheidet jedes Mitglied, basierend auf seinem aktuellen Zustand, selbst über sein eigenes Fahrmanöver. Bevor ich das Fahrmanöver vorstelle, möchte ich die zwei wichtigsten Modelle „Folgemodell“ und „Spurwechselmodell“ für autonome Fahrzeuge vorstellen. Die beiden Modellen werden benutzt, um Geschwindigkeiten von autonomen Fahrzeugen zu regeln und die Spurwechselaktionen durchzuführen. Im Kapitel 2 wurden viele relevante Folge- und Spurwechselmodelle präsentiert. Diese Modelle wurden jedoch mit dem Ziel entwickelt, das Verhalten von menschlichen Verkehrsteilnehmern zu imitieren. Deswegen ist dort beispielsweise die Funktion „Verhandlungen zwischen Fahrzeugen“, nicht vorhanden (Weil zwei Verkehrsteilnehmer nicht mit einander direkt kommunizieren können). Im folgenden stelle ich ein Folgemodell und ein kooperatives Spurwechselmodell vor. Das kooperative Spurwechselmodell nutzt die Vorteile der Kommunikation zwischen Fahrzeugen um ein *Lücke-Problem* zu lösen.

3.5.1 Das Folgemodell für autonome Fahrzeuge

Es seien $V_a(ds_a, acc_a, dcc_a)$ und $V_b(ds_b, acc_b, dcc_b)$ zwei autonome Fahrzeuge auf der Spur s. Das Fahrzeug V_a folgt dem Fahrzeug V_b (siehe Abbildung 3.3). Je nach T Zeitperiode erhält das Fahrzeug V_a neue Informationen über seine Umgebung (z.B. Änderung

der Geschwindigkeit und aktuelle Position von V_b) und reagiert auf darauf [1]. Es sei $V_{V_a}(t+T)(V_a, V_b) \rightarrow \mathbb{R}$ ein Folgemodell, welches die Geschwindigkeit von V_a zum Zeitpunkt $t+T$ berechnet so dass keine Kollision zwischen V_a und V_b stattfindet. Das

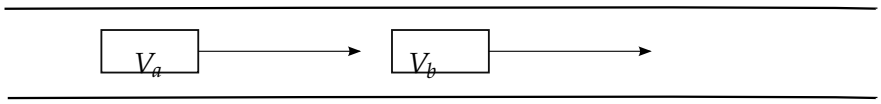


Abbildung 3.3: V_a verfolgt V_b

Folgemodell dieser Arbeit ist ein modifiziertes Modell von Gipps [18]. Wie im Abschnitt 2.5.4 vorgestellt, ist es für das Fahrzeug V_a sicherer die Geschwindigkeit $V_{V_a}(t+T)$ zum Zeitpunkt $t+T$ zu fahren, um noch rechtzeitig anhalten zu können, falls V_b zum Zeitpunkt t eine Notabbremsung durchführt. Das Folgemodell von Gipps lautet folgendermaßen:

$$V_{V_a}(t+T) = min\left[V_{max,V_a}, V_{V_a,V_b}\right] \tag{3.7}$$

Das Modell 3.7 ist eine Kombination von zwei Funktionen. Die erste Funktion V_{max,V_a} beschreibt die maximale Geschwindigkeit, die V_a in der Zeitperiode $t, t+T$ erreichen kann. Wenn der Vorgänger V_b sehr weit vorne ist, fährt V_a die Geschwindigkeit V_{max,V_a}. Die zweite Funktion V_{V_a,V_b} berechnet die Folgegeschwindigkeit von V_a (die Schreibweise V_{V_a,V_b} besagt, dass V_a ein Nachfolger von V_b ist). Je näher das Fahrzeug V_a an V_b heranfährt, desto kleiner wird die Folgegeschwindigkeit der Funkion V_{V_a,V_b}. Als Konsequenz wird diese Folgegeschwindigkeit von V_a als seine aktuelle Geschwindigkeit gewählt.

Meine erste Modifikation ist die erste Funktion V_{max,V_a}. Die originale Funktion von Gipps [18, 42] lautet:

$$V_{max,V_a}(t+T) = V_{V_a}(t) + 2.5A_{V_a}T(1 - \frac{V_{V_a}(t)}{V_{V_a}})\sqrt{0.025 + \frac{V_{V_a}(t)}{V_{V_a}}} \tag{3.8}$$

In der Funktion 3.8 benutzt Gipps zwei Parameter 2.5 und 0.025 um das Verhalten des menschlichen Verkehrsteilnehmers zu imitieren. Jedoch wird mein Fahrzeug autonom vom Computer gesteuert. Aus diesem Grund benutze ich die folgende Funktion für das Beschleunigungsverhalten:

$$V^{new}_{max,V_a}(t+T) = min(V_{V_a}(t) + acc_a * T, ds_a) \tag{3.9}$$

Die Funkion 3.9 berechnet die maximal erreichbare Geschwindigkeit des Fahrzeuges V_a in der Zeitperiode $(t, t+T)$. Die interne Funkion $V_{V_a}(t) + acc_a * T$ berechnet die maximale Geschwindigkeit, auf die das Fahrzeug V_a in der Zeitperiode $(t, t+T)$ beschleunigen kann. Der Parameter ds_a ist die gewünschte Geschwindigkeit von V_a. Die Funkion 3.9 setzt die maximale Geschwindigkeit $V^{new}_{max,V_a}(t+T)$ auf das Minimum aus erreichbarer Geschwindigkeit $V_{V_a}(t) + acc_a * T$ und gewünschter Geschwindigkeit ds_a von V_a.

[1] In vielen Verkehrssimulationssystemen, wird die Zeitperiode T als der Zeitabstand zwischen zwei Simulationsschritten genommen

Meine zweite Modifikation betrifft die zweite Funktion V_{V_a,V_b}. Bei Gipps lautet die Funkion V_{V_a,V_b} wie folgt:

$$V_{V_a,V_b}(t+T) = D_{V_a}T + \sqrt{D_{V_a}T - D_{V_a}\begin{pmatrix} 2\left(X_{V_b}(t) - X_{V_a}(t) - L_{V_b} - M_{V_a}\right) \\ -V_{V_a}(t)T - V_{V_b}^2(t)0.5 \end{pmatrix}} \tag{3.10}$$

Wie bereits in Abschnitt 2.5.4 erwähnt, muss das Fahrzeug V_a die Halteposition X_{V_b} von V_b berechnen um V_{V_a,V_b} zu erhalten. Wegen fehlender Kommunikation zwischen V_a und V_b muss V_a die maximale Verzögerung von V_b einschätzen. Als Konsequenz kann V_a die Halteposition X_{V_b} von V_b nicht genau berechnen. Die Funktion 3.11 wird von V_a benutzt um X_{V_n} einzuschätzen.

$$X_{V_b} = X_{V_b}(t) - \frac{V_{V_b}^2(t)}{2dcc_b^*} \tag{3.11}$$

Wobei dcc_b^* ist die von V_a eingeschätzte maximale Verzögerungsrate von V_b. Weil die autonomen Fahrzeuge wie angenommen in der Lage sind mit einander zu kommunizieren, tauschen sie ihre statischen Eigenschaften aus. Diese bedeutet, V_a verwendet die maximale Verzögerungsrate dcc_b in seiner Berechnung der Halteposition von V_b. Deshalb wird die Funktion X_{V_b} von mir wie folgt modifiziert:

$$X_{V_b}^{new} = X_{V_b}(t) - \frac{V_{V_b}^2(t)}{2dcc_b} \tag{3.12}$$

Damit ergibt sich meine Modifikation der Funktion V_{V_a,V_b} von Gipps zu:

$$V_{V_a,V_b}^{new}(t+T) = D_{V_a}T + \sqrt{D_{V_a}T - D_{V_a}\begin{pmatrix} 2\left(X_{V_b}^{new}(t) - X_{V_a}(t) - L_{V_b} - M_{V_a}\right) \\ -V_{V_a}(t)T - V_{V_b}^2(t)0.5 \end{pmatrix}} \tag{3.13}$$

Damit lautet mein modifiziertes Folgemodell wie folgt:

$$V_{V_a,V_b}^{follow}(t+T) = min\left[V_{max,V_a}^{new}, V_{V_a,V_b}^{new}\right] \tag{3.14}$$

3.5.2 Das Spurwechselmodell für autonome Fahrzeuge

Im Abschnitt 2.6 habe ich die zwei wichtigen Phasen einer Spurwechselaktion vorgestellt. In der ersten Phase werden die Motivationen für einen Spurwechsel bewertet. In der zweiten Phase verwendet ein Fahrer das „Gap Acceptance Model (GAP)"-Modell um die Größe einer freien Lücke auf der Zielspur zu messen. Wenn die Lücke groß genug ist, wird der Spurwechsel durchgeführt. Ansonsten bleibt der Fahrer auf seiner aktuellen Spur. Das Hauptproblem der traditionellen Spurwechselmodelle liegt darin, dass die Spurwechselaktion scheitert, wenn es keine ausreichend große Lücke für den Spurwechsel gibt. Weil keine Kommunikation zwischen Fahrzeugen stattfindet, können die Fahrer nicht miteinander verhandeln um die nötige Lücke zu erzeugen. Dieses

Problem wird als *Lücke-Problem* bezeichnet. In dieser Arbeit nutzen die autonomen Fahrzeuge ihre Kombinationsfähigkeiten zur Lösung des *Lücke-Problems*. Dadurch kann eine Spurwechselaktion sicher und effizient durchgeführt werden. In diesem Abschnitt stelle ich eine kooperative Spurwechselmethode vor. Diese Methode erlaubt autonomen Fahrzeugen miteinander zu kooperieren um die ausreichend große Lücke für den Spurwechsel zu erzeugen. Es wird angenommen, dass die Fahrzeuge schon vorher motiviert waren ihre Spur zu wechseln. Im Abschnitt 3.5.3 beschreibe ich die Motivationen eines Spurwechsels konkreter.

Die Abbildung 3.4 beschreibt das Problem des Spurwechsels. Es wird angenommen, dass zum Zeitpunkt t das Fahrzeug V_c auf die Spur 1 wechseln möchte. Auf der Spur 1 fahren V_a und V_b. Die Länge der Pfeile der Fahrzeuge bezeichnet die Bremswege, die

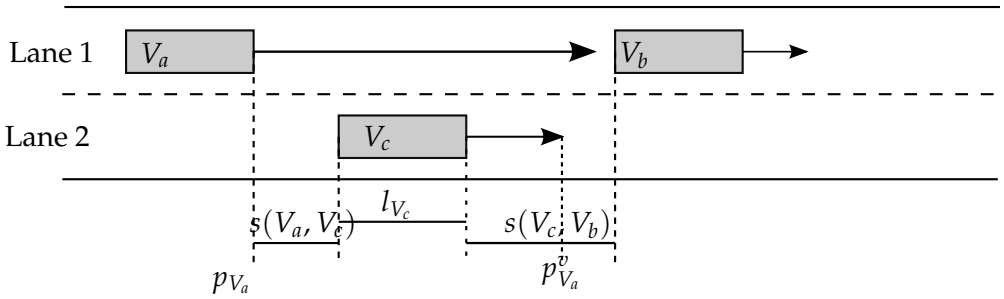


Abbildung 3.4: V_c versucht zur Spur von V_a und V_b zu wechseln

die Fahrzeuge brauchen um anzuhalten.

Es ist erkennbar, falls V_c die Spurwechselaktion durchführt, die Gefahr besteht, dass V_c mit V_a kollidiert. Um das Problem des Spurwechsels zu lösen, stelle ich die Methode „Cooperative Collison Resolve(CCR)“ vor. Zunächst formuliere ich die mathematische Bedingungen für einen sicheren Spurwechsel. Laut der Theorie von Gipps [19] kann das Fahrzeug V_c zum Zeitpunkt t nur zur Spur 1 wechseln wenn:

1. Es eine Lücke neben der aktuellen Position von V_c gibt. Diese Bedingung kann mathematisch wie folgt beschrieben werden:

$$s(V_a, V_c) = p_{V_c} - l_{V_c} - p_{V_a} \wedge s(V_a, V_c) \geq 0 \tag{3.15}$$
$$s(V_c, V_b) = p_{V_p} - l_{V_b} - p_{V_c} \wedge s(V_c, V_b) \geq 0 \tag{3.16}$$

   Wobei die Parameter p_{V_a}, p_{V_c} die Positionen der Fahrzeuge V_a bzw. V_c sind. Der Parameter $s(V_a, V_c)$ ist der Abstand zwischen V_a und V_b (siehe Abbildung 3.4). Der Parameter l_{V_c} ist die Länge von V_c.

2. V_a kann V_c als seinen nächsten Vorgänger einsetzen. Dabei muss V_c im Falle seines Spurwechsels und einer Notbremsung sicher stellen, dass V_a immer noch hinter ihm anhalten kann. Sei

$$V_{min,V_a}(t+T) = max(V_{V_a}(t) + dcc_a * T, 0) \tag{3.17}$$

die minimale Geschwindigkeit, auf die das Fahrzeug V_a in der Zeit T abbremsen kann. Die Bedingung für V_a wird wie folgt darstellt.

$$V^{new}_{V_a,V_c} \geq V_{min,V_a}(t+T) \tag{3.18}$$

Wobei $V^{new}_{V_a,V_c}(t+T)$ mit Hilfe der Funktion 3.13 berechnet wird. Diese Bedingung muss nur dann nicht erfüllt werden, wenn das Fahrzeug V_c keinen Nachfolger V_a auf seiner Zielspur hat.

3. V_c kann V_b als seinen nächsten Vorgänger einsetzen. Die Bedingung für V_c ergibt sich analog zur Bedingung des V_a:

$$\begin{aligned} V_{min,V_c}(t+T) &= max(V_{V_c}(t) + dcc_c * T, 0) \\ V^{new}_{V_c,V_b} &\geq V_{min,V_c}(t+T) \end{aligned} \tag{3.19}$$

Wobei $V_{min,V_c}(t+T)$ die minimale Geschwindigkeit ist, auf die das Fahrzeug V_c in der Zeit T abbremsen kann. Die Geschwindigkeit $V^{new}_{V_c,V_b}$ wird mit Hilfe der Funkion 3.13 berechnet. Diese Bedingung muss nur dann nicht erfüllt werden, wenn V_c keinen Vorgänger V_b auf seiner Zielspur hat.

Im nächsten Abschnitt beschreibe ich die CCR-Methode.

CCR-Methode

Das Ziel der CCR-Methode ist es, einen gemeinsamen Plan für V_a, V_b und V_c (siehe Abbildung 3.4) aufzustellen, um die Bedingungen für den Spurwechsel zu erfüllen. Eine bekannte Methode ist, statischen Pläne für jedes Fahrzeug V_a, V_b, V_c zu generieren. Das bedeutet, für jedes V_a, V_b, V_c eine Reihenfolge von Aktionen (Beschleunigung, Verzögerung, und Spurwechsel) vorzudefinieren. Beim Ausführen dieser Aktionen werden die obigen Bedingungen zu einem bestimmten Zeitpunkt erfüllt. In einer dynamischen Verkehrsumgebung ist diese Methode sehr kompliziert und nicht effizient. Der Grund liegt darin, dass man jede Aktion der Fahrzeuge zu einem festgelegten Zeitpunkt definieren muss. Wegen der Änderungen der Umgebung sind die vordefinierten Aktionen jedoch nicht effizient (manchmal nicht ausführbar).

Beispielsweise ist es schwer für das Fahrzeug V_c eine effiziente Reihenfolge der Aktionen zu finden, wenn es noch mehrere Fahrzeuge gibt, die vor V_c und V_b fahren. Hält V_b seine Geschwindigkeit in nT Zeiteinheiten konstant und reduziert V_a seine Geschwindigkeit in nT Zeiteinheiten, dann passiert bei der Ausführung der statischen Pläne von V_c und V_b folgendes: Nach $iT < nT$ Zeiteinheiten der Ausführung seiner vordefinierten Aktionen merkt V_c, dass die Geschwindigkeit von V_b erhöht werden könnte (weil z.B. der Vorgänger von V_b seine Spur erfolgreich gewechselt hat). Dadurch könnte auch V_c seine Geschwindigkeit erhöhen. Allerdings muss V_b seine Geschwindigkeit wegen der festgelegten Aktionen unverändert halten. Deswegen sind die vordefinierten Aktionen für V_b und V_c nicht mehr effizient.

Mein Lösungskonzept für das vorstehende Problem besteht darin, die Geschwindigkeiten der Fahrzeuge V_a, V_b und V_c dynamisch und abhängig von einander in einer

bestimmten Zeit zu verändern. Dies bedeutet, dass V_a seine Geschwindigkeit in Abhängigkeit der Geschwindigkeit von V_c verändert und V_c seine Geschwindigkeit in Abhängigkeit der Geschwindigkeit von V_b verändert. Aus diesem Grund konstruiere ich die folgenden Folgemodelle zur Regelung der Geschwindigkeit für V_a:

$$V^{ccr}_{V_a,V_c}(t+T) = max\left[V^{follow}_{V_a,V_c}, V_{min,V_a}(t+T)\right] \tag{3.20}$$

und für V_c:

$$V^{ccr}_{V_c,V_b}(t+T) = max\left[V^{follow}_{V_c,V_b}, V_{min,V_c}(t+T)\right] \tag{3.21}$$

Die obigen Folgemodelle erweitern die Folgemodell 3.13. Das Verhalten von V_a, welches durch das Folgemodell 3.20 beschrieben wird, kann folgendermaßen erklärt werden.

Das Fahrzeug V_a berechnet zunächst seine Folgegeschwindigkeit im Bezug auf das Fahrzeug V_c mit Hilfe der Funkion $V^{follow}_{V_a,V_c}$. Wenn diese Folgegeschwindigkeit größer als die minimale Geschwindigkeit V_{min,V_a} ist, die V_a in der Zeit T fahren kann, dann kann V_a seine Folgegeschwindigkeit ohne Kollisionsgefahr mit V_c fahren. Jedoch wenn die Folgegeschwindigkeit kleiner als die minimale Geschwindigkeit V_{min,V_a} ist, dann muss V_a seine Geschwindigkeit auf die minimale Geschwindigkeit V_{min,V_a} reduzieren. Diese Reduzierung wird durchgeführt bis die Geschwindigkeit von V_a kleiner oder gleich der Folgegeschwindigkeit $V^{follow}_{V_a,V_c}$ ist. Die Vorteile des CCR- Modells 3.20 liegen darin, dass das Fahrzeug V_a die aktuelle Geschwindigkeit von V_c als Parameter der Berechnung seiner eignen Geschwindigkeit nutzt. Es adaptiert seine Geschwindigkeit an der Änderung der Geschwindigkeit von V_c, so dass die Bedingung 3.18 erfüllt wird.

Analog wie V_a, benutzt V_c das Modell 3.21 beim Setzen des V_b als seinen Vorgänger um adaptiv seine Geschwindigkeit an die Änderung der Geschwindigkeit von V_b anzupassen.

Somit ergibt sich die folgende Frage:
Wie lange müssen V_a und V_c ihre Vorgänger V_c bzw. V_b folgen?.
Um diese Frage zu beantworten, betrachte ich zunächst die drei Halteposition $p^s_{V_a}$, $p^s_{V_b}$, $p^s_{V_c}$ von V_a,V_b und V_c. Es seien $p^v_{V_a}$, $p^v_{V_b}$, $p^v_{V_c}$ die virtuellen Postionen von V_a, V_b und V_c auf Zielspur von V_c (Spur 1). Die virtuellen Positionen lassen sich durch die Gleichungen

$$\begin{aligned} p^v_{V_a} &= max\left[p^s_{V_a}, p^s_{V_c}\right] \\ p^v_{V_c} &= p^v_{V_a} + l_{V_a} + buff_{V_a} \\ p^v_{V_b} &= max\left[p^s_{V_b}, p^v_{V_c} + l_{V_c} + buff_{V_c}\right] \end{aligned} \tag{3.22}$$

berechnen. Die Abbildung 3.5 stellt die virtuellen Positionen von V_a, V_c und V_b dar. Der Parameter $buff_V$ bezeichnet einen minimalen Abstand zwischen zwei Fahrzeugen. Es ist erkennbar, dass die virtuellen Positionen eine künftige Zuordnung von V_a,V_b,V_c auf der Zielspur von V_c sind. Beim Folgen des Fahrzeuges V_b kann V_c seine Spur spätestens an der Position $p^v_{V_c}$ wechseln. Es sei $t_V(p_V)$ die minimale Zeit, die ein Fahrzeug V braucht, um eine Position p_V zu erreichen. Dann ist die Zeit $t^{ccr}_{V_c}$ für den Spurwechsel von V_c das Maximum der Zeiten, die die Fahrzeuge V_a, V_c, V_b brauchen um ihre

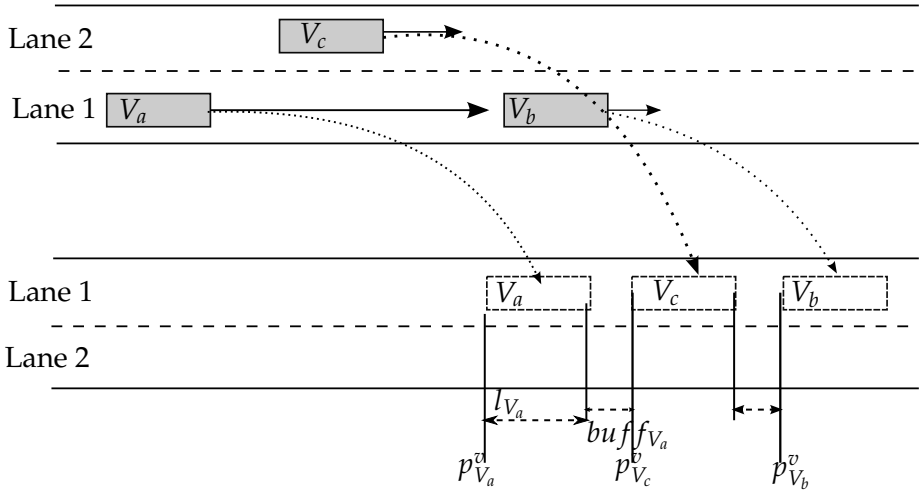


Abbildung 3.5: Darstellung der virtuellen Positionen von V_a, V_c, V_b

virtuellen Positionen zu erreichen. Die Zeit $t^{ccr}_{V_c}$ lässt sich durch die folgende Gleichung ermittelt:

$$t^{ccr}_{V_c} = max\left[t_{V_c}(p^v_{V_c}), t_{V_b}(p^v_{V_b}), t_{V_a}(p^v_{V_a})\right] \tag{3.23}$$

Die Antwort der vorstehenden Fragen kann wie folgt formuliert werden:
Nach $t^{ccr}_{V_c}$ Zeiteinheiten folgen V_a und V_c ihrem Vorgänger V_c bzw. V_b nicht mehr.
Bei der Berechnung der maximalen Spurwechselzeit von V_c wurde angenommen, dass alle Fahrzeuge V_a, V_b und V_c nicht von der Umgebung gestört wurden. Jedoch im wirklichen Straßenverkehr werden V_a, V_b und V_c sehr oft von anderen Fahrzeugen beeinflusst. Zum Beispiel: V_b muss wegen seines Vorgängers seine Geschwindigkeit langsam halten. Folglich muss V_c sich verlangsamen. Aus diesem Grund ist es schwierig für V_c eine optimale Zeit für seine Folgen-Aktion zu berechnen. Deshalb ersetze ich solche Störungen durch einen α-Wert. Folglich ist die maximale Zeit für die Folgen-Aktionen von V_a und V_c gleich $t^{ccr}_{V_c} + \alpha$.

Bei der Konstruktion eines gemeinsamen Planes für seinen Spurwechsel braucht das Fahrzeug V_c die Kooperation von V_a. Nur wenn das Fahrzeug V_a akzeptiert dem Fahrzeug V_c zu folgen, kann eine sichere freie Lücke für V_c erzeugt werden. Das Fahrzeug V_b nimmt die Rolle als Prozess-Monitor ein. Bemerkt V_b, dass es seine virtuelle Position nicht in $t^{ccr}_{V_c}$ Zeiteinheiten erreichen kann, dann benachrichtigt es V_a und V_c um deren Folge-Aktionen abzubrechen. Vielen Verkehrsszenarios bestehen nur aus zwei Fahrzeugen. Nimmt man beispielsweise an, dass das Fahrzeug V_b nicht existiert, dann wird die CCR-Methode nur auf die beiden Fahrzeuge V_a und V_c angewendet. Das heißt, das Fahrzeug V_a folgt dem Fahrzeug V_c und weil V_c keinem Vorgänger zu folgen hat, verwendet V_c das normale Folgemodell 3.14.

3.5.3 Lokale Entscheidung des Fahrmanövers

Wie im Abschnitt 2.6 beschrieben, ist ein Spurwechsel entweder vom Typ MLC oder DLC. In dieser Arbeit nehme ich an, dass die autonomen Fahrzeuge nur auf einer Autobahnstrecke fahren müssen und die Strecke fehlerfrei ist. Das bedeutet, dass die Motivationen eines Spurwechsels wie „Ausfahrt erreichen“ oder „Baustelle vermeiden“ nicht existieren. Im Abschnitt 3.2 wurde die Gruppenbildungsmethode vorgestellt. Jedoch ist es für ein Fahrzeug nicht immer möglich seine gewünschte Gruppe zu finden. Ich unterscheide zwei Gruppenzustände von Fahrzeugen: 1. freie Fahrzeuge (FF) und 2. einer Gruppe zugehöriger Fahrzeuge (GF). Wenn ein Fahrzeug im FF-Zustand ist, entscheidet es über seinen Spurwechsel mit Hilfe einer Nutzenfunktion. Die Nutzenfunktion berechnet für jede Spur einen Nutzwert. Die Spur mit dem größten Nutzwert wird gewählt. Ein Spurwechselentscheidung, die auf Nutzwerten basiert, ist immer ein DLC-Spurwechsel. Der Unterschied zwischen einem MLC-Spurwechsel und einem DLC-Spurwechsel liegt darin, dass ein DLC-Spurwechsel nicht betrachtet wird wenn er eine Kollisionsgefahr verursacht. Wenn ein Fahrzeug ein MLC-Spurwechsel durchführen muss und eine Kollisionsgefahr besteht, dann benutzt das Fahrzeug die CCR-Methode um die Kollisionsgefahr zu beseitigen. Wenn ein Fahrzeug im GF-Zustand ist, hängt seine Spurwechselentscheidung von seiner aktuellen Position, Spur, Geschwindigkeit, Gruppenspuren ab. Im folgenden beschreibe ich die Nutzenfunktion zur Berechnung des Nutzwertes einer Spur und die Spurwechselentscheidung eines GF-Fahrzeuges.

Nutzwert-basierte Spurwechselentscheidung

Die Entscheidung über die künftige Spur eines Fahrzeuges erfolgt durch Bewertung der Nutzwerte der Spuren neben ihm. Ein normaler Verkehrsteilnehmer betrachtet den Nutzwert einer Spur als die maximale erreichbare Geschwindigkeit, die die Spur ihm erlaubt. Im Unterschied dazu verwendet ein autonomes Fahrzeug die verlorene Geschwindigkeit des anderen Fahrzeuges in der Nutzwertberechnung. Das autonome Fahrzeug V wird die Spur auswählen, auf der es nicht nur seine gewünschte Geschwindigkeit bestens erreicht, sondern auch die gewünschte Geschwindigkeit seines Nachfolgers wenigstens blockiert. Aus den obigen Grund wird die Bewertungsfunkion wie folgt konstruiert:

$$Util_x(V) = [V^{ccr}_{V,V_{pre}}(t+T) - V^{new}_{max,V}(t+T)] + [V^{ccr}_{V_{succ},V}(t+T) - V^{new}_{max,V_{succ}}(t+T)] \quad (3.24)$$

Wobei $Util_x$ der Nutzwert der Spur x ist. $V^{new}_{max,V}(t+T)$ ist die maximale Geschwindigkeit, die das Fahrzeug V in der Zeitperiode $t+T$ erreichen kann (siehe die Funkion 3.9). $V^{ccr}_{V,V_{pre}}(t+T)$ steht für die potentielle erreichbare Geschwindigkeit, die das Fahrzeug V erreichen kann wenn es zur Spur x wechselt. $V^{ccr}_{V_{succ},V}(t+T)$ ist die potentielle Geschwindigkeit des Nachfolgers V_{succ} von V auf der Spur x.

Spurwechselentscheidung eines autonomen GF-Fahrzeuges

Im Gegensatz zu den FF-Fahrzeugen wird die Spurwechselentscheidung eines GF-Fahrzeuges von Faktoren wie: aktuelle Position, aktuelle Spur, aktuelle Geschwindigkeit, Gruppenspuren, beeinflusst. Wie ich am Anfang dieses Abschnittes vorgestellt habe, haben die Gruppenspuren die Aufgabe der globalen Koordination zur Konfliktgruppenvermeidung.

Jedoch muss ein GF-Fahrzeug nicht immer auf seinen Gruppenspuren fahren. Als Beispiel wird das Szenario in Abbildung 3.6 betrachtet. Wie in Abbildung 3.6 gezeigt,

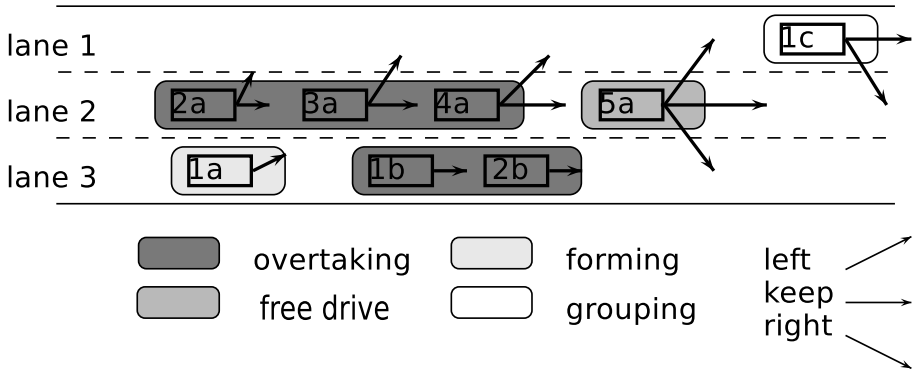


Abbildung 3.6: Konfliktgruppen

überlappen sich die Gruppe $G_a = \{1a, 2a, 3a, 4a, 5a\}$ und die Gruppe $G_b = \{1b, 2b\}$. Angenommen es existiert einen Konflikt $Conf(G_a, G_b)$ zwischen G_a und G_b. Dann wählt der Gruppenführer von G_a die Spuren 1 und 2 für seine Gruppe und verbreitet die Gruppenspuren an alle Mitglieder, um die G_b zu überholen. Es ist sichtbar, dass das Fahrzeug $5a$ nicht in Konflikt mit allen Mitgliedern von G_a steht. Es fährt vor allen anderen Mitgliedern der Konfliktgruppe G_b. Aus diesem Grund ist es für das Fahrzeug $5a$ nicht nötig seinen Gruppenspuren zu folgen. Es kann deswegen eine beste Spur zum Fahren auswählen. Ich definiere vier Zustände für ein autonomes Fahrzeug. Im jedem Zustand, wird der Spurwechsel entweder vom Typ MLC oder DLC definiert.

- `Grouping` (Gruppierung): Dieser Zustand beschreibt, dass ein autonomes Fahrzeug noch zu keiner Gruppe gehört. In diesem Zustand ist das Fahrzeug ein FF-Fahrzeug. Es wählt immer die Spur, die seinem Nutzwert maximiert. In dem Fall, dass die nächste Spur einen Spurwechsel verursacht (Die gewählte Spur ist nicht seine aktuelle Spur), ist der Typ des Spurwechsels DLC. Das Fahrzeug $1c$ im obigen Beispiel befindet sich in diesem Zustand.

- `Forming` (Bildung): Dieser Zustand beschreibt, dass ein Fahrzeug zu einer Gruppe gehört und nicht auf einer seiner Gruppenspuren fährt. In diesem Zustand versucht das Fahrzeug auf eine Spur, die eine seiner Gruppenspuren ist, zu wechseln. Der Spurwechsel ist ein MLC-Spurwechsel. Das Fahrzeug des obigen Beispiel $1a$ befindet sich in diesem Zustand.

- `Overtaking` (Überholung): Dieser Zustand beschreibt, dass ein Fahrzeug zu einer Gruppe gehört und auf einer seiner Gruppenspuren fährt. In diesem Zustand verwendet das Fahrzeug die Nutzwertmethode um seine gewünschte Spur zu finden. Im Unterschied zum `Grouping`-Zustand wird das Fahrzeug nur eine der Gruppenspuren als seine nächste gewünschte Spur betrachten. Die Spurwechselentscheidung basiert auf den Nutzwerten der Gruppenspuren. Der Spurwechsel dieses Zustandes ist ein DLC-Spurwechsel.

- `Free Drive` (Freie Fahrt): Dieser Zustand beschreibt, dass das Fahrzeug zu einer Gruppe gehört und auf einer seiner Gruppenspuren fährt. Jedoch muss es nicht unbedingt auf einer der Gruppenspuren fahren. Wenn das Fahrzeug kein Gruppenführer ist, verhält es wie ein Fahrzeug im `Grouping`-Zustand. Es wird definiert, dass falls das Fahrzeug zu einer Spur wechselt, die keine der Gruppenspuren ist, es seine Gruppe verlässt. Ist das Fahrzeug ein Gruppenführer, dann verhält es sich wie ein Fahrzeug des `Overtaking`-Zustandes. Ein Spurwechsel in diesem Zustand ist ein DLC. Das Fahrzeug $5a$ in Abbildung ist dafür ein Beispiel. Weil $5a$ kein Gruppenführer ist, kann es zur Spur $1, 2$ oder 3 wechseln. Wenn das Fahrzeug $5a$ zur Spur 3 wechselt, verlässt es seine Gruppe.

Im Kapitel 5 wird ein Computer-Agent als ein virtueller Fahrer eines autonomen Fahrzeuges konstruiert. Die beschriebenen Zustände spielen eine wichtige Rolle zur Aktivierung eines geeigneten Verhaltens um ein Agentenziel zu erreichen.

3.6 Zusammenfassung

In diesem Kapitel wurde die gruppenorientierte Fahrmethode vorgestellt. Dieser Abschnitt fasst die wichtigen Aspekte der gruppenorientierten Methode zusammen.

- Gruppenbildung ist ein instinktives Verhalten von autonomen Fahrzeugen. Die autonomen Fahrzeugen bilden sich in Gruppen potentielle Konflikte mit anderen Fahrzeugen zu entdecken und zu lösen.

- Je kleiner die Eigenschaftsunterschiede zwischen den Mitgliedern einer Gruppe sind, desto stabiler ist die Fahrzeuggruppe.

- Die Gruppenkonfliktlösung ist die Vereinbarung der Gruppenspuren zwischen Gruppenführern. Die Entscheidung eines Gruppenführers über seine Gruppenspuren wird mit Hilfe der Dominanz-Methode getroffen.

- Ein Mitglied einer Gruppe muss nicht immer auf seinen Gruppenspuren fahren. Die lokale Entscheidung über seine Fahraktion hängt von seinem aktuellen Zustand ab. Ein Fahrzeug befindet sich immer in einem der `Grouping`, `Forming`, `Overtaking`, `Free Drive` Zustände.

- Die modifizierten Folge- und Spurwechselmodelle zielt sich nicht darauf die natürlichen Verhalten von Verkehrsteilnehmern zu imitieren sondern sie nutzen die

Vorteile der Kommunikationen zwischen Fahrzeugen zur effizienteren Konstruktion des Verhalten von autonomen Fahrzeugen.

- Die CCR-Methode wurde zur Lösung des Lückenproblems eines Spurwechsels entwickelt. Sie erlaubt allen relevanten Fahrzeugen, die das Lückenproblem verursachen, miteinander zu kooperieren um eine genug große Lücke für den Spurwechsel zu erzeugen. Die CCR-Methode wird nur benutzt, wenn ein Fahrzeug im `Forming`-Zustand ist.

4 ATSim : Das Agenten-gesteuerte Verkehrssimulationssystem

4.1 Einleitung

Das Ziel dieser Arbeit ist die Antwort für die folgende Frage zu finden:
Fahren die Fahrzeuge mit Hilfe der gruppenorientierten Fahrmethode schneller zu ihren Zielen im Vergleich mit den Fahrzeugen, die keine gruppenorientierte Methode, benutzen?
Damit das Ziel realisiert werden kann, sind die Verkehrsdaten, die sich aus den beiden Fahrmethoden ergeben, zum Vergleichen nötig. Es ist sehr aufwendig (Zeit und Kosten) die beiden Fahrmethoden mit echten Fahrzeugen zu testen. Deshalb simuliere ich die beiden Fahrmethoden mit Hilfe eines Computer-Programms. Im Kapitel 2 wird das AIMSUN -Verkehrssimulationssystem vorgestellt. AIMSUN simuliert das natürliche Verhalten von Verkehrsteilnehmern und speichert die Verkehrsdaten in einer Datenbank ab. Wie bei allen anderen Verkehrssimulationssystemen, erlaubt AIMSUN den simulierten Fahrzeugen nicht miteinander zu kommunizieren. Weil die gruppenorientierte Fahrmethode sehr stark von der Kommunikationsfähigkeit der autonomen Fahrzeuge abhängt, muss diese Beschränkung von AIMSUN behoben werden. AIMSUN bietet anderen externen Softwareprogrammen die Möglichkeit auf alle simulierten Verkehrsobjekte (Fahrzeuge, Straßenampel usw) via API zuzugreifen. Dadurch können die externen Programme die simulierten Fahrzeuge im AIMSUN steuern. Beim Nutzen dieses Vorteils, kombiniere ich AIMSUN mit einem Multiagentensystem (MAS) um die Fahrzeuge von AIMSUN kommunikationsfähig zu machen. Das neue System wird als Agenten-gesteuertes Verkehrssimulationssystem (ATSim) bezeichnet. Die Abkürzung ATSim steht für den englischen Begriff: „**A**gent-based **T**raffic **Sim**ulation System“

Das ATSim besteht aus vier Komponenten: AIMSUN Simulation Model(ASM), MAS-Connector (MASC), MAS-Service (MASS) und Agent Manager (AM). Die Abbildung 4.1 stellt die abstrakte Struktur des ATSim dar. Informell wird das ATSim wie folgt beschrieben. Ein Verkehrsszenario wird mit Hilfe von AIMSUN entworfen. Der Simulator von AIMSUN nimmt das Szenario als Eingabe für seinen Simulationsprozess. Der MASC ist ein externes Programm, welches die Rolle als Verbindungsglied zwischen den autonomen Agenten und den Fahrzeugen des AIMSUN übernimmt. Vor der Simulation des Verkehrsszenarios, wird der MASC vom AIMSUN -Simulator geladen. In jedem Simulationsschritt sammelt der MASC die für die Agenten nötigen Daten aller simulierten Verkehrsobjekte und transferiert sie an das MAS. Jedes Fahrzeug des AIMSUN wird von einem autonomen Agenten im MAS gesteuert. Beim Aufruf einer MASS-Dienstes werden die Daten eines Fahrzeuges an den entsprechenden Agenten übermittelt. Nachdem alle Agenten die Daten von AIMSUN bekommen haben, kommunizie-

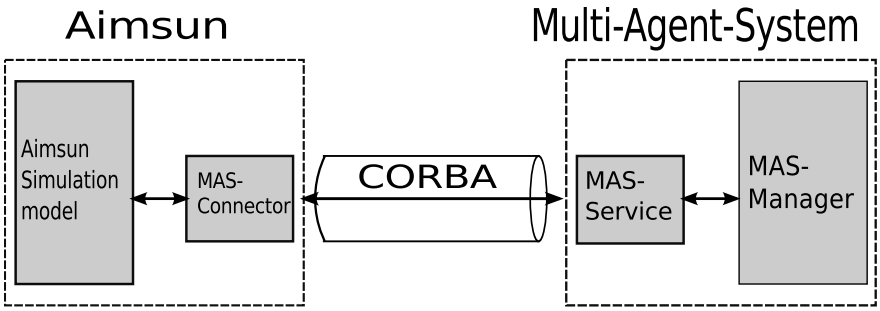


Abbildung 4.1: Die Struktur des Agenten-gesteuerten Verkehrssimulationssystems(ATSim)

ren sie miteinander und geben die Steuerbefehle an die entsprechenden Fahrzeuge des AIMSUN zurück. Die ASM-Komponente der Abbildung ist ein Bestandteil des AIMSUN . In dieser Arbeit beschreibe ich nicht wie ein ASM konstruiert wird und vom AIMSUN -Simulator ausgeführt wird [1], sondern konzentriere mich auf die Darstellung der Struktur des MASC und des Multi Agenten Systems. Wie Abbildung 4.1 zeigt, ist das MAS eine Komposition von zwei Komponenten: MASS und AM. In den nächsten Abschnitten wird jede Komponente konkreter beschrieben. Dieses Kapitel ist wie folgt aufgebaut. Im Abschnitt 4.2 beschreibe ich die Struktur und die Funktionalitäten des MASC. Der Abschnitt 4.3 beschreibt die wichtigsten Dienste der MASS. Diese Dienste werden benutzt um die Daten zwischen dem AIMSUN und dem MAS auszutauschen. Im Abschnitt 4.4 wird beschrieben, wie das MAS die Agenten verwaltet.

4.2 MAS-Connector

Die Abkürzung MASC steht für einen Multiagentensystem-Verbinder. Der MASC ist eine Schnittstelle für die Interaktion zwischen AIMSUN und MAS. Seine Aufgabe ist nicht nur die dynamischen und statischen Daten eines Simulationsmodells auszulesen und an das MAS weiterzuleiten sondern auch die neu eingehenden Daten des MAS an die richtigen Verkehrsobjekte zu leiten. Der Zugriff des MASC auf ein laufendes Simulationsmodell erfolgt durch ein C oder ein Python-Programm, welches die bestimmten vordefinierten Funktionen des AIMSUN implementiert. Wegen persönlicher Präferenz implementiere ich den MASC mit der Python-Programmiersprache. Die Struktur des MASC besteht aus drei kleinen Komponenten: Process Controller (PC), Data Reader/Writer (DRW), Data Converter (DC), die in der Abbildung 4.2 gezeigt werden.

4.2.1 Simulationsprozess kontrollieren

Der PC ist die zentrale Komponente des MASC. Er kann durch den Aufruf des MASS-Dienstes mit dem MAS kommunizieren. In jedem Simulationsschritt werden die folgenden Funktionen des PC von dem AIMSUN -Simulator aufgerufen.

[1] Die relevanten Informationen über ASM können in [41, 42] gefunden werden

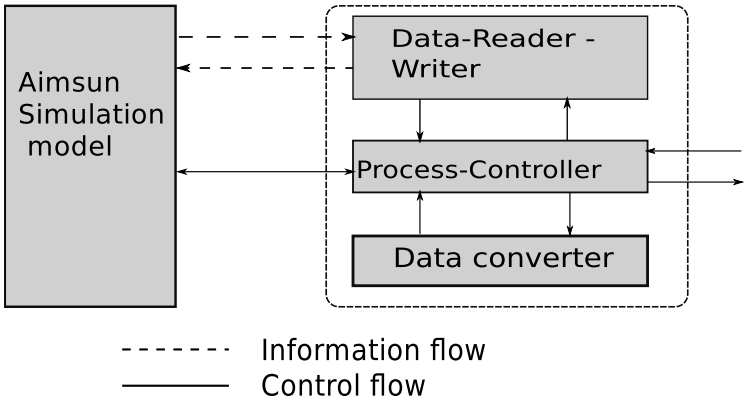


Abbildung 4.2: Die Struktur des MASC

- `AAPIInit()`: Diese Methode wird beim Starten der Simulation aufgerufen. Beim Aufruf dieser Methode werden die statischen Informationen eines Verkehrsszenarios (z. B. Informationen einer Straße, einer Kreuzung) gelesen und an das MAS weitergeleitet.

- `AAPIManager(...)`: Bei Aufruf dieser Funktion werden die folgenden Aktionen durchgeführt. Wenn ein neues Fahrzeug vom AIMSUN generiert wird, wird ein entsprechender Agent erzeugt. Diese Funktion sammelt alle dynamischen Informationen der Verkehrsobjekte und leitet diese Informationen an das MAS weiter. Beim Schicken der `step` Nachricht an das MAS, fordert sie die Agenten auf ihre Steuerbefehle an ihre Fahrzeuge zu geben.

- `AAPIExitVehicle(...)`: Diese Funktion wird aufgerufen wenn ein Fahrzeug sein Ziel erreicht, das dann vom Simulationsszenario gelöscht wird. Der Aufruf der Funktion löst den entsprechenden Agenten vom MAS.

Ein neuer Simulationsschritt wird nur weiter durchgeführt, wenn alle obigen Funktionen erfolgreich durchgeführt würden.

4.2.2 Dynamische und statische Daten der Verkehrsobjekte

Während der Simulation werden die dynamischen und statischen Informationen der Verkehrsobjekte für die Agenten des MAS benötigt. Das Auslesen der dynamischen und statischen Informationen eines Verkehrsobjektes erfolgt mit Hilfe der DRW-Komponente. AIMSUN bietet verschiedene Funktionen (siehe [41]) um die bestimmten Daten eines Verkehrsobjektes auszulesen. Die DRW-Komponente kombiniert diese Funktionen und bietet die geeignete Schnittstelle für den PC um auf ein bestimmtes Verkehrsobjekt zuzugreifen.

Der Austausch der Informationen zwischen dem MASC und dem MAS erfolgt durch den Aufruf eines entsprechenden CORBA-Dienstes. Dies erfordert die Darstellung der dynamischen und statischen Informationen als CORBA-Objekte. Die Aufgabe des DC

(Daten-Konverters) besteht darin die dynamischen und statischen Informationen in geeignete CORBA-Objekte zu konvertieren. Der DC konvertiert auch die Antworten (Kontrollbefehle von Fahrzeugen) des MAS in geeignete Informationen für die simulierten Verkehrsobjekte. Im folgenden beschreibe ich die Interaktionen zwischen PC, DRW und DC durch ein Beispiel des Ablaufes eines Simulationsschrittes.

4.2.3 Beispielablauf eines Simulationsschrittes

Der AIMSUN -Simulator startet einen neuen Simulationsschritt beim Schicken der Nachricht `AAPIManager` an den PC. Bei Erhalt der `AAPIManager` Nachricht des AIMSUN -Simulators

```
01: AIMSUN-Simulator------AAPIManager(...)------->PC;
02: PC--------readVehDynInfo(...)---------------->DRW;
03: DRW---reply readVehDynInfo(...)-------------->PC;
04: PC--------convertToCORBAObjecte(...)--------->DC;
05: DC----reply convertToCORBAObjecte(...)------->PC;
06: PC--------updateVehicleInfor(...)------------>MAS;
07: MAS---reply updateVehicleInfor(...)---------->PC
08: PC--------step(...)------------------------->MAS;
09: MAS---reply step(...)----------------------->PC;
10: PC--------extractControllCommand(...)-------->DC;
11: DC----reply extractControllCommand(...)------>PC;
12: PC--------setControllCommand(...)------------>DRW;
13: DRW---reply setControllCommand(...)---------->PC;
14: PC----reply AAPIManager(...)----------------->AIMSUN-Simulator;
```

Abbildung 4.3: Interaktionen zwischen PC, DRW und DC

führt der PC einen Simulationsschritt durch. Zunächst schickt der PC die `readVehDynInfor`-Nachricht an den DRW um die neuen Informationen aller Fahrzeuge auszulesen. Der DRW fragt AIMSUN nach den benötigen Informationen und antwortet auf die `readVehDynInfor` Nachricht des PC. Nach dem Erhalt der Antwort des DRW, sendet der PC diese an den DC um die Informationen von Verkehrsobjekten in die CORBA-Objekte zu konvertieren. Die konvertierten Informationsobjekte werden von dem PC in die `updateVehicleInfor (Objecte)`- Nachricht eingefügt und an den MASS geschickt. Nachdem alle Agenten ihre neuen Informationen bekommen haben, schickt der PC eine `step()` Nachricht an den MASS und wartet auf die Fahrzeug-Kontrollbefehle der Agenten. Der PC benutzt den DRW um die Kontrollbefehle des MASS (neue Geschwindigkeiten, und Fahrrichtungen der Fahrzeuge) an die entsprechenden Fahrzeuge weiterzuleiten. Nachdem alle Fahrzeuge die neuen Daten (Geschwindigkeit und Spur) bekommen haben, beantwortet der PC die `AAPIManager(...)` Nachricht um anzuzeigen dass er für den nächsten Simulationsschritt bereit ist. Die Abbildung 4.3 zeigt die Interaktionen zwischen den Komponenten: PC, DRW und DC.

4.3 MAS-Service

Die MASS-Komponente ist für die Kommunikationen zwischen dem MASC und dem MAS zuständig. Wie im Abschnitt 4.2 beschrieben, wurde der MASC mit der Programmiersprache „Python“ implementiert. Jedoch nutze ich in dieser Arbeit die Programmiersprache „Java“ für die Entwicklung des MAS. Die MASS-Komponente soll deswegen so konstruiert werden dass alle externen Programme unabhängig von Programmiersprachen mit dem MAS kommunizieren können.

Eine bekannte Methode ist ein Middleware für die Konstruktion des MASS zu nutzen. Ein Middleware ist eine verteilte Softwareebene, die zwischen der Applikation-Ebene und der Network-Ebene sitzt [32]. Sie erlaubt unterschiedlichen Softwarekomponenten, die mit verschiedenen Programmiersprachen implementiert werden und auf verschiedenen Betriebssystemen laufen, miteinander zu kommunizieren. In dieser Arbeit benutze ich die CORBA-Technologie als Middleware für die Konstruktion des MASS. Die Abkürzung CORBA steht für den englischen Begriff „Common Object Request Broker Architecture and Specification“. CORBA ist eine Standardarchitektur. Sie erlaubt den Softwarekomponenten durch CORBA-Dienste miteinander zu kommunizieren [2]. Die Kommunikation zwischen dem MASC und dem MASS erfolgt durch den Austauschen von CORBA-Objekten. Beim Aufruf eines geeigneten CORBA-Dienstes von MASS kann der MASC die Informationen von den Verkehrsobjekten an den Agenten im MAS schicken. MASS bietet die folgenden CORBA-Dienste:

- Initialisierung eines Simulationsmodells: Die Konstruktion eines Agenten fordert, dass der Agent seine Umgebung erkennen kann. Mit diesem Dienst werden alle statische Informationen eines Simulationsmodells an dem MAS transferiert. Die aktuelle Version des ATSim kann nur die Straßeninformationen (zB. Anzahl der Spuren, die Breite und Länge einer Straße) an dem MAS schicken.

- Erzeugen eines neuen Agenten: Während der Simulation werden die Fahrzeuge von AIMSUN dynamisch erzeugt. Durch den Aufruf dieses Dienstes wird ein Agent erzeugt, der die Steuerung des neu in die Simulation eingeführten Fahrzeuges übernimmt.

- Entfernen eines Agenten: Die Entfernung eines Agenten aus dem MAS wird durchgeführt wenn das entsprechende Fahrzeug sein Ziel erreicht hat und aus der Simulation entfernt wird.

- Aktualisierung der dynamischen Informationen eines Agenten: Nachdem ein Agent eine Aktion durchgeführt hat, muss er die Informationen seines Fahrzeuges aktualisieren. Beim Aufruf dieses Dienstes werden die neuen Informationen eines Fahrzeuges an den entsprechenden Agenten weitergeleitet.

- Vermitteln eines neuen Simulationsschrittes: Beim Aufrufen dieser Dienste werden alle Agenten aufgefordert ihre Aktionen durchzuführen.

[2] Weitere Informationen über CORBA kann unter http://www.omg.org/oma/ gefunden werden

Die aktuellen MASS-Dienste wurden entwickelt zur Simulation von kleinen Verkehrsszenarien mit Fahrzeugen und Straßen. Für die komplexeren Szenarios mit mehreren Verkehrsobjekten (Kreuzungen, Fußgänger, Verkehrsampel) sollen weitere Dienste eingefügt werden. Die MASS verarbeitet nicht alle Anfragen von externen Programmen sondern leitet die Anfragen an den MAS-Manager(MASM) weiter.

4.4 MAS-Manager

Der MAS-Manager(MASM) ist die Hauptkomponente des MAS. Er verarbeitet die Anfragen der externen Programme und verwaltet die Verkehrsobjekte. Der MASM besteht aus drei kleinen Komponenten: Simulation Controller(SC), Traffic Object Container(TOC), Jade Agent Container(JAC). Die Abbildung 4.4 stellt die Struktur des MASM dar.

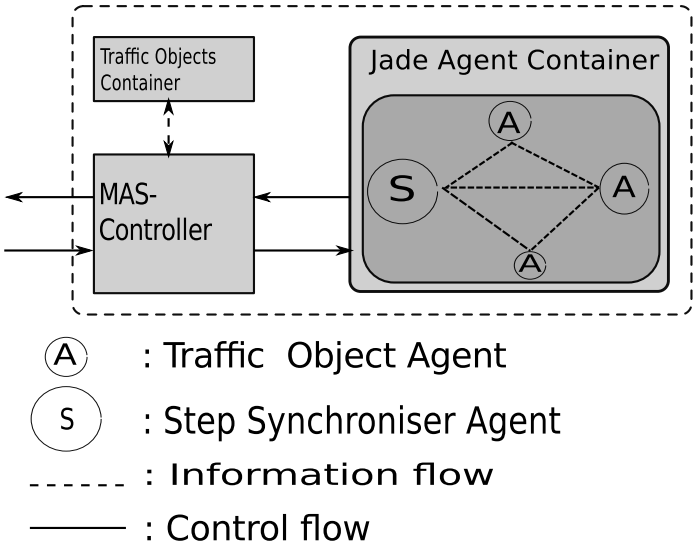


Abbildung 4.4: Die Struktur des MASM

4.4.1 TOC-Verkehrsobjektcontainer

TOC ist ein Zwischenspeicher des MAS. Darin werden die statischen und dynamischen Informationsobjekte gespeichert. Die Existenz des TOC erfüllt zwei Aufgaben:

Die erste Aufgabe besteht darin die Menge der Daten, die zwischen MAS und AIMSUN ausgetauscht werden, zu reduzieren. Viele Verkehrsobjekte verfügen über zwei Arten von Informationen: statische und dynamische Informationen. Die dynamischen Informationen sind die veränderbaren Eigenschaften von Verkehrsobjekten und müssen nach jedem Simulationsschritt aktualisiert werden. Die statischen Informationen sind die unveränderbaren Eigenschaften der Verkehrsobjekte (z.B. Länge von Fahrzeugen). Es ist nicht nötig diese Eigenschaften nach jedem Simulationsschritt zu aktualisieren.

Das Speichern der statischen Informationen in TOC erlaubt dem MAS diese Informationen bei Bedarf abzurufen ohne mit AIMSUN zu kommunizieren.

Die zweite Aufgabe des TOC ist eine Schnittstelle für externe Programme zur Überwachung des MAS bereitzustellen. Zu diesem Zweck speichert TOC auch die dynamischen Informationen der Verkehrsobjekte. Das MAS sollte die Möglichkeit bieten alle Informationen der Verkehrsobjekte anzuzeigen (z. B. zum Debuggen). Ein externes Programm (z. B. Visualizer) kann auf die Informationen des TOC zugreifen und diese verarbeiten. Es ist zu beachten, dass ein externes Programm nur die Kopien der Verkehrsobjektinformationen des TOC erhält. Diese Beschränkung verhindert die Veränderung der Informationen der Verkehrsobjekte durch externe Programme.

4.4.2 JAC-Agentencontainer

JAC ist ein Container zum Abspeichern der Agenten. Wie ich im Abschnitt 2.4 vorgestellt habe, ist Jade ein mächtiges Framework für die Konstruktion eines MAS. Jade bietet einem Java-Programm die Möglichkeit einen Agentencontainer zu erzeugen. JAC ist ein Jade-Container. Ein im JAC gespeicherter Agent ist eine Erweiterung des Jade-Agenten. Jeder Agent existiert als ein Prozess und kann mit anderen Agenten kommunizieren. Ich unterscheide zwei Arten von Agenten. Die TOA (Traffic Object Agent) Agenten sind die Repräsentanten der Verkehrsobjekte des AIMSUN . Es ist zu beachten, dass nur die Verkehrsobjekte, die ihre Zustände verändern können, als TOA-Agenten im MAS betrachtet werden. Zum Beispiel, können Verkehrsampel und Fahrzeuge als TOC-Agenten dargestellt werden aber Kreuzungen und Straßen nicht. Im Unterschied zu den TOA Agenten, ist ein SS (Step Synchroniser)-Agent kein Repräsentanten eines Verkehrsobjektes des AIMSUN . Der AIMSUN -Simulator erfordert, dass in einem Simulationsschritt alle Fahrzeuge ihre Aktionen durchführen müssen. Die Rolle des SS-Agenten ist es, die TOA-Agenten zu informieren wann diese ihre Aktionen für den nächsten Simulationsschritt durchführen müssen. Die Aktionen der TOA Agenten werden an den SS-Agenten geschickt.

4.4.3 Simulation Kontroller

Der Simulation Kontroller (SC) verwaltet den Agentenlebenszyklus und den Ablauf der Simulation. Er verarbeitet die Anfragen von MASC und gibt die Antworten an MASC zurück. In dieser Arbeit verfügt SC über die folgenden Funktionen:

Einen Agenten erzeugen und entfernen: Das Erzeugen eines neuen Agenten erfolgt durch Hinzufügen eines neuen Agenten im JAC. Wenn ein neuer Agent erzeugt wird, dann wird dessen Referenz auch gleich im TOC abgespeichert. Die Referenz wird aus dem TOC entfernt, wenn der entsprechende Agent aus dem JAC entfernt wird.

Den sichtbaren Bereich eines Agenten beschränken: Wie bereits in Abschnitt 3.2 erläutert, kann ein Fahrzeug nur mit anderen Fahrzeugen innerhalb eines bestimmten

Radius kommunizieren. Somit können zwei Agenten nur miteinander kommunizieren, wenn sie sich Sichtweite voneinander befinden. In diesem Fall übermittelt der MASC die statischen und dynamischen Informationen zwischen den beiden Agenten. Diese Methode erlaubt den Agenten Informationen von anderen Agenten zu bekommen ohne mit ihnen direkt zu kommunizieren. Die im TOC gespeicherte Referenz eines Agenten wird von SC benutzt um die Umgebungsinformationen wie Straßen, Fahrspuren an jeden TOA-Agenten zu informieren.

Simulationsschritte an Agenten informieren: Beim Starten des MAS, erzeugt SC einen SS-Agenten und behält seine Referenz. SC benutzt den SS-Agenten um die Simulationsschritte des AIMSUN an den TOC-Agenten zu übermitteln. Ich teile einen Simulationsschritt in drei Phasen auf. In der ersten Phase, befiehlt der SC den SS-Agenten die „preStep"-Nachricht an alle TOC-Agenten zu schicken. Die zwei anderen Nachrichten „step" und „postStep" werden an die TOC-Agenten in der zweiten Phase bzw. dritten Phase geschickt. Wie in Abschnitt 4.4.2 beschrieben, schicken die TOC-Agenten ihre Aktionen an den SS-Agenten. Diese Aktionen werden von SC an MASC weitergeleitet.

5 Agentenstruktur

5.1 Einleitung

Im Kapitel 3 wurde die Theorie der gruppenorienterten Fahrmethode vorgestellt. Die Konstruktion des TOA-Agenten [1] hat die folgenden Anforderungen der vorgestellten Fahrmethode zu erfüllen.

Planung: Die Planungsfähigkeit konzentriert sich auf den Aufbau von Aktionen für die langfristigen Ziele. Ein Agent kann beispielsweise keinen MLC-Spurwechsel wegen der Kollisionsgefahr mit anderen Agenten durchführen. Er versucht einen Plan für seine künftige Aktionen aufzustellen, der ihm erlaubt zur gewünschten Spur zu wechseln. Der Agent sollte in der Lage sein nicht nur einen lokalen Plan sondern auch einen kooperativen Plan aufzustellen. Ein kooperativer Plan enthält mehrere Agenten als Teilnehmer.

Verhandlungen: Die Verhandlungsfähigkeit des TOA-Agenten ist Voraussetzung für die Vereinbarung eines kooperativen Plans. Ein Agent bekommt an einem Zeitpunkt eine Einladung an einem kooperativen Plan teilzunehmen. Die Verhandlungsfähigkeit eines Agenten besteht nicht nur darin die Einladung zu akzeptieren oder abzulehnen, sondern auch darin einen andren Vorschlag für einen Plan zu machen. In dieser Arbeit kann ein TOA-Agent nur die Entscheidung treffen ob er die Einladung akzeptiert oder nicht.

Kooperation: Kooperation ist die Fähigkeit des TOA-Agenten mit anderen Agenten einen gemeinsamen Plan durchzuführen. Wenn es nötig ist, muss der TOA-Agent bereit sein auf sein persönliches Ziel zu verzichten.

Im Abschnitt 2.3 wurden drei Agentenarchitekturen vorgestellt. Es wurde gezeigt, dass die reaktiven Agenten sich sehr flexibel an die Änderungen der Umgebung anpassen. Jedoch kann ein reaktiver Agent keine geplante Aktion durchführen. Dagegen führt ein deliberativer Agent seine Aktion geplant und zielorientiert durch. Jedoch fehlt dem deliberativen Agenten die Verhandlungsfähigkeit. Ein Agent mit Schichtenarchitektur, wie zum Beispiel InteRRaP -Agent, erfüllt alle vorstehende Forderungen. Basierend auf der InteRRaP -Architektur von Müller [35, 34] konstruiere ich den TOA-Agenten mit drei verschiedenen Schichten. Jede Schicht wird als eine Komponente des Agenten

[1] Wie ich im vorherigen Abschnitt beschrieben habe, stand der Begriff TOA-Agent für „Traffic Object Agent“ (alle Verkehrsobjekte, die ihre Zustände ändern können, sind TOA-Agent). Weil in dieser Arbeit nur die „Fahrzeug“ Verkehrsobjekte ihre Zustände verändern und von Agenten gesteuert werden können, werden TOA-Agenten als autonome Fahrzeuge verstanden

dargestellt. Der Rest dieses Kapitels baut sich wie folgt auf. Im Abschnitt 5.2 beschreibe ich einen Überblick über die Struktur des TOA-Agenten und die Darstellung der Schichten. Die Funktionen und die Kommunikationen zwischen Schichten des TOA-Agenten werden in den folgenden Abschnitten dargestellt. Der Abschnitt 5.3 beschreibt die Schnittstelle für die Kommunikationen zwischen internen Komponenten des Agenten und der Außenwelt. Der Abschnitt 5.4 beschreibt das Verhalten des Agenten und die Auswahl des geeigneten Verhaltens um ein bestimmtes Ziel zu erreichen. Der Abschnitt 5.5 beschreibt die Planung des Agenten um ein bestimmtes Problem zu lösen. Dadurch erreicht der Agent sein Ziel.

5.2 Generelle Struktur

Die Abbildung 5.1 stellt eine generelle Struktur eines TOA-Agenten dar. Es ist erkennbar, dass die Struktur aus drei Ebenen besteht: „Word Interface (WI)“,„Behaviours Control Component (BCC)“, „Plan Control Component (PCC)“. Jede Ebene übernimmt eine bestimmte Rolle und verfügt über eine eigene Wissensbasis. Die Aufgabe der Wissens-

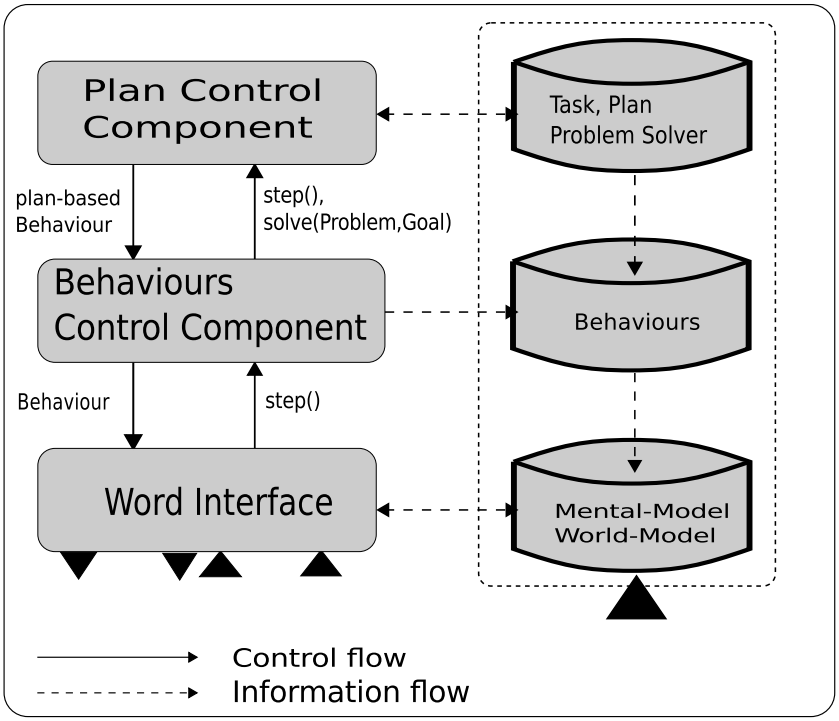


Abbildung 5.1: Generelle Struktur eines TOA-Agenten

basis ist das Wissen von Agenten abzuspeichern. In Abhängigkeit von der Ebene wird das Wissen eines Agenten in verschiedenen Strukturen abspeichert. Die Organisation der Strukturen kann mit der Darstellung von Müller [35] et al. verglichen werden. Die Objekte der oberen Ebene werden von den Objekten der niedrigeren Ebene konstru-

iert. Zum Beispiel werden die statischen und dynamischen Verkehrsobjekte in einem Welt-Modell der niedrigsten Ebene abgespeichert. Ein Verhalten (z. B. `DriveToTheLeft`) der BCC-Ebene nimmt die dynamischen Objekte (die anderen Agenten auf der linken Spur) und die statischen Objekte (die linke Spur der Straße) der WI-Ebene als Bestandteile seiner Struktur auf. Ein Plan der PCC-Ebene wird wieder vom Verhalten der BCC-Ebene konstruiert.

Wie in der InterRap-Struktur, spielt auch der Kontrollfluss eine wichtige Rolle bei der Verteilung des Ausführungsrechtes an die Komponenten. Wenn eine Komponente nicht in der Lager ist ihr Ziel zu erreichen, dann wird die Kontrolle an die obere Komponente übergeben. Zum Beispiel kann ein Agent durch ein primitives Verhalten der BBC-Ebene seine Spur nicht wechseln, dann wird das Kontrollrecht an die PCC-Ebene weitergeleitet um ein Plan-basiertes Verhalten zu finden, welches ihm erlaubt seine Spur künftig wechseln zu können.

5.3 Die Weltschnittstelle-Komponente

Die WI-Komponente ist eine Schnittstelle des TOA-Agenten. Sie erlaubt den Agenten mit der Außenwelt zu interagieren. Im wirklichen Fahrzeug-Agenten, ist die WI-Komponente eine Zusammenfassung aller Elemente (z.B. Kamera, Sensor, Wifi), die ein Agent braucht, um die Umgebung wahrzunehmen, zu beeinflussen und mit anderen Agenten zu kommunizieren. In dieser Arbeit, übernimmt die WI-Komponente die Rolle als Nachrichtenempfänger und Nachrichtensender. Ich unterscheide zwei Typen von Nachrichten. Der erste Nachrichtentyp umfasst die Simulationsnachrichten, die ein Agent von dem SS-Agenten (siehe Abschnitt 4.4.2) bekommt. Wie im Abschnitt 4.4.3 beschrieben, werden die TOA-Agenten im JAC-Container abgespeichert. Das MAS behält die Referenzen aller TOA-Agenten und benutzt diese Referenzen um die Umgebungsinformationen an Agenten zu aktualisieren. Die Aufgabe der Simulationsnachrichten besteht darin die Agenten darüber zu informieren wann der Aktualisierungsprozess fertig ist. Es gibt nur drei Simulationsnachrichten: `Pre-Step`, `Post-Step`, `Step`. Der zweite Nachrichtentyp umfasst die Anfrage-Nachrichten, die ein TOA-Agent von anderen TOA-Agenten bekommt. Jeder Nachrichtentyp wird an ein bestimmtes Modul geleitet. Die Abbildung 5.2 bezeichnet drei Module: „Action Executor (AE)“, „Message Interceptor (MI)“, „Belief Updater (BU)“ mit den dazugehörigen Nachrichtentypen.

Das BU-Modul ist für den Empfang der Simulationsnachrichten `Pre-Step` und `Post-Step` zuständig. Die Simulationsnachricht `Pre-Step` informiert einen Agenten über die Notwendigkeit seine Ansicht aktualisieren zu müssen. Die aktualisierten Informationen sind während der Simulation von dem Agenten selbst generierte Informationen. Ein Beispiel ist die Aktualisierung von Gruppeninformationen. Die Gruppeninformationen werden vom Agenten selbst erzeugt, wenn er an einer Gruppe teilnimmt. Nach der Durchführung eines Simulationsschrittes wird die Position des Agenten geändert. Wenn der Agent zu weit weg von seiner Gruppe liegt, muss er von seiner Gruppe getrennt werden. Mit die Aktualisierung der Gruppeninformationen kann der Agent

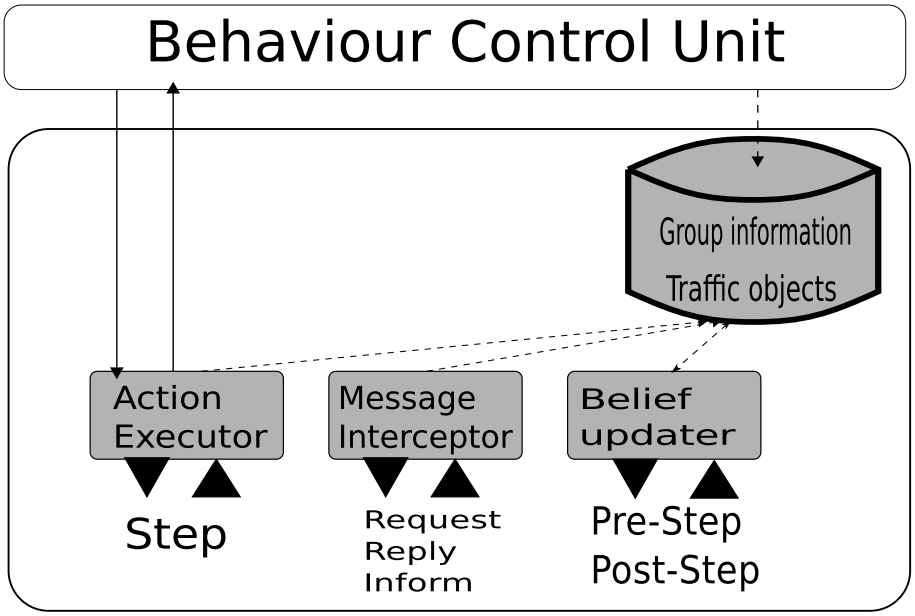


Abbildung 5.2: Schnittstelle zwischen Außenwelt und dem Agenten

sicherstellen, dass er noch zu seiner Gruppe gehört.

Die Simulationsnachricht `Post-Step` informiert einen Agenten dass seine Aktion erfolgreich durchgeführt wurde. Beim Empfang dieser Nachricht löscht der Agent alle zwischengespeicherten Informationen, die für interne Kommunikation zwischen Komponenten gebraucht werden. Es ist zu beachten, dass diese Nachricht keine Änderung der Ansicht des Agenten bewirkt.

Die Simulationsnachricht `Step` wird an das AE-Modul geleitet. Beim Empfang dieser Nachricht, muss der Agent eine Aktion durchführen. Die Aufgabe des AE-Modules besteht die `Step`-Nachricht weiter an die BCC-Komponente leiten und die Aktion der BCC-Komponente an dem SS-Agenten zurückzuschicken.

Das MI-Modul ist für die Kommunikation zwischen TOA-Agenten zuständig. Ich benutze drei Typen von Kommunikationsnachrichten zwischen Agenten: `Request`, `Reply`, `Information`. In dieser Arbeit sind die Inhalte dieser Nachrichtentypen die Java-Objekte. Beim Empfang einer `Request` Nachricht, verarbeitet MI diese Nachricht und schickt eine entsprechende `Reply` zurück. Die `Information`-Nachricht wird ebenfalls vom MI empfangen aber es wird keine `Reply`-Nachricht für diesen Nachrichtentyp an dem Sender zurückgeschickt.

In der Rolle des Nachrichtenabsenders ist WI für die Lieferung der Antwort der Simulationsnachricht an den SS-Agenten zuständig.

5.4 Die Verhaltenskontrolle-Komponente

Die BCC-Komponente bekommt eine `step` Nachricht von WI, die sie auffordert ein Verhalten auszuführen. BCC ist eine Komposition von zwei Subkomponenten: „Behaviour Control Unit (BCU)“, „Behaviour Activator (BA)“ und „Behaviour Chooser (BC)“. Die Abbildung 5.3 zeigt die Struktur der BBC. Es ist erkennbar, dass die `step`-

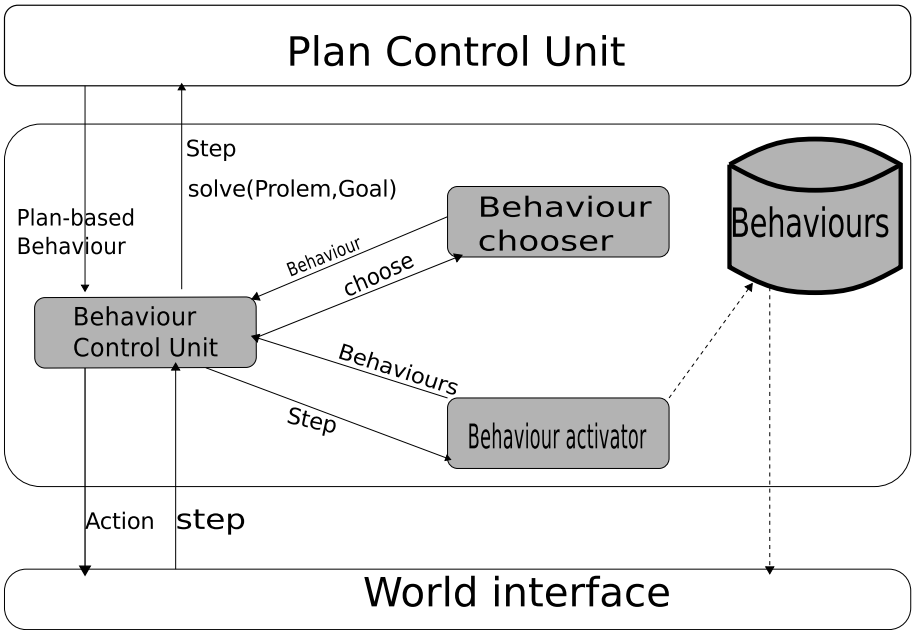


Abbildung 5.3: Verhalten Kontroller

Nachricht der WI-Ebene direkt an die BCU-Komponente geschickt wird. Die `step`-Nachricht wird an die BA-Komponente weitergeleitet. Die Aufgabe des BA besteht darin, die geeigneten Verhaltensweise, die in der Wissensbasis von Agenten gespeichert werden, zu aktivieren. Nachdem eine Menge von Verhaltensweisen aktiviert wurde, wird ein geeignetes Verhalten von BC gewählt an die WI-Ebene geschickt. Die Aktivierung einer Verhaltensweise versucht ein bestimmtes Ziel des Agenten zu erreichen. Bevor ich den Algorithmus zur Verhaltensaktivierung vorstelle, beschriebe ich die Ziele und die primitiven Verhaltensweisen des Agenten.

5.4.1 Ziele des TOA-Agenten

Die Ziele des Agenten werden in drei Typen unterschieden.

Persönliches Ziel: Wie im ersten Kapitel erwähnt, möchte jedes Fahrzeug eine gewünschte Geschwindigkeit fahren. Der Agent benutzt diese Geschwindigkeit als sein persönliches Ziel. Bei der Verfolgung seines persönlichen Zieles wählt der Agent eine Spur so, dass es ihm einen maximalen Nutzwert bringt.

Gruppenziel: Als ein Mitglied einer Gruppe, bekommt ein Agent in jedem Simulationsschritt eine Aufgabe seines Gruppenführers. Diese Aufgabe dient zur Koordination des Agenten um einen potentiellen Gruppenkonflikt zu vermeiden. In Abhängigkeit vom aktuellen Zustand des Agenten, entscheidet der Agent ob er die Aufgabe durchführen muss. Wenn der Agent diese Aufgabe akzeptiert, extrahiert er ein Gruppenziel aus dieser Aufgabe. Zur konkreten Beschreibung verwende ich das folgende Beispiel. Angenommen, es existieren zwei Fahrzeug-

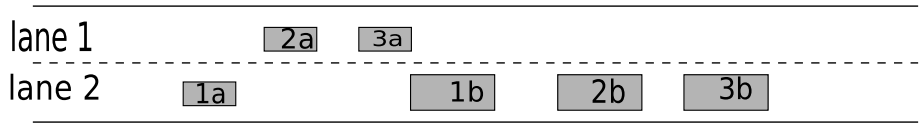


Abbildung 5.4: Konflikte Fahrzeuggruppen

gruppen $G^{2a} = 1a, 2a, 3a$ und $G^{2b} = 1b, 2b, 3b$ in einem Konflikt $Conf(G^{2a}, G^{2b})$. Die Aufgabe, die der Agent $1a$ von seinem Gruppenführer $2a$ bekommt, kann wie folgt aussehen:

```
Aufgabe:
     Erste Konfliktagenten-> [1b]
     Letzte Konfliktagenten-> [3b]
     Gruppenspuren-> [1]
```

Die Aufgabe enthält zwei Informationen. Die erste Information ist der Konflikt, den der Agent $1a$ haben könnte. Im Beispiel wird der Konflikt als eine Strecke zwischen den Agenten $1b$ und $3b$ dargestellt. Anhand dieser Information weiß der Agent $1a$, dass zwischen Agent $1b$ und Agent $3b$ die Agenten mit langsamen Geschwindigkeiten fahren. Die zweite Information sind die Gruppenspuren. Wie ich im vorstehenden Kapitel beschrieben habe, dienen die Gruppenspuren der globalen Koordination aller Mitglieder einer Gruppe. Wenn der Agent $1a$ nicht von langsamen Agenten $1b, 2b, 3b$ blockiert werden möchte, versucht er auf der Gruppenspur 1 zu fahren. In diesem Fall, hat der Agent $1a$ die Aufgabe vom Gruppenführer zur Spur 1 zu wechseln.

Gemeinsames Ziel: Ein kooperativer Plan definiert die Aufgaben für jeden Teilnehmer. Beim Akzeptieren eines kooperativen Planes, ist der Agent verpflichtet seine Aufgabe zu erfüllen. Aus dieser Aufgabe extrahiert der Agent ein Ziel.

5.4.2 Verhalten

In ihrer Arbeit [35] beschrieben Müller et al. das Verhalten eines Agenten durch den Begriff „Pattern of Behaviour (PoB)“. Ein PoB beschreibt ein kleines Stück des prozeduralen Wissens eines Agenten und kann vom Agenten selbst oder unter bestimmten Situationen aktiviert werden. Wie ich im Abschnitt 3.5 vorgestellt habe, sind das Folge-Verhalten und das Spurwechsel-Verhalten zwei grundlegende Verhaltensweisen eines

Verkehrsteilnehmers. Mit Hilfe des Folgemodells und des Spurwechselmodells können die beiden Verhaltensweisen des Agenten konstruiert werden. Im Allgemeinen definiere ich vier grundlegende Verhaltensweisen `KeepOnCurrentLane`, `FollowVehicle`, `DriveToTheLeft`, `DriveToTheRight` für einen Agenten. Die Verhaltensweise werden in einer Verhaltens-Wissensbasis des Agenten gespeichert. Es hängt vom aktuellen Zustand des Agenten ab, ob die entsprechenden Verhaltensweisen aktiviert werden. Die Abbildung 5.5 zeigt die grobe Struktur der Verhaltensweisen. Es ist erkennbar, dass alle

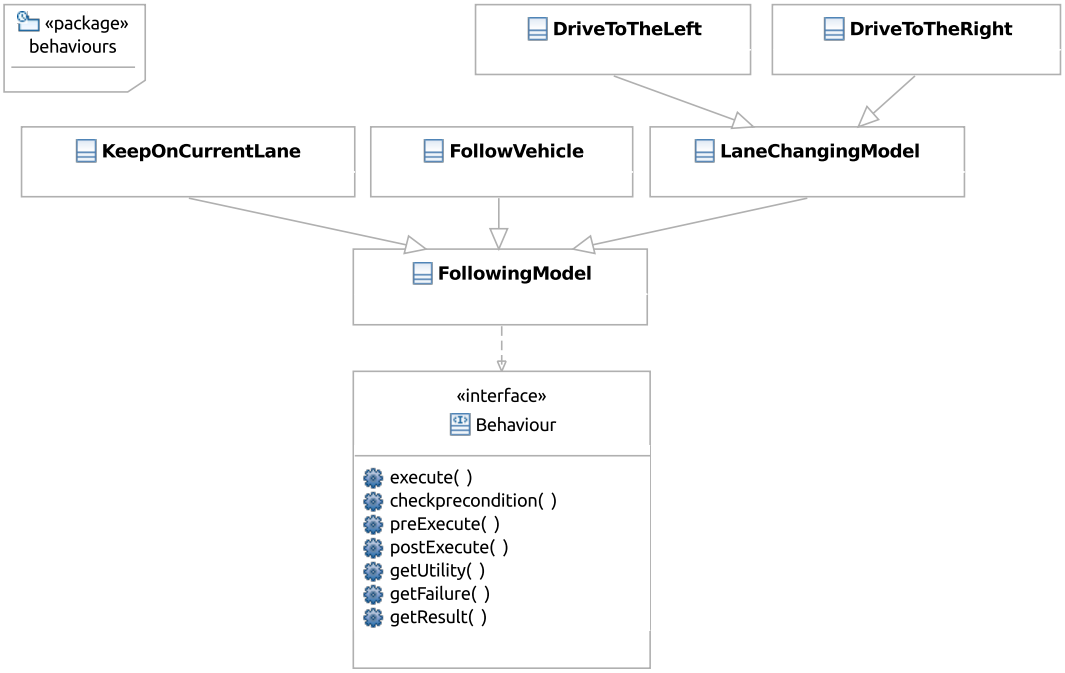


Abbildung 5.5: Verhalten

Verhaltensweisen die folgenden Funktionen implementieren:

1. Vorbedingungen (`checkpreconditon`): Ein Verhalten wird definiert um nur in bestimmtem Situationen durchgeführt zu werden. Die Funktion `checkpreconditon` prüft ob die aktuellen Umgebungsbedingungen und der mentale Agentenzustand das Verhalten auszuführen erlauben.

2. Vor der Durchführung (`preExecute()`): Diese Funktion definiert alle Aktionen, die ein Agent durchführen muss, bevor sein Verhalten ausgeführt wird.

3. Nach der Durchführung (`postExecute()`): Diese Funktion definiert alle Aktionen, die ein Agent durchführen muss, nachdem sein Verhalten durchgeführt wurde.

4. Misserfolg (`getFailure()`): Falls die Funktion `checkpreconditon` einen Fehler entdeckt, der das Ausführen des Verhalten verhindert, kann der Fehler durch die Funktion `getFailure()` identifiziert werden.

5. Ausführen (`execute`): Diese Funktion führt das Verhalten aus. Wenn ein Agent diese Funktion einer Verhaltensweise durchführen darf, gibt die Funktion `getFailure` immer den Wert `null` zurück.

6. Nutzwert (`getUtility`): Diese Funktion definiert den Nutzwert einer Verhaltensweise, den ein Agent bekommt, falls er die Verhaltensweise ausführt. Es ist zu beachten, dass ein Nutzwert nur berechnet wird, wenn die Verhaltensweise ausführbar ist. Diese Funktion implementiert die Gleichung 3.24 des Abschnittes 3.5.3.

7. Ergebnis (`getResult`): Diese Funktion gibt das Ergebnis einer Verhaltensweise zurück. Die Ergebnisse aller Verhaltensweise haben die folgende Form:

```
Result:
 Geschwindigkeit-> n    ; n ist eine positive Gleitkommazahl
 Richtung-> i           ; i ist eine Zahl der Menge [-1,0,1].
                        ; Wobei die Zahle -1,0,1 stehen für
                        ; die linke, gerade bzw rechte Richtungen.
```

Im Vergleich mit der Struktur des PoB von Müller et al. [35] definiere ich keine Abbruchbedingung für eine Verhaltensweise. Dies liegt an der Tatsache, dass die Verhaltensweise so definiert wird , dass sie innerhalb der Dauer des Simulationsschrittes liegt. Eine Verhaltensweise wird terminiert wenn ihre Vorbedingungen nicht erfüllt werden. Zum Beispiel hat ein Agent A die Aufgabe einen anderen Agenten B auf einer bestimmten Spur n im Abstand von drei Sekunden zu folgen. Der Agent A kann wegen der Technologie immer nur nach einer Sekunde die Informationen der Umgebung aktualisieren. Um die Aufgabe durchzuführen, plant er die drei Verhaltensweisen `FollowVehicle` nacheinander auszuführen.

Angenommen, nach der Ausführung der zweiten Verhaltensweise zeigen die aktuellen Informationen des Agenten A an, dass der Agent B seine Spur gewechselt hat, dann terminiert die nächste geplante Verhaltensweise wegen der nicht erfüllten Vorbedingung „der Agent B musst auf der definierten Spur n sein".

Verhaltensweise `KeepOnCurrentLane`

Die Verhaltensweise `KeepOnCurrentLane` entspricht der meist benutzten Verhaltensweise eines Agenten. Ein spezielles Merkmal dieser Verhaltensweise ist, dass sie immer ausführbar ist. Sie hat keine Vorbedingungen und benutzt das Folgemodell im Abschnitt 3.14 für die Berechnung der nächsten Geschwindigkeit des Agenten. Diese Verhaltensweise erfordert keinen Spurwechsel.

Verhaltensweise `FollowVehicle`

Die Verhaltensweise `FollowVehicle` berechnet die Geschwindigkeit eines Agenten für den Fall, dass er einen bestimmten Agent folgen muss. Im Unterschied zu der `KeepOnCurrentLane` Verhaltensweise muss der verfolgte Agent nicht unbedingt auf derselben

Spur des verfolgenden Agenten fahren. Diese Verhaltensweise implementiert die Funktion 3.20 des Abschnittes 3.5.2.

Verhaltensweise `DriveToTheLeft` und `DriveToTheRight`

Die beiden Verhaltensweisen `DriveToTheLeft` und `DriveToTheRight` implementieren das Spurwechselmodell, welches im Abschnitt 3.5.2 beschrieben wurde. Die Vorbedingungen implementieren die mathematischen Bedingungen 3.16 3.15,3.18,3.19 des Abschnittes 3.5.2

5.4.3 Verhaltensaktivierung

Im Abschnitt 3.5.3 habe ich die vier Zustände: `Grouping`, `Forming`, `Overtaking`, `Free Drive` eines Agenten definiert. Anhand des aktuellen Zustandes benutzt BA den folgenden Algorithmus um die Verhaltensweise zu aktivieren. Es ist zu beachten, dass die

Algorithm 3 Verhaltensaktivierung

```
Verhaltensliste={ } ; eine Liste von aktivierten Verhaltensweisen
if Zustand ist Gruppierung oder Loslösen then
  Verhaltensweisenliste={ DriveToTheLeft, DriveToTheRight, KeepOnCurrentLane }
else if Zustand ist Gefrieren then
  Verhaltensweisenliste={ KeepOnCurrentLane }
  if die linke Spur ist eine der Gruppenspuren then
    addiert DriveToTheLeft zur Verhaltensweisenliste
  end if
  if die rechte Spur ist eine der Gruppenspuren then
    addiert DriveToTheRight zur Verhaltensweisenliste
  end if
else
  if die linke Spur ist selbst eine oder befindet sich näher an einer der Gruppenspuren then
    addiert DriveToTheLeft zur Verhaltensweisenliste
  end if
  if die rechte Spur ist selbst eine oder befindet sich näher an einer der Gruppenspuren then
    addiert DriveToTheRight zur Verhaltensweisenliste
  end if
end if
```

aktivierten Verhaltensweisen nicht unbedingt ausführbar sind. Sie werden an die BC-Komponente weitergeleitet. Die Rolle der BC-Komponente besteht darin eine Verhaltensweise aus der Verhaltensliste zu wählen und an die AC-Komponente der WI-Ebene zu schicken.

Die vom BA aktivierten Verhaltensweisen werden als *reaktive Verhaltensweisen* bezeichnet (um sich von den *Plan-basierten Verhaltensweise* der PCC-Ebene zu unterscheiden). Die Bezeichnung *reaktive Verhaltensweise* geht darauf zurück, dass die Verhaltensweise nur im Fall eines kurzfristigen Zieles des nächsten Simulationsschrittes aktiviert wird.

Dagegen wird eine *Plan-basierte Verhaltensweise* im Fall eines langfristigen Zieles aktiviert. Befindet sich ein Agent beispielsweise im Zustand `Bildung`, dann wird der Agent aufgefordert zu einer seiner Gruppenspuren zu wechseln. Das Gruppenziel *"Zu Gruppenspur wechseln„* wird zunächst als kurzfristiges Ziel betrachtet. Der Agent aktiviert die Verhaltensweise `DriveToTheLeft` (falls eine der Gruppenspuren die linke Spur ist) um das Gruppenziel zu erreichen. Wenn die Verhaltensweise `DriveToTheLeft` vom BC ausgewählt wird, dann erreicht der Agent sein Ziel.

5.4.4 Auswahl einer Verhaltensweise

Die BC-Komponente bekommt die Verhaltensweisen vom BCU und von der PCC-Ebene übermittelt. Die Aufgabe der BC-Komponente besteht darin eine Verhaltensweise auszuwählen und deren Durchführbarkeit zu testen. Wenn die Verhaltensweise durchführbar ist, dann werden die Informationen der Verhaltensweise (neue Geschwindigkeit und neue Spur) an den AE der WI-Ebene geschickt. Die Auswahl einer geeigneten Verhaltensweise basiert auf dem aktuellen Ziel des Agenten. Ich definiere eine einfache Auswahlstrategie durch zwei Regeln:

1. Wenn der Agent einen Plan verfolgt und eine reaktive ausführbare Verhaltensweise existiert, die dem Agenten erlaubt seinen Plan zu terminieren, dann wählt er die reaktive Verhaltensweise. Existiert jedoch keine reaktive Verhaltensweise, dann wählt der Agent eine Plan-basierte Verhaltensweise.

2. Wenn der Agent keinen Plan zu verfolgen hat, dann wählt er eine ausführbare Verhaltensweise mit maximalem Nutzwert.

Die erste Regel nutzt die Vorteile der Schichtenarchitektur um einen Plan schnell zu terminieren. Weil die Umgebung dynamisch ist, kann eine Plan-basierte Verhaltensweise manchmal ineffizient sein. Die Möglichkeit eine der beiden Verhaltensweisen *reaktive Verhaltensweise* oder *Plan-basierten Verhalten* zu wählen, erlaubt dem Agenten seinen Plan flexibler zu terminieren. Dadurch kann er sein Ziel schneller erreichen. Die zweite Regel erlaubt dem Agenten eine Verhaltensweise auszuwählen, die sich am besten für ihn und für seine benachbarten Agenten eignet (siehe Abschnitt 3.5.3). Es kann vorkommen, dass der Agent keine ausführbare Verhaltensweise finden kann.

Diese Situation tritt auf, wenn keine der aktivierten Verhaltensweisen die Vorbedingungen erfüllt. Sollte eine Bedingung nicht erfüllt sein, dann wird ein *Problem* identifiziert. In diesem Fall hat der Agent zwei Auswahlmöglichkeiten. Entweder aktiviert er die Verhaltensweise `KeepOnCurrentLane` oder er versucht einen Plan aufzustellen um das Problem zu lösen.

Der Agent meiner Arbeit wird einen Plan nur dann erstellen, wenn die Vorbedingungen der `DriveToTheLeft-` oder `DriveToTheRight`-Verhaltensweise ein Problem verursacht und der Agent im `Forming`-Zustand ist. Das heißt, der Agent versucht einen MLC-Spurwechsel durchzuführen. Die Planung eines Spurwechsels, die das Hauptziel der `DriveToTheLeft` und `DriveToTheRight`-Verhaltensweisen ist, beschreibe ich im Abschnitt 5.5 konkreter.

5.5 Die Plan-Kontrolle-Komponente

Wenn ein TOC-Agent sein Ziel nicht mit einem reaktiven Verhalten erreichen kann, plant er eine Reihe von Verhalten um das Ziel zu erreichen. Die PCC-Komponente ist für die Planung zuständig. Die Struktur der PCC-Komponente umfasst drei Subkomponenten: Plan Control Uni (PCU), Plan Monitor (PM), Plan Extractor (PE), Problem Solver Activator (PSA). Die Abbildung 5.6 zeigt die Struktur der PCC-Komponente.

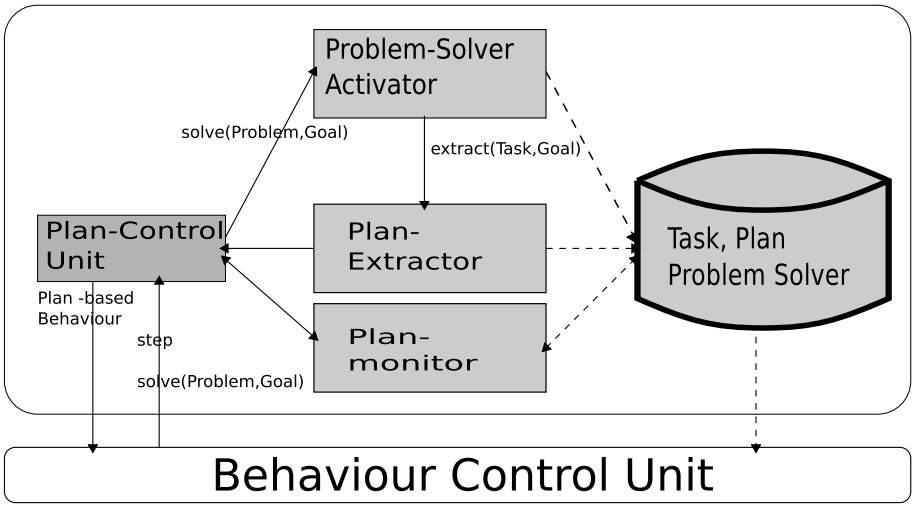


Abbildung 5.6: Plan Kontroler

5.5.1 Die PCU-Komponente als Koordinator der Planung

PCU ist die zentrale Komponente von PCC. Sie empfängt verschiedene Nachrichten von der BCC-Ebene und leitet sie zu anderen Komponenten weiter. Die aktuelle Implementierung von PCU erlaubt ihr zwei Nachrichten verarbeiten:

- Die `solve(Problem,Goal)`-Nachricht: Diese Nachricht erhält die Informationen über das zu erreichende Ziel und das Problem. Beim Empfang dieser Nachricht wird die PCU aufgefordert einen Plan zu erstellen um das Ziel zu erreichen. Die Nachricht wird weiter an die PSA-Komponente geleitet. Wenn ein Plan erfolgreich konstruiert wird, schickt die PCU ein Plan-basiertes Verhalten an die BCC-Ebene zurück.

- Die `step`-Nachricht: Dieser Nachricht bekommt die PCU-Komponente während der Ausführung eines Planes. Beim Empfang dieser Nachricht benutzt PCU die PM-Komponente um zu wissen ob der Agent seinen Plan noch durchführen sollte. Wenn PM eine positive Antwort zurückgibt, nimmt PCU ein Plan-basiertes Verhalten aus seinem Plan und leitet es an die BCC-Ebene weiter.

5.5.2 Die PSA-Komponente als Planer

Der Planungsprozess wird von der PSA-Komponente durchgeführt. PSA bekommt die Probleme und Ziele von der PCU. Der Agent verfügt über einer vordefinierte Bibliothek von „Problem Solver"-Modulen. Jedes „Problem Solver"-Modul ist für ein Paar [Problem, Ziel] zuständig. In Abhängigkeit von den Problemen und Zielen wird ein entsprechendes „Problem Solver"-Modul aktiviert. Ein „Problem Solver"-Modul kann einen kooperativen Plan (einen Plan mit vielen Teilnehmern) oder einen persönlichen Plan (der Plan hat nur den Agenten selbst als Teilnehmer) erstellen. Im Unterschied zu der InterRap-Struktur [35] integriere ich die Fähigkeit der kooperativen Planung und der persönlichen Planung in einem einzelnen „Problem Solver"-Modul. Dies liegt an der Tatsache, dass in vielen Verkehrsszenarios der Agent durch die beiden kooperativen und persönlichen Pläne sein Ziel erreichen kann. Somit existieren für den Agenten die Auswahlmöglichkeiten kooperativer Plan und persönlicher Plan. Die Integration der kooperativen Planung und persönlichen Planung in einem Modul erlaubt dem Agenten die beiden Plan-Typen miteinander zu vergleichen. Dadurch kann er einen besseren Plan auswählen.

Die kooperative Planung erfordert Interaktionen zwischen Teilnehmern zur Einigung auf einen gemeinsamen Plan. FIPA spezifizierte einen Standardprotokoll für die Interaktionen zwischen Agenten unter den Namen „Contract Net Protocol (CNP)" [37]. Das CNP-Protokoll eignet sich besonders gut für komplexe Interaktionen. Wegen der Einfachheit der Interaktion meiner Arbeit, benutze ich die folgende Form:

```
Vc--contract.-->Va ;Vc schickt eine Vertrag-Anfrage an Va
Va--yes or no-->Vc ;Va schickt die Antwort an Vc zurück
```

Die obige Interaktion erlaubt die Vereinbarung eines gemeinsamen Planes zwischen Agenten V_c und V_a via Vertrag. Es ist erkennbar, dass mit der obigen Form der Agent V_a keinen Vorschlag für den gemeinsamen Plan machen kann. Meine künftige Vision ist, die obige Interaktion-Form durch das CMP-Protokoll zu ersetzen, wenn die Interaktionen zwischen Agenten komplexer werden.

Struktur eines Vertrages

Die Definition eines Vertrages muss alle wichtigen Informationen für Plan-Teilnehmer enthalten. Ich definiere die folgende Struktur für einen Vertrag:

```
Contract:
 Goal;
```

```
Task of participant;
Condition of contract;
Duration time;
```

Wobei das „Goal“-Element das Ziel des Vertrages ist. Das „Task of participant“-Element ist die Aufgabe, die der Vertragsteilnehmer erfüllen muss. Das „Condition of contract“-Element umfasst alle Bedingungen, die alle Vertragsteilnehmer einhalten müssen. Das „Duration time“-Element legt die Vertragsdauer fest.

5.5.3 Die PE-Komponente

Die Aufgabe der PE-Komponente besteht darin einen Plan aus einer Aufgabe zu extrahieren. Es ist zu beachten, dass die PSA-Komponente nicht direkt einen Plan sondern eine Aufgabe für den Agenten erzeugt. Das heißt, wenn das zu lösende Problem einen kooperativen Plan erfordert, dann werden verschiedene Aufgaben für jeden Teilnehmer erzeugt. PE verarbeitet diese Aufgaben und plant die nötigen Aktionen für den Agenten.

5.5.4 Die PM-Komponente als ein Monitor des Planes

Wegen des beschränkten sichtbaren Bereiches ist die Umgebung für einen Agenten nicht vollständig beobachtbar. Ein Plan wird nur mit bestimmter Erfolgswahrscheinlichkeit erstellt. Deswegen muss der Plan während der Durchführung überwacht werden. Die PM-Komponente hat zwei Aufgaben zu erfüllen:

- Die erste Aufgabe besteht darin abzuschätzen, ob die Durchführung des Planes rechtzeitig beendet werden kann. Die Abschätzung wird in jedem Simulationsschritt durchgeführt.
- Die zweite Aufgabe besteht darin die Bedingungen eines Planes zu prüfen. Wenn der Agent an einem kooperativen Plan teilnimmt, prüft PM ob alle Bedingungen des Vertrages noch gehalten wird.

Wenn die Abschätzung eines Planes oder die Prüfung eines Vertrages ein negatives Ergebnis ergibt, wird der Plan bzw. der Vertrag als nicht erfolgreich bewertet und von der Wissensbasis entfernt.

5.5.5 Beispiel: Plan für einen Spurwechsel

In diesem Abschnitt stelle ich ein Planungsbeispiel für einen Spurwechsel eines TOA-Agenten vor. Es wurde beschrieben, dass ein TOA-Agent im `Forming`-Zustand einen MLC-Spurwechsel durchführen muss. Ein Plan für den Spurwechsel ist nur nötig wenn der Agent durch einen Simulationsschritt seine gewünschte Spur nicht erreichen kann. Betrachten wir Abbildung 5.7. Angenommen der Agent V_c will zu der Spur 2 wechseln. Das Problem seines Spurwechsels besteht darin, dass es keine ausreichend große Lücke auf Spur 2 gibt. In diesem Fall führt ein Spurwechsel von V_c zur Kollisionsgefahr mit V_a

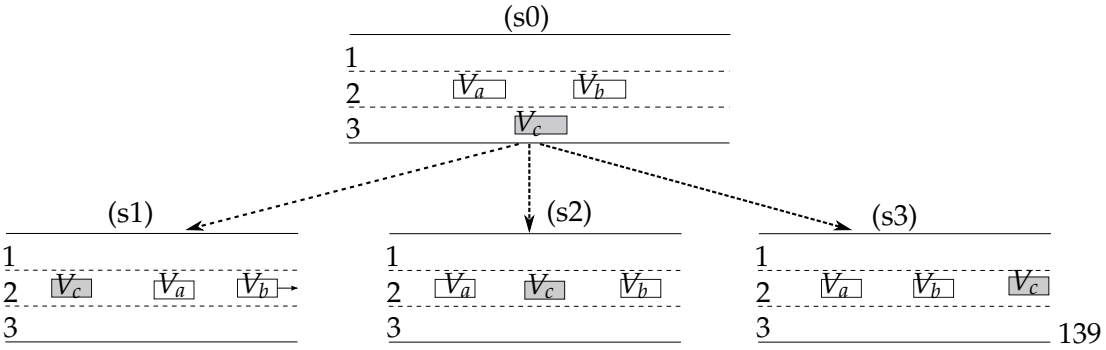


Abbildung 5.7: Drei Szenarios für einen Spurwechsel

```
Task:
 Action: Following ; Vc folgt Va
 pre-verhicle: Va  ; Va ist der Vorgänger von Vc
 duration time: 2  ; Die maximale Dauer der Aufgabe

 Action: Change lane ;Spurwechsel-Verhalten
 lane: 2             ;Zielspur
 duration time: 1    ;Dauer ist ein Simulationsschritt
```

und mit V_b. Anhand der Informationen des Problems wird PCU aufgerufen um einen Plan für den Spurwechsel aufzustellen. Die PCU benutzt PSA um das „LaneChangingSolver (LCS)"-Modul („LaneChangingSolver" ist ein Modul der „Problem Solver"-Module des Agenten und wird in der Wissensbasis abgespeichert) zu aktivieren.

Beim Planen eines Spurwechsels konstruiert das LCS-Modul drei Szenarien $s1, s2, s3$ für einen Spurwechsel (siehe Abbildung 5.7). Wenn V_c sich entscheidet das Szenario $s1$ zu realisieren, dann definiert er für sich selbst eine Aufgabe. Die Aufgabe kann wie folgt beschrieben werden: Die obige Definition der Aufgabe legt fest, dass V_c den Agenten V_a als seinen Vorgänger setzt. V_c wird V_a in zwei Simulationsschritten folgen. Nachdem diese Aufgabe erfüllt wurde, wird eine Lücke auf Spur 2 erzeugt. V_c benutzt die letzte „Change lane"-Aktion für seinen Spurwechsel. Es ist erkennbar, dass dieses Szenario keinen kooperativen Plan für den Spurwechsel erfordert. Für den Fall, dass V_c das Szenario $s2$ oder $s3$ realisieren will, braucht er die Hilfe von V_a bzw. V_b. Das bedeutet, der Agent V_a bzw. V_b muss verzögern um eine Lücke für den Spurwechsel von V_c erzeugen zu können. In diesem Fall ist ein kooperativer Plan nötig. Angenommen V_c wählt das Szenario 2, dann benötigt V_c Kooperation von V_a um eine sichere Lücke für seinen Spurwechsel zu erzeugen.

Vertrag für einen kooperativen Plan

Im Abschnitt 3.5.2 beschriebe ich die CCR-Methode für die Kooperation zwischen Fahrzeugen um das Lücke-Problem zu lösen. Das LCS-Modul implementiert diese Methode.

Laut der CCR-Methode ergeben sich folgende Verhaltensweisen von V_c und V_a:

1. V_a setzt V_c als seinen Vorgänger und folgt V_c in der Zeit $t_{V_c}^{ccr} + \alpha$.
2. V_c setzt V_b als seinen Vorgänger und folgt V_b in der Zeit $t_{V_c}^{ccr} + \alpha$.

Ein kooperativer Plan für das Szenario $s2$ wird nur aufgestellt, wenn V_a akzeptiert V_c zu folgen. Ein von LCS-Modul generierter Vertrag kann beispielsweise wie folgt aussehen:

```
Contract:
  contract ID: a_vs_b_001 ;Identifier des Vertrages
  contract creator: Vc
  Partner A: Va           ;Vertragsteilnehmer A
  Partner B: Vc           ;Vertragsteilnehmer B
  Goal:                   ;Ziel des Vertrages
    lane of partner B: 2 ;Vc fährt auf der Spur 2
  Task:                   ;Aufgabe für Partner A
    typ: Following
    pre-vehicle: Vc
    start at step: 3      ;Dauer der Aufgabe
    stop at step : 4      ;2 Simulationsschrittzeit
  Condition:              ;Bedingungen des Vertrages
    lane of A: 2          ;Partner A musst auf Spur 2 sein
    lane of B: 3
    position of A: 300.0 ;Endbedingung. Wenn A diese Position
                          ;erreicht, wird der Vertrag beendet
    position of B: 350.0
  Contract time: 2        ;maximale Dauer des Vertrages.
```

Wenn der Agent V_a den Vertrag akzeptiert, extrahiert er seine Aufgabe aus dem Vertrag und stellt einen Plan auf.

Generierung eines Planes

Nachdem V_a den Vertrag vom V_c akzeptiert, benutzt er PA um einen Plan für sich selbst zu erstellen. Der V_c generiert seinen Plan auch anhand der Aufgabe, welche sein LCS-Modul generiert. Die Abbildung 5.8 zeigt die Pläne von V_b und V_c. Laut der Pläne von V_a und V_c werden die beiden Simulationsschritte 3 und 4 eine Lücke zwischen V_a und V_b erzeugen. Der Simulationsschritt 5 des V_c erlaubt den Agenten V_c seine Spur zu wechseln.

5.5.6 Diskussion

Bis dato wurde nur die Planung eines Spurwechsels beschrieben. Das Lösung des Spurwechselproblems erlaubt dem Agenten sein Gruppenziel zu erreichen. Jedoch hat ein Agent noch sein persönliches Ziel. Es ergibt sich die Frage:

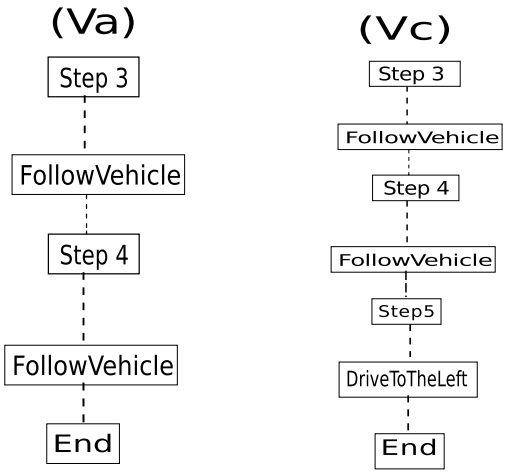


Abbildung 5.8: V_a und V_c benutzen die Pläne Va bzw. Vc für den Spurwechsel des V_c

Wie plant ein Agent um seine gewünschte Geschwindigkeit und damit sein persönliches Ziel zu erreichen?
Die Planung zum Erreichen der gewünschten Geschwindigkeit könnte in vielen Fällen unsicher und kompliziert werden. Die Unsicherheit des Planes liegt darin, dass der Agent nur einen beschränkt beobachtbaren Bereich hat. Fährt der Agent mit sehr niedriger Geschwindigkeit, braucht er einen großen Plan, der viele Simulationsschritte enthält. Dies fordert entweder einen großen beobachtbaren Bereich oder eine gute Abschätzung der künftigen Zustände der Umgebung, die im nicht beobachtbaren Bereich liegen. Je größer ein Plan ist, desto unsicherer ist es zu realisieren.

Die Leistung für die Berechnung eines großen Planes ist ebenfalls auch ein kritischer Punkt. Ein großer Plan erfordert längere Berechnungszeit und viele Leistungen des Agenten. Im Straßenverkehr muss der Agent schnell auf Änderungen der Umgebung reagieren und eine solch lange Planung ist nicht geeignet. Jedoch ist ein Plan sinnvoll wenn der Agent ihn nur in wenigen Schritten terminieren kann. Solcher Plan erfordert keine große Rechenleistung und verspricht eine bessere Realisierbarkeit. Damit solche Pläne aufgestellt werden, ist eine gewisse Beschränkung der maximalen Dauer nötig.

In dieser Arbeit, verhält sich ein Agent zielorientiert und reaktiv um sein gewünschte Geschwindigkeit zu erreichen. Er stellt keinen Plan für sein persönliches Ziel (gewünschte Geschwindigkeit zu erreichen) auf. In jedem Simulationsschritt wählt er ein Verhalten, welches ihm erlaubt seine gewünschte Geschwindigkeit bestens zu erreichen (maximaler Nutzwert). Es wurde noch keine Vergleich der Effizientz zwischen plan-basiertem Verhalten und reaktivem Verhalten für dieses Ziel durchgeführt. Der Vergleich könnte ein interessantes Thema für die künftigen Arbeiten sein.

6 Experiment

In diesem Kapitel führe ich ein Testszenario durch. Das Szenario beschreibt verschiedene Typen von Verkehrsteilnehmern. Aus unterschiedlichen Gründen haben sich die Verkehrsteilnehmer für ihre jeweilige gewünschte Geschwindigkeit entschieden. Damit die gruppenorienterte Fahrmethode mit der normalen Fahrmethode verglichen werden kann, werden die Abweichungen der wirklichen Geschwindigkeiten von den gewünschten Geschwindigkeiten der Verkehrsteilnehmer berechnet. Die Fahrmethode mit kleineren Abweichungen ist deswegen für die Verkehrsteilnehmer vorteilhaft.

6.1 Beschreibung des Szenarios

Angenommen auf einer dreispurigen Autobahn, zum Beispiel der *A*1 (siehe Abbildung 6.1) befinden sich die folgenden Typen von Verkehrsteilnehmern:

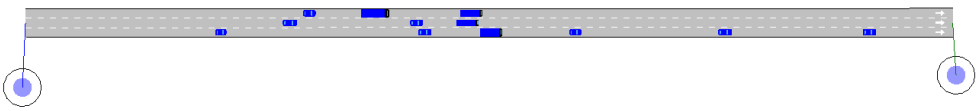


Abbildung 6.1: Ein 5 km langer Abschnitt der Autobahn *A*1

Geschäftsmann: John ist Geschäftsführer. Wegen eines Vertrages mit einem Partner muss er so schnell wie möglich zu einem verabredeten Termin fahren. John ist Vertreter einer Klasse von beschäftigten Personen. Die allgemeinen Merkmale dieser Personen können wie folgt beschrieben werden. Sie besitzen moderne, schnelle Fahrzeuge mit starken Motoren. Ihre Zeit ist wertvoll, deswegen wollen sie immer so schnell wie möglich zum Ziel fahren.

LKW-Fahrer: Tom ist Mitarbeiter eines Logistik-Unternehmens. Er arbeitet als LKW-Fahrer. Tom muss die Waren zu einem bestimmten Ziel transportieren. Wegen der Wirtschaftlichkeit wird die durchschnittliche Geschwindigkeit seines Fahrzeuges von seinem Unternehmen abgeschätzt und beschränkt, so dass er nicht zu früh, aber auch nicht zu spät zum Ziel kommt. Tom ist Vertreter einer Klasse von LKW-Fahrern. Die Personen dieser Klasse haben die folgenden Merkmale: Sie fahren große Fahrzeuge mit starken Motoren. Auf Grund der schweren Ladung können sie nicht so schnell beschleunigen. Ihre maximalen Geschwindigkeiten werden gesetzlich oder von ihrem Unternehmen beschränkt.

Rentner: Johann ist ein 70-jähriger Rentner und fährt mit seiner Frau in den Urlaub. Er besitzt ein kleines altes Auto und ist Vertreter einer Klasse von langsamen Fahrern. Die allgemeinen Merkmale dieser Klasse von Personen können wie folgt beschrieben

werden. Sie besitzen alte Fahrzeuge mit schwachen Motoren und sind nicht beschäftigt. Wegen ihres Alters und der Kapazitäten ihrer Fahrzeuge wollen sie nicht schnell fahren.

Die Eigenschaften der drei obigen Klassen von Verkehrsteilnehmern werden in meinen Tests durch die gewünschte Geschwindigkeit, maximale Verzögerung und maximale Beschleunigung dargestellt. In jedem Test werden die geeigneten Werte der Eigenschaften explizit angegebenen. In den folgenden Abschnitten werden zwei unterschiedlich konfigurierte Tests durchgeführt. Ein 5 km langer Abschnitt der Autobahn $A1$ wird mit Hilfe des AIMSUN konstruiert. Die Abbildung 6.1 zeigt einen Abschnitt der $A1$. Die Fahrzeuge der drei vorstehenden Personenklassen werden automatisch von AIMSUN generiert. In jedem Test läuft der AIMSUN -Simulator zweimal durch. Beim ersten Ablauf des AIMSUN -Simulators werden die natürlichen Verhaltensweisen der Verkehrsteilnehmer simuliert (normale Fahrmethode) und die Daten des Autobahnabschnittes $A1$ gesammelt. Beim zweiten Ablauf benutze ich das ATSim zur Steuerung der generierten Fahrzeuge (gruppenorienterte Fahrmethode). Jedes Fahrzeug wird von einem TOA-Agenten gesteuert. Die Daten des ersten Ablaufs werden mit den Daten des zweiten Ablaufs verglichen. Im folgenden beschreibe ich die von AIMSUN gesammelten Daten.

6.1.1 Beschreibung der gesammelten Daten

Die Daten werden periodisch von AIMSUN gesammelt und in einer Sqlite-Datenbank abgespeichert. Es wird definiert, dass der Zeitabstand zwischen zwei Datensätzen immer eine Minute beträgt. Die Bewertung der beiden Fahrmethoden erfolgt durch den Vergleich folgender Daten:

- **Geschwindigkeit einer Fahrzeugklasse:** Diese Geschwindigkeit ist die durchschnittliche Geschwindigkeit aller Fahrzeuge einer Klasse die ihre Ziele erreicht haben. AIMSUN [42] berechnet diese Geschwindigkeit wie folgt: Zunächst wird die durchschnittliche Geschwindigkeit S_i eines Fahrzeuges durch die Formel6.1 berechnet.

$$S_i = \frac{D_i}{TEX_i - TEN_i} \tag{6.1}$$

  Wobei D_i die Länge des simulierten Straßenabschnittes ist und $TEX_i - TEN_i$ die Zeit ist, die das Fahrzeug i braucht um D_i zu fahren. Die Einheit dieser Geschwindigkeit wird von AIMSUN in m/s berechnet. Die durchschnittliche Geschwindigkeit S_{str} der Fahrzeuge innerhalb einer Minute wird durch die Formel 6.2 berechnet

$$S_{str} = \frac{\Sigma_{i=1}^{N_{str}} S_i}{N_{str}} * 3.6 \tag{6.2}$$

  Wobei N_{str} die Anzahl der Fahrzeuge ist, die innerhalb einer Minute ihre Ziele erreicht haben. Der Wert 3.6 dient der Umwandlung der Einheit m/s in die Einheit km/h.

- **Reisezeit einer Fahrzeugklasse:** Diese Zeit ist die durchschnittliche Zeit in Sekunden, die ein Fahrzeug einer Klasse braucht um einen Kilometer zu fahren. AIMSUN berechnet die Reisezeit TT_{str} wie folgt [42]

$$TT_{str} = \frac{\Sigma_{i=1}^{N_{str}} TT_i}{N_{str}} \tag{6.3}$$

  Wobei TT_i die Zeit ist, die das Fahrzeug i braucht um einen Kilometer zu fahren. Die Reisezeit wird zur Berechnung der Verzögerungszeit eines Fahrzeuges benutzt.

- **Verzögerungszeit einer Fahrzeugklasse:** Diese Zeit ist die durchschnittliche Verzögerungszeit aller Fahrzeuge einer Klasse. Die Verzögerungszeit eines Fahrzeuges ist die Differenz zwischen der erwarteten Reisezeit (die Zeit, die ein Fahrzeug im optimalen Zustand braucht um einen Kilometer zu fahren) und der wirklichen Reisezeit. Die Werte der Verzögerungszeiten werden von AIMSUN automatisch in der Sqlite-Datenbank gespeichert.

6.2 Test mit gleichen Eigenschaften von Fahrzeugen einer Klasse

In diesem Test wird es angenommen, dass die Fahrzeuge einer Klasse die gleichen Eigenschaften: gewünschte Geschwindigkeit, maximale Beschleunigung und maximale Verzögerung haben. Die Tabelle 6.1 gibt die konfigurierten Werte der Eigenschaften an.

	Geschäftsmann	LKW-Fahrer	Rentner	Einheiten
gewünschte Geschwindigkeit	160	80	60	km/h
maximale Beschleunigung	6	4	2	m/s^2
maximale Verzögerung	-8	-8	-6	m/s^2

Tabelle 6.1: Darstellung der drei Typen von Verkehrsteilnehmern

Die Bewertungsfunktion 3.4 erfordert es, die Einteilungsfaktoren $\alpha_1, \alpha_2, \alpha_3$ und die Standardabweichungen für die Eigenschaften gewünschte Geschwindigkeit s_{ds}, maximale Beschleunigung s_{acc} und maximale Verzögerung s_{dcc} zu definieren. Weil es in diesem Test keinen Unterschied zwischen Fahrzeugen einer Klasse gibt, definiere ich die Einteilungsfaktoren und die Standardabweichungen wie folgt: Der kleine Wert 0.01

α_1	α_2	α_3	s_{ds}	s_{acc}	s_{dcc}
1	1	1	0.01 km/h	0.01 m/s^2	0.01 m/s^2

Tabelle 6.2: Werte der Parameter der Bewertungsfunktion 3.4

aller Standardabweichungen erlaubt den Fahrzeugen einer Klasse eine Gruppe zu bilden.

Abschließend muss die Methode zur Generierung von Fahrzeugen konfiguriert werden. Eine bekannte Methode zur Generierung des Zeitabstandes zwischen zwei Ereignissen ist die exponentielle Verteilungsmethode. AIMSUN bietet diese Methode zur Berechnung des Zeitabstandes zwischen zwei generierten Fahrzeugen [42]. Der Algorithmus zur Berechnung des Zeitabstandes wird wie folgt beschrieben:

```
 u ist eine zufällige Zahl zwischen 0 und 1
 λ ist die durchschnittliche Anzahl von Fahrzeugen pro Sekunde. t ist
der Zeitabstand zwischen zwei generierten Fahrzeugen
Wenn λ > 0.0 dann:
  t=(−1/λ) ∗ ln(u)
Sonst:
  t=die maximal von Computer darstellte Gleitkommazahl
```

Abbildung 6.2: Algorithmus zur Berechnung des Zeitabstandes zwischen zwei generierten Fahrzeugen [42]

Die Konfigurationen der λ-Werte jeder Fahrzeugklasse erfolgt durch die folgende Tabelle. Mit den λ-Werten der Tabelle 6.3 werden insgesamt ungefähr 2500 Fahrzeuge in einer

	Geschäftsmann	LKW-Fahrer	Rentner
λ	0.27	0.138	0.27

Tabelle 6.3: λ-Werte der drei Fahrzeuge-Klasse

Stunde generiert. Davon sind 1000 Fahrzeuge von Geschäftsmännern, 500 Fahrzeuge von LKW-Fahrern und 1000 Fahrzeuge von Rentnern.

6.2.1 Testergebnisse und Analyse

Die Abbildung 6.3 zeigt die Geschwindigkeiten aller Verkehrsteilnehmerklassen.

Der linke Chart zeichnet die Geschwindigkeitsdaten der autonomen Fahrzeuge auf. Auf den ersten Blick sieht man, dass die durchschnittlichen Geschwindigkeiten der autonomen Fahrzeugklassen deutlich getrennt werden. Die durchschnittliche Geschwindigkeit eines autonomen Fahrzeuges der Geschäftsmann-Klasse beträgt ungefähr 130 km/h (die rote Querlinie). Im Durchschnitt fährt ein LKW-Fahrer mit einer Geschwindigkeit von 77 km/h (die grüne Querlinie). Die Rentner-Fahrer sind mit ihren gewünschten Geschwindigkeiten (60 km/h) fast zufrieden.

Der rechte Chart sind die Geschwindigkeitsdaten der bemannten Fahrzeuge. Im Vergleich mit den autonomen Fahrzeugen, haben die Geschwindigkeiten der Klassen der bemannten Fahrzeuge keine klare Trennung. Bei der Geschäftsmann-Klasse beträgt die durchschnittliche Geschwindigkeit ungefähr 90 km/h. Diese Geschwindigkeit ist ungefähr 60 km/h niedriger als die durchschnittliche Geschwindigkeit der selben Klasse der autonomen Fahrzeuge (ungefähr 130 km/h). Auch bei der LKW-Fahrer-Klasse liegt die durchschnittliche Geschwindigkeit eines bemannten Fahrzeuges (ungefähr 72 km/h) 5

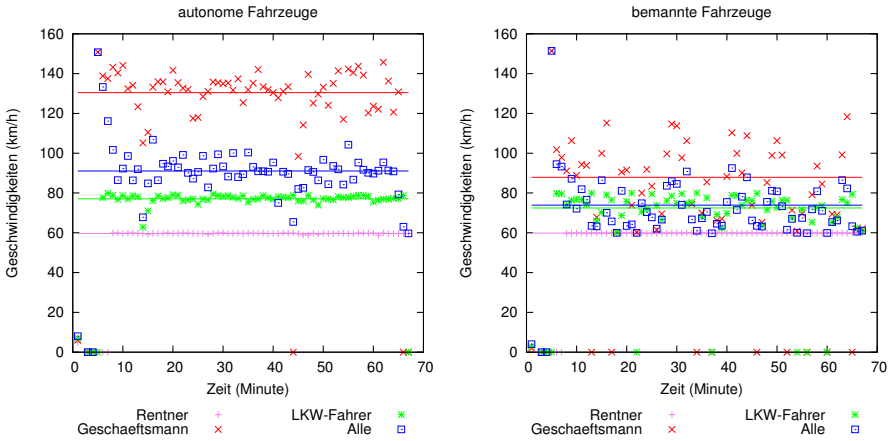


Abbildung 6.3: Diese Abbildung stellt die Geschwindigkeitsdaten aller Verkehrsteilnehmerklassen dar. Jeder Punkt ist ein Datensatz, der die durchschnittliche Geschwindigkeit der Fahrzeuge in einer Minute darstellt. Die Querlinien beschreiben die durchschnittlichen Werte aller Datensätze.

km/h unter der Durchschnittsgeschwindigkeit eines autonomen Fahrzeuges. Die Rentner-Klasse zeigt nur einen kleinen Geschwindigkeitsunterschied zwischen autonomen Fahrzeugen und bemannten Fahrzeugen.

Im Durchschnitt beträgt die Geschwindigkeit eines autonomen Fahrzeuges ungefähr 91 km/h (die blaue Durchschnittslinie der „Alle"-Datensätze). Diese Geschwindigkeit ist deutlich höher als die durchschnittliche Geschwindigkeit eines bemannten Fahrzeuges (ungefähr 74 km/h).

Die Unterschiede der Geschwindigkeiten zwischen den autonomen Fahrzeugen und den bemannten Fahrzeugen haben großen Einfluss auf die Verzögerungszeiten von Fahrzeugen. Die Abbildung 6.4 stellt die Verzögerungszeiten aller Verkehrsteilnehmer-Klassen dar. Der linke Chart stellt die Datensätze der Verzögerungszeiten der autonomen Fahrzeuge dar. Es ist sichtbar, dass die Verzögerungszeiten aller Fahrzeugklassen sehr dicht im Bereich von 0 bis 5 Sekunden/km liegen. Die Durchschnittsverzögerungszeit eines autonomen Fahrzeuges dieser Klasse (grüne Querlinie) beträgt ungefähr 1,8 Sekunden/km. Es ist bemerkenswert, dass die Verzögerungszeiten der LKW-Klasse eng aneinander im Bereich von 0,5 Sekunden/km bis 2,2 Sekunden/km liegen. Dies bedeutet, dass die Verzögerungszeit eines zufälligen autonomen LKW-Fahrzeuges maximal nur 1.5 Sekunden/km von der Durchschnittsverzögerungszeit abweicht. Aus Sicht eines Logistik-Unternehmens, welches die Verzögerungszeit der Lieferung von Produkten abschätzen muss, spielt diese Abweichung eine wichtige Rolle. Je kleiner die Abweichung ist, desto genauer ist die Abschätzung der Verzögerungszeit ihrer Produkte.

Der rechte Chart visualisiert die Daten der Verzögerungszeiten der bemannten Fahrzeuge. Es ist erkennbar, dass die Datenpunkte der LKW-Fahrer-Klasse im Bereich von 0 Sekunden/km bis 15 Sekunden/km verteilt sind. Die Durchschnittsverzögerungszeit eines Fahrzeuges dieser Klasse beträgt ungefähr 5 Sekunden/km (grüne Querlinie).

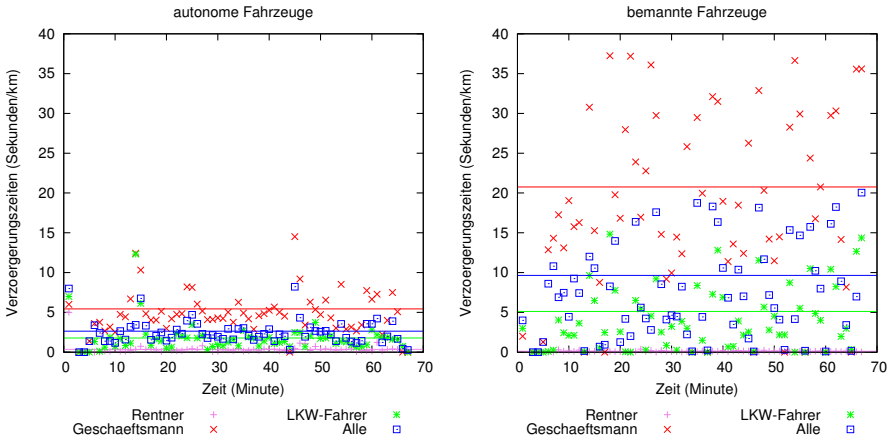


Abbildung 6.4: Diese Abbildung beschreibt die Verzögerungszeiten aller Fahrzeugklassen. Jeder Punkt ist ein Datensatz, der die durchschnittliche Verzögerungszeit einer Fahrzeugklasse in einer Minute darstellt. Die Querlinien sind die durchschnittlichen Werte aller Datensätze.

Das heißt, dass die Verzögerungszeit eines zufälligen bemannten LKW-Fahrzeuges maximal 10 Sekunden/km von der Durchschnittsverzögerungszeit abweicht. Als Konsequenz ergibt sich eine große Ungenauigkeit, wenn ein Logistik-Unternehmen die Durchschnittsverzögerungszeit als die Verzögerungszeit ihrer Lieferung berechnet.

Im Durchschnitt wird ein bemanntes Fahrzeug der Geschäftsmann-Klasse bis zu ungefähr 21 Sekunden/km (siehe die rote Querlinie) verzögert. Diese durchschnittliche Verzögerung ist viel höher als die durchschnittliche Verzögerung eines autonomen Fahrzeuges der selben Klassen (ungefähr 5.5 Sekunden/km). Bei der LKW-Fahrer-Klasse der bemannten Fahrzeuge, ist die durchschnittliche Verzögerungszeit auch ungefähr 3,2 Sekunden/km höher als die durchschnittliche Verzögerungszeit der autonomen Fahrzeuge.

Eine Besonderheit tritt bei der Rentner-Klasse auf. Die autonomen Fahrzeuge der Rentner-Klasse verzögern sich leicht (ungefähr 0,4 Sekunden pro Kilometer). Im Gegensatz dazu verzögern sich die bemannten Fahrzeuge der Rentner-Klasse fast nicht. Dies liegt daran, dass die autonomen Fahrzeuge manchmal ihre Geschwindigkeiten verzögern müssen um die kooperative Spurwechselmethode (siehe Abschnitt 3.5.2) durchzuführen. Im Gegensatz dazu müssen sich die bemannten Fahrzeuge der selben Klasse nicht verzögern. Die blaue Querlinie zeigt die durchschnittliche Verzögerungszeit eines Fahrzeuges aller Klassen. Es ist erkennbar, dass im Allgemeinen die durchschnittliche Verzögerungszeit eines autonomen Fahrzeuges 7 Sekunden/km niedriger als die durchschnittliche Verzögerungszeit eines bemannten Fahrzeuges ist.

Die Ergebnisse der Geschwindigkeiten und der Verzögerungszeiten haben gezeigt, dass die autonomen Fahrzeuge schneller zum Ziel fahren. Der Test wurde so konfiguriert, dass keine Unterschiede in den Eigenschaften zwischen Fahrzeugen der selben Klasse entstehen. Jedoch weisen im realen Verkehrszustand die Fahrzeuge einer

Gruppe immer kleine Unterschiede in ihren Eigenschaften auf. Ein solcher Verkehrszustand wird im nächsten Testszenario durchgeführt.

6.3 Test mit leichten Eigenschaftsunterschieden von Fahrzeugen einer Klasse

In diesem Test werden die Eigenschaftsunterschiede nicht nur für die Fahrzeuge der unterschiedlichen Klassen, sondern auch für die Fahrzeuge derselben Klassen konfiguriert. Jedoch sind die Eigenschaftsunterschiede der Fahrzeuge derselben Klassen signifikant kleiner als die Eigenschaftsunterschiede der Fahrzeugen der unterschiedlichen Klassen. Tabelle 6.4 gibt die konfigurierten Werte der Eigenschaften an. Die Konfigu-

	Geschäftsmann	LKW-Fahrer	Rentner	Einheiten
gewünschte Geschwindigkeit	[150, 160]	[80, 90]	[50, 60]	km/h
maximale Beschleunigung	[5.5, 6]	[3.5, 4]	[2, 2.5]	m/s^2
maximale Verzögerung	[-8, -7.5]	[-8, -7.5]	[-6, -5.5]	m/s^2

Tabelle 6.4: Darstellung der drei Typen von Verkehrsteilnehmern

rationen erlauben zwei Fahrzeugen einer Gruppe einen maximalen Unterschied in der gewünschten Geschwindigkeit 10 km/h zu haben. Die Unterschiede der maximalen Beschleunigung und der maximalen Verzögerung der Fahrzeuge der selben Klasse sind 0.5 m/s^2.

Die Parameter der Bewertungsfunktion 3.4 werden wie folgt konfiguriert:

α_1	α_2	α_3	s_{ds}	s_{acc}	s_{dcc}
2	0.4	0.6	10 km/h	0.5 m/s^2	0.5 m/s^2

Tabelle 6.5: Werte der Parameter der Bewertungsfunktion 3.4

Danach folgt die Konfiguration der Fahrzeuggenerierung. In diesem Test verwende ich wieder die exponentielle Methode zur Fahrzeuggenerierung, die im ersten Test beschrieben wurde (siehe Abbildung 6.2). Die λ-Werde der Fahrzeugklassen werden wie in Tabelle 6.6 geändert.

	Geschäftsmann	LKW-Fahrer	Rentner
λ	0.21	0.27	0.21

Tabelle 6.6: λ-Werte der drei Fahrzeugklassen

Mit den λ-Werten der Tabelle 6.6 generiert AIMSUN in einer Stunde ungefähr 750 Fahrzeuge der Geschäftsmann-Klasse, 750 Fahrzeuge der Rentner-Klasse und 1000 Fahrzeugen der LKW-Fahrer-Klasse. Im nächsten Abschnitt werden die Ergebnisse dieses Tests vorgestellt.

6.3.1 Ergebnisse und Analyse

Abbildung 6.5 zeigt die durchschnittliche Geschwindigkeit der autonomen Fahrzeuge und der bemannten Fahrzeuge. Wie in Abbildung 6.5 dargestellt, gibt es bei der LKW-

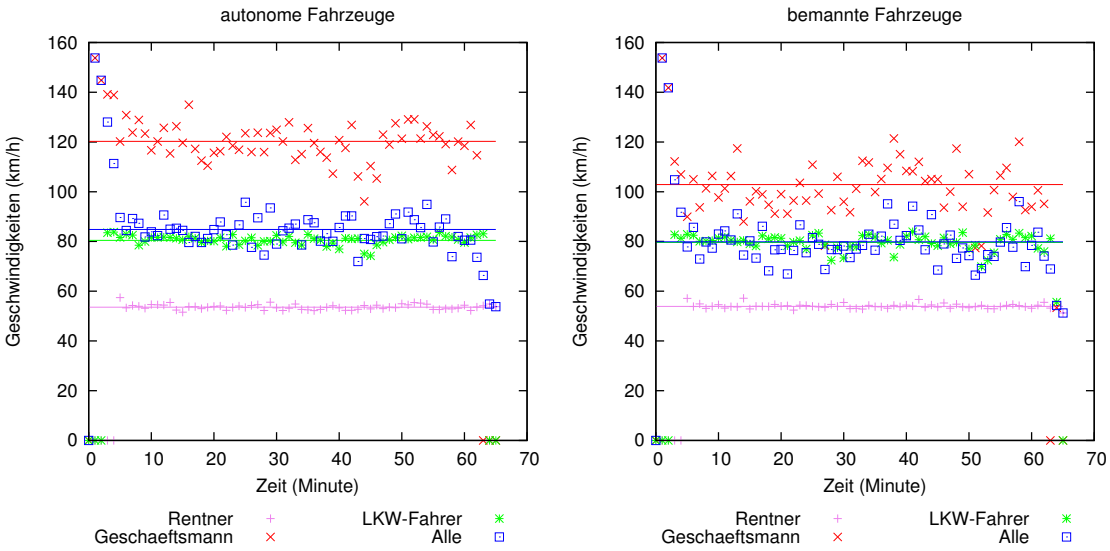


Abbildung 6.5: Geschwindigkeiten aller Fahrzeugklassen des zweiten Tests

Fahrer-Klasse nur einen kleinen Geschwindigkeitsunterschied. Die Durchschnittsgeschwindigkeiten der autonomen Fahrzeuge und der bemannten Fahrzeuge betragen ungefähr 80,5 km/h bzw. 79.6 km/h. Der Grund für diesen kleinen Geschwindigkeitsunterschied liegt darin, dass die LKW-Fahrer-Klasse nur von der Rentner-Klasse blockiert wird. Im Vergleich mit den generierten Fahrzeugen der Rentner-Klasse des ersten Tests werden weniger Fahrzeuge der Rentner-Klasse dieses Tests generiert (siehe λ-Werte aus Tabelle 6.6). Als Konsequenz werden die LKW-Fahrer-Fahrzeuge seltener von Fahrzeugen der Rentner-Klasse blockiert. Aufgrund der hohen Anzahl der generierten LKW-Fahrzeuge blockiert die LKW-Fahrer-Klasse die Geschäftsmann-Klasse stark. Es ist erkennbar, dass bei der Geschäftsmann-Klasse die Geschwindigkeit der autonomen Fahrzeuge (120 km/h) höher als die Geschwindigkeit der bemannten Fahrzeuge (102 km/h) ist. Der Unterschied beträgt ungefähr 18 km/h. Dieser Unterschiedsbetrag ist kleiner gegenüber dem aus dem ersten Test. Die Rentner-Klassen der autonomen Fahrzeuge und der bemannten Fahrzeuge zeigen einen geringen Geschwindigkeitsunterschied. Im Durchschnitt fahren ein autonomes Fahrzeug und ein bemanntes Fahrzeug der Rentner-Klasse mit der Geschwindigkeit 53,5 km/h bzw. 53,8 km/h.

Die blauen Querlinien der beiden Charts zeigen die Durchschnittsgeschwindigkeit aller Fahrzeugklassen. Im Allgemeinen fährt ein autonomes Fahrzeug mit der Geschwindigkeit 84.8 km/h. Diese Geschwindigkeit ist höher als die Geschwindigkeit eines bemannten Fahrzeuges (79.5 km/h).

Abbildung 6.6 stellt die Verzögerungszeiten dar. Im Vergleich mit den bemannten

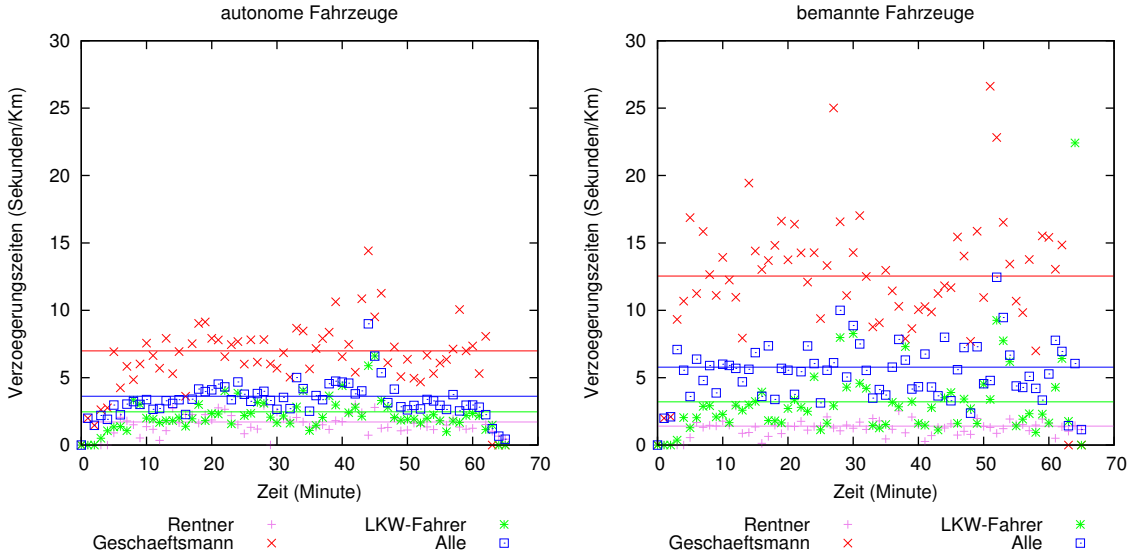


Abbildung 6.6: 326 Verzögerung

Fahrzeugen wurden die autonomen Fahrzeuge der Geschäftsmann-Klasse (rote Querlinie) weniger stark verzögert. Die Durchschnittsverzögerungszeit eines autonomen Fahrzeugs und eines bemannten Fahrzeugs der Geschäftsmann-Klasse betragen ungefähr 7 Sekunden/km bzw. 12,5 Sekunden/km. Das heißt, dass ein autonomes Fahrzeug der Geschäftsmann-Klasse ungefähr 5,6 Sekunden/km schneller als ein bemanntes Fahrzeug derselben Klasse fährt.

Bei der LKW-Fahrer-Klasse gibt es keinen großen Unterschied in der Verzögerungszeit zwischen einem autonomen Fahrzeug und einem bemannten Fahrzeug. Ein autonomes Fahrzeug und ein bemanntes Fahrzeug werden durchschnittlich ungefähr 2,5 Sekunden/km bzw. 3,3 Sekunden/km verzögert. Jedoch liegen die Verzögerungszeiten der bemannten LKW-Fahrzeuge überall in dem großen Bereich von 0,2 Sekunden/km bis 10 Sekunden/km (die grünen Datenpunkte). Im Gegensatz dazu werden die Verzögerungszeiten der autonomen Fahrzeuge dicht im Bereich von 1 Sekunden/km bis 4 Sekunden/km verteilt. Wie die Analyse des ersten Tests gezeigt hat, hat diese Verteilung der Verzögerungszeiten einen Einfluss auf die Abschätzung der allgemeinen Verzögerungszeit eines Fahrzeugs. Die Wahrscheinlichkeit, dass ein zufälliges autonomes Fahrzeug durchschnittlich 2,5 Sekunden/km verzögert wird, ist höher als die Wahrscheinlichkeit, dass ein zufälliges bemanntes Fahrzeug durchschnittlich 3,3 Sekunden/km verzögert wird.

Im Allgemeinen wird gezeigt, dass ein autonomes Fahrzeug nur ungefähr 3,6 Sekunden/km (blaue Querlinie) verzögert wird. Dagegen wird ein bemanntes Fahrzeug 5,7 Sekunden/km verzögert. Dieser Test hat gezeigt, dass obwohl es kleine Unterschiede in den Eigenschaften zwischen Fahrzeugen einer Klasse gibt, die gruppenorienterte Fahrmethode (die von autonomen Fahrzeugen durchgeführt wird) ihre Vorteile in der

Geschwindigkeit und der Verzögerungszeit gegenüber der normalen Fahrmethode hat.

6.4 Leistungstest

Das ATSim System ist eine Kombination zwischen dem AIMSUN System und dem Jade Rahmenwerk. In diesem Abschnitt führe ich einen Test zur Bewertung der Leistung und der Skalierbarkeit des Systems durch. Bevor der Test beschrieben wird, stelle ich den Simulationsprozess mit Hilfe von Ablaufdiagrammen dar. Die Abbildung 6.7 zeigt den Prozess, indem das AIMSUN die CORBA-Dienste `createVehicle()`, `updateDynamicVehInfor()`, `step()` benutzt um Fahrzeuge zu erzeugen und um aktuelle Informationen und Simulationsschrittnachrichten an laufende Agenten zu schicken. Die Ab-

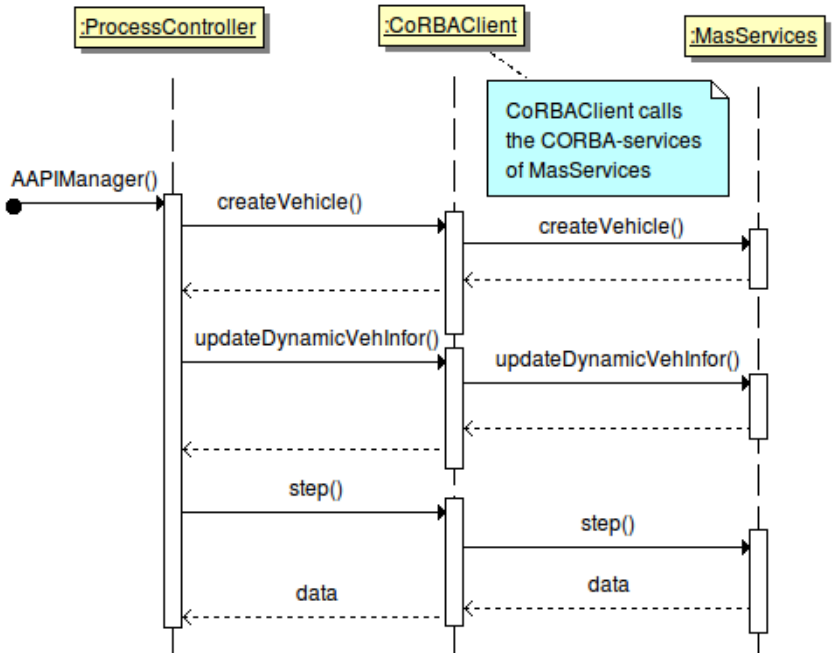


Abbildung 6.7: ATSim : Aufrufprozess des AIMSUN

bildung 6.8 beschreibt die Verarbeitung der Simulationsschrittnachricht `step()`. Es ist erkennbar, dass ein Simulationsschritt in die drei Phasen `PreStep`, `Step`, `PostStep` aufgeteilt wird. In jeder Phase wird die entsprechende Simulationsnachricht: `preStep()`, `step()` oder postStep() an die Agenten geleitet. Die Abbildung zeigt den Prozess wie die Nachricht `prestep()` an die Agenten geschickt wird. Die anderen Nachrichten `step()`, `postStep()` werden auf gleiche Weise an die Agenten geschickt. Im folgenden beschreibe ich den Aufrufprozess der Nachricht `preStep()`.

Zunächst schickt `MasServices` die Nachricht `preStep()` an den `MasController`. Der `MasController` schickt die `setNeighbourVehicles()` an alle Agenten um ihre sichtbaren Bereiche zu aktualisieren. Es ist zu beachten, dass nachdem ein Agent die `setNeig-`

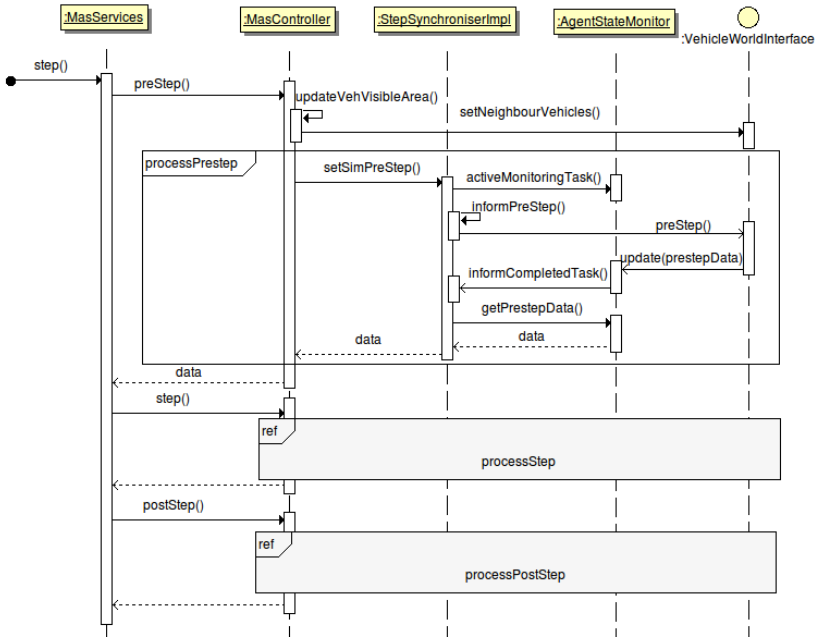


Abbildung 6.8: ATSim :Aufrufprozess des Multiagentensystems

`hbourVehicles()` Nachricht bekommt, er die anderen Agenten erkennen kann. Als nächstes aktiviert der `MasController` seinen Agentenzustandsüberwacher `AgentStateMonitor` und schickt die `setSimPreStep()` Nachricht an den `StepSynchroniserImpl` Agenten. Der `StepSynchroniserImpl` informiert alle Agenten über die PreStep-Phase beim Schicken der Nachricht `preStep()` an allen Agenten. Wenn ein Agent die `preStep()` Nachricht bekommt, führt er eine Aktion durch und aktualisiert seinen gewünschten Zustand (in Form eines Datenobjektes) an den `AgentStateMonitor`. Der `AgentStateMonitor` sammelt alle gewünschten Zustände von Agenten und schickt die `informCompletedTask()` an den `StepSynchroniserImpl` um diesen zu informieren, dass alle Agenten ihre Zustände geändert haben. Der `StepSynchroniserImpl` nimmt diese Zustände und leitet sie an den `MasController` weiter.

Zum Testen der Performance des ATSim benutze ich Dummy-Agenten. Ein Dummy-Agent führt keine rationale Aktion durch. Das heiß, er versucht nicht seine gewünschte Geschwindigkeit zu erreichen, sondern generiert eine zufällige sichere Fahraktion. Nach jedem Simulationsschritt erzeugt das AIMSUN so viel wie möglich neue Fahrzeuge. Der für den Test genutzte Computer hat die folgenden Eigenschaften:

Prozessor	Core 2 Duo Processor T5870 2.00GHz
Random-Access Memory (RAM)	2GB
Betriebssystem	Linux: Ubuntu 10.10

Die gesammelten Daten dieses Tests sind:

- Die Zeit, die das ATSim braucht um einen Simulationsschritt durchzuführen.

- Den Speicher, den das ATSim braucht um die Simulation durchzuführen.

Die Abbildung 6.9 zeigt die Ergebnisse des Tests. Die Ergebnisse haben gezeigt, dass

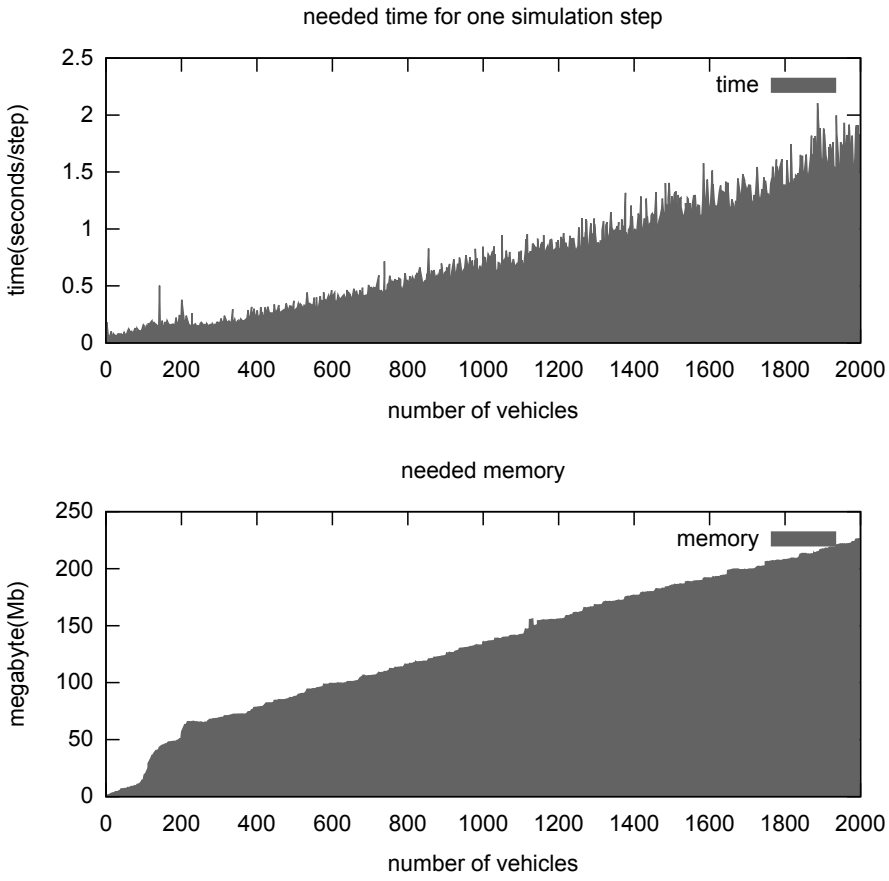


Abbildung 6.9: Leistungstest mit 2000 Fahrzeugen

mit steigender Anzahl der erzeugten Fahrzeuge das ATSim mehr Zeit für einen Simulationsschritt und mehr Speicher für die Simulation braucht. Es ist erkennbar, dass das System sehr langsam wird (ungefähr 1,8 Sekunden/Schritt), wenn die Anzahl der Fahrzeuge auf 2000 ansteigt. Anhand dieser Ergebnisse mache ich die folgende Aussage:

Das ATSim simuliert gut kleine Verkehrsszenarien mit geringer Anzahl von Verkehrsteilnehmern (ungefähr 800 Verkehrsteilnehmer). Zur Simulation eines größeren Verkehrsszenarios ist ein anderes System nötig.

6.5 Fazit

In diesem Kapitel wurden zwei Tests durchgeführt. Wie die Ergebnisse der Geschwindigkeiten und der Verzögerungszeiten jedes Tests gezeigt haben, bringt die gruppenorienterte Fahrmethode viele Vorteile gegenüber der normalen Fahrmethode. Die

gruppenorienterte Fahrmethode erlaubt den besseren Fahrzeugen (höhere gewünschte Geschwindigkeit, größere maximale Verzögerung und größere maximale Beschleunigung) schneller zum Ziel zu fahren ohne die Geschwindigkeit der anderen Fahrzeuge stark zu reduzieren. Dadurch wurden die Verzögerungszeiten verringert. Die Verzögerungszeiten der Fahrzeuge, die die gruppenorienterte Fahrmethode benutzen, können genauer abgeschätzt werden. Für Logistik-Unternehmen ist die genaue Abschätzung der durchschnittlichen Verzögerungszeit besonders wichtig. Mit dieser Abschätzung können die Unternehmen ihre Kunden über Lieferzeit von Produkten genauer informieren.

Im zweiten Test wurde gezeigt, dass die Unterschiede der Eigenschaften zwischen Fahrzeugen innerhalb einer Gruppe einen negativen Einfluss auf die Geschwindigkeiten der autonomen Fahrzeuge haben. Als Konsequenz ist es wichtig die Fahrzeuggruppen so zu bilden, dass die Unterschiede in den Eigenschaften zwischen Mitgliedsfahrzeugen so klein wie möglich werden. In den beiden Tests wurde im Allgemeinen nicht gezeigt, inwiefern die Eigenschaften der Mitgliedsfahrzeuge einer Gruppe unterschiedlich sein dürfen, so dass es weiterhin zu einer Verbesserung der Durchschnittsgeschwindigkeiten der Fahrzeuge kommt.

Die beiden Tests haben gezeigt, dass je größer die Eigenschaftsunterschiede zwischen Fahrzeuggruppen (die gewünschte Geschwindigkeit, die Verzögerung und die Beschleunigung) sind, desto bessere Geschwindigkeit- und Verzögerungszeit-Ergebnisse bringt die gruppenorienterte Fahrmethode.

7 Zusammenfassung und Ausblick

In dieser Arbeit habe ich eine neue gruppenorientierte Fahrmethode vorgestellt. Die Methode kombiniert die globale Koordination des Gruppenführers und die lokale Entscheidung der Gruppenmitglieder um eine der folgenden Verhaltensweisen `KeepOnCurrentLane`, `FollowVehicle`, `DriveToTheLeft`, `DriveToTheRight` auszuwählen. Für die Konstruktion der Fahrverhaltensweisen der autonomen Fahrzeugen wird das Folgemodell von Gipps [19] geeignet modifiziert. Die Modifikation nutzt die Vorteile der Kommunikationen zwischen Fahrzeugen um deren Geschwindigkeit zu regeln und deren Spurwechsel effizienter durchzuführen. Als Teil der gruppenorientierten Fahrmethode habe ich die CCR-Methode vorgestellt. Die CCR-Methode erlaubt verschiedenen Fahrzeugen miteinander zu kooperieren um das Lücke-Problem des Spurwechsels zu lösen.

Zur Prüfung meiner Theorie habe ich das ATSim konstruiert. Das System ist eine Kombination des AIMSUN Verkehrssimulationssystems und des JADE-Frameworks. ATSim erlaubt verschiedene Verkehrsszenarien flexibel zu konfigurieren und zu simulieren. Ich habe die autonomen Agenten zur Steuerung von Fahrzeugen des ATSim konstruiert. Die Struktur der Agenten orientiert sich stark an der InteRRaP -Struktur von Müller [35, 34]. Die Agenten wurden so konstruiert, dass sie die gruppenorientierte Fahrmethode durchführen. Die Ergebnisse und die Analyse der verschiedenen Tests haben gezeigt, dass die Anwendung der gruppenorientierten Fahrmethode die Geschwindigkeiten von Fahrzeugen erhöht und die Verzögerungszeiten von Fahrzeugen reduziert. Aus meiner Arbeit entstehen weitere Fragen, die für die zukünftige Forschung interessant sind. Im Abschnitt 3.2 habe ich die Bewertungsfunktion 3.4 zur Bildung von Fahrzeuggruppen vorgestellt. Es wurden drei statische Eigenschaften eines Fahrzeuges: gewünschte Geschwindigkeit, maximale Beschleunigung und maximale Verzögerung als Parameters der Funktion betrachtet. Die vorgestellte Bewertungsfunktion fordert, dass jedes Fahrzeug die maximalen akzeptablen Standardabweichungen für seine Eigenschaften definieren muss. Es wurde jedoch noch keine Methode beschrieben wie die akzeptablen maximalen Abweichungen automatisch von einem Fahrzeug gewählt werden sollen. Es wurde ebenfalls nicht festgelegt wie groß diese Abweichungen sein sollen, so dass die gruppenorientierte Fahrmethode noch vorteilhaft gegenüber der normalen Fahrmethode ist. Meine langfristige Vision besteht darin eine geeignete Methode zu entwickeln, die es den autonomen Fahrzeugen erlaubt, ihre akzeptablen Eigenschaftsunterschiede passend automatisch und dynamisch umzustellen. Die Bewertungsfunktion beschränkt sich bisher nur auf drei statische Eigenschaften eines Fahrzeuges. Eine zusätzliche Erweiterung dieser Bewertungsfunktion soll entwickelt werden, die die dynamischen Eigenschaften, wie zum Beispiel aktuelle Geschwindigkeit, näher betrachtet.

In dieser Arbeit wurden nur zwei Tests durchgeführt. Es gibt noch viele weitere Verkehrsszenarien, in denen die gruppenorientierte Fahrmethode getestet werden soll. Zum Beispiel ist nicht klar ob die gruppenorientierte Fahrmethode noch Vorteile gegenüber der normalen Fahrmethode besitzt, wenn die Verkehrsteilnehmer sehr dicht nebeneinander fahren.

Obwohl das ATSim erlaubt verschiedene Verkehrsszenarien zu konfigurieren und zu simulieren, fehlen noch viele Funktionen. Zum Beispiel können viele Verkehrsobjekte (wie Straßenampeln oder Kreuzungen) nicht von Agenten erkannt werden. Dies bietet einen Ansatzpunkt zur Weiterentwicklung des ATSim .

Literaturverzeichnis

[1] Philip E. Agre and David Chapman. Pengi: an implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, page 268–272, 1987. 21

[2] K. I Ahmed. *Modeling drivers' acceleration and lane changing behavior*. PhD thesis, Massachusetts Institute of Technology, 1999. 39, 41

[3] C. Astengo and R. Brena. Flock traffic navigation based on negotiation. In *ICARA. Proceedings of 3rd International Conference on Autonomous Robots and Agents, Palmerston North, New Zealand*, page 381–384, 2006. 20

[4] F. Bellifemine, A. Poggi, and G. Rimassa. Developing multi-agent systems with JADE. *Intelligent Agents VII Agent Theories Architectures and Languages*, page 42–47, 2001. 31, 32

[5] C. Bergenhem, Q. Huang, A. Benmimoun, and T. Robinson. CHALLENGES OF PLATOONING ON PUBLIC MOTORWAYS. October 2010. 18

[6] Michael E. Bratman. *Intention, Plans, and Practical Reason*. Cambridge University Press, new edition edition, May 1999. 23

[7] Rodney Brooks. Intelligence without representation. *ARTIFICIAL INTELLIGENCE*, 47:139—-159, 1991. 21, 22

[8] R. E Chandler, R. Herman, and E. W Montroll. Traffic dynamics: studies in car following. *Operations Research*, page 165–184, 1958. 34

[9] P. R Cohen and H. J Levesque. Intention is choice with commitment. *Artificial intelligence*, 42(2-3):213–261, 1990. 25

[10] R. Dowling, P.E.J. Holland, and P.E.A. Huang. California Department of Transportation Guidelines for Applying Traffic Microsimulation Modeling Software. *way*, 3:3–2, 2002. 47

[11] E. H Durfee and J. S Rosenschein. Distributed problem solving and multi-agent systems: Comparisons and examples. *Ann Arbor*, 1001:48109, 1994. 29

[12] L. C Edie. Car-following and steady-state theory for noncongested traffic. *Operations Research*, 9(1):66–76, 1961. 35

[13] Innes Andrew Ferguson. TouringMachines: an architecture for dynamic, rational, mobile agents. 1992. 26

[14] Christian Frese, Jürgen Beyerer, and Peter Zimmer. Cooperation of cars and formation of cooperative groups. In *Intelligent Vehicles Symposium, 2007 IEEE*, 2007. 20

[15] H. T Fritzsche. A model for traffic simulation. *Traffic engineering & control*, 35(5):317–321, 1994. 45

[16] D. C Gazis, R. Herman, and R. B Potts. Car-following theory of steady-state traffic flow. *Operations Research*, page 499–505, 1959. 34

[17] M. P Georgeff and A. L Lansky. Reactive reasoning and planning. In *Proceedings of the sixth national conference on artificial intelligence (AAAI-87)*, page 677–682, 1987. 24

[18] P. G. Gipps. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, 15(2):105–111, 1981. 43, 60

[19] P. G Gipps. A model for the structure of lane-changing decisions. *Transportation Research Part B: Methodological*, 20(5):403–414, 1986. 37, 42, 43, 62, 109

[20] P. G. Gipps. Multsim: a model for simulating vehicular traffic on multi-lane arterial roads. *Mathematics and Computers in Simulation*, 28(4):291–295, 1986. 42

[21] C. U Guide. FHWA. *US Department of Transportation, Washington, DC*, 2001. 38

[22] O. Gutknecht and J. Ferber. The m ad k it agent platform architecture. *Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems*, page 48–55, 2001. 9, 33

[23] Randolph Hall and Chinan Chin. Vehicle sorting for platoon formation: Impacts on highway entry and throughput. Technical report, Institute of transportation studies university of california, berkeley, March 2002. 18

[24] S. Hallé. *Automated highway systems: Platoons of vehicles viewed as a multiagent system*. Citeseer, 2005. 18

[25] P. Hidas. A functional evaluation of the AIMSUN, PARAMICS and VISSIM microsimulation models. *Road & Transport Research Journal*, 14(4):45–59, 2005. 46

[26] N. Howden, R. R\önnquist, A. Hodgson, and A. Lucas. JACK intelligent agents-summary of an agent infrastructure. In *5th International Conference on Autonomous Agents*, 2001. 32

[27] M. Larburu, J. Sanchez, and D. J Rodriguez. SAFE ROAD TRAINS FOR ENVIRONMENT: human factors' aspects in dual mode transport systems. *17th World Congress on Intelligent Transport Systems*, October 2010. 18

[28] D. H Lee and P. Chandrasekar. A framework for parallel traffic simulation using multiple instancing of a simulation program. *ITS Journal-Intelligent Transportation Systems Journal*, 7(3):279–294, 2001. 47

[29] H. Liu, W. Ma, R. Jayakrishnan, and W. Recker. Distributed modeling framework for large-scale microscopic traffic simulation. In *Proc. of 84th Annual Meeting of the Transportation Research Board (CDROM)*, 2005. 47

[30] Agent Oriented Software Pty. Ltd. Agent manual: JACK® intelligent agents agent manual. 32

[31] Agent Oriented Software Pty. Ltd. Teams manual: JACK® intelligent agents teams manual. 33

[32] Qusay H. Mahmoud. *Middleware for communications*. John Wiley and Sons, 2004. 75

[33] M. Michaelian, F. Browand, California. Dept. of Transportation, Berkeley. Institute of Transportation Studies University of California, University of Southern California. Dept. of Aerospace, Mechanical Engineering, Partners for Advanced Transit, and Highways (Calif.). *Field experiments demonstrate fuel savings for close-following*. California PATH Program, Institute of Transportation Studies, University of California at Berkeley, 2000. 17

[34] Jörg P. Müller. *The design of intelligent agents: a layered approach*. Springer, 1996. 22, 23, 79, 109

[35] Jörg P Müller and Markus Pischel. The agent architecture InteRRaP: concept and application. 1993. 9, 26, 27, 79, 80, 84, 86, 90, 109

[36] S. Panwai and H. Dia. Comparative evaluation of microscopic car-following behavior. *Intelligent Transportation Systems, IEEE Transactions on*, 6(3):314–325, 2005. 47

[37] F. I.C.N.I Protocol. Specification http://www. fipa. org/specs/fipa00030. *PC00030D. pdf*, 2000. 90

[38] PTV. VISSIM 5.30-Benutzerhandbuch, November 2010. 44

[39] Anand S Rao and Michael P Georgeff. Modeling rational agents within a BDI-Architecture. 1991. 23, 25

[40] P. Sayeg and P. Charles. Intelligent transport system. *A Sourcebook for Policy-makers in Developing Cities Module 4e, Germany*, 2005. 13

[41] TSS-Transport Simulation Systems. Aimsun 6.1 microsimulator API manual, November 2009. 37, 72, 73

[42] TSS-Transport Simulation Systems. Microsimulator and mesosimulator in aimsun 6.1 user's manual, October 2009. 60, 72, 96, 97, 98

[43] T. Toledo, H. N Koutsopoulos, and M. E Ben-Akiva. Modeling integrated lane-changing behavior. *Transportation Research Record: Journal of the Transportation Research Board*, 1857(-1):30–38, 2003. 40

[44] Robinson Tom, Chan Eric, and Coelingh Erik. Operating platoons on public motorway: An introduction to the SARTRE platooning programme. *17th World Congress on Intelligent Transport Systems*, October 2010. 18, 19

[45] P. Varaiya. Smart cars on smart roads: problems of control. *Automatic Control, IEEE Transactions on*, 38(2):195–207, 2002. 17

[46] M. Veloso and P. Stone. *Multiagent Systems: A Survey from a Machine Learning Perspective*. CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, 1997. 28

[47] Gerhard Weiss. *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT Press, 1999. 29, 30

[48] R. Wiedemann. *Simulation des stra\ssenverkehrsflusses*, volume 8. Institut f\ür Verkehrswesen der Universit\ät Karlsruhe, 1974. 44

[49] M. Wooldridge and N. R Jennings. Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(02):115–152, 1995. 20, 23

[50] Michael Wooldridge. *An Introduction to MultiAgent Systems*. Wiley, 2nd edition, July 2009. 21

[51] H. Xiao, R. Ambadipudi, J. Hourdakis, and P. Michalopoulos. Methodology for selecting microscopic simulators: Comparative evaluation of AIMSUN and VISSIM. *Methodology*, page 6, 2005. 47

[52] W. Yan-lin and W. U. Tie-jun. Car-following models of vehicular traffic. *Journal of Zhejiang University-Science A*, 3(4):412–417, 2002. 35

[53] Q. Yang and H. N Koutsopoulos. A microscopic traffic simulator for evaluation of dynamic traffic management systems. *Transportation Research Part C: Emerging Technologies*, 4(3):113–129, 1996. 38, 40, 41