Christiansen, Sander Aker
Fauske, Ivar Austin

Implementing a Virtual Driving Instructor

Master's thesis in Computer Science
Supervisor: Gundersen, Odd Erik
June 2019

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Christiansen, Sander Aker
Fauske, Ivar Austin

Implementing a Virtual Driving Instructor

Master's thesis in Computer Science
Supervisor: Gundersen, Odd Erik
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

Today, most adults understand the domain of traffic to some extent. Ways of representing this domain has been an active and popular area of research in recent years, especially focusing on traffic management and autonomous vehicles. Consequently, a good understanding of the domain and feasible ways of reasoning about traffic situations have been discovered. However, research regarding ways of evaluating a driver's performance has received less focus.

This thesis investigates the implementation of a virtual driving instructor, which frames the driver's understanding of the situation using situation awareness and provides corrective feedback on the basis of it. This novel approach to identifying driver mistakes is investigated through analysis and discussion of scenarios driven in a driving simulator and later evaluated by a proof of concept implementation. The proof of concept was implemented as a multi-agent system representing the situation in an ontology.

A professional driving instructor was consulted to provide feedback for the same scenarios as the proof of concept. A comparison between the two served as the basis for the validation of the proof of concept. Initial results were somewhat promising, with the system being able to correctly evaluate violations of having to yield or speeding. The proof of concept and the driving instructor agreed 83% of the time. However, the agreement could go as low as 42% for difficult situations. Suggested improvements are identified and future work for a full system implementation, designed for real-time evaluation and feedback, is proposed.

Keywords: *Situation Awareness, Explainability, Traffic Situations, Ontologies, Virtual Driving Instructor, Artificial Intelligence*

Sammendrag

De fleste av dagens voksne har en viss grad av forståelse for trafikkdomenet. Hvordan representere dette domenet har vært et aktivt og populært forskningsområde i mange år, spesielt innenfor trafikkhåndtering og selvkjørende biler. Som en konsekvens har god forståelse for domenet og mulige måter å resonere rundt trafikk blitt oppdaget. Derimot har forskning rundt hvordan evaluere en sjåførs prestasjoner fått mindre oppmerksomhet.

Denne oppgaven undersøker implementasjonen av en virtuell kjørelærer, som vinkler en sjåførs situasjonsforståelse ut i fra "situasjonsbevissthet" og gir konstruktiv tilbakemelding basert på denne. Denne nyskapende tilnærmingen til å identifisere sjåførfeil, undersøkes gjennom analyse og diskusjon av scenarier kjørt gjennom i en kjøresimulator som senere evalueres i en implementasjon av konseptbevis. Dette konseptbeviset ble implementert som et multi-agent system hvor situasjonen ble representert i en ontologi.

En profesjonell kjørelærer ble bedt om å gi tilbakemelding på de samme scenariene som konseptbeviset. En sammenligning mellom de to fungerte som valideringsgrunnlag for konseptbeviset. Initielle resultater var noe lovende, hvor systemet var i stand til å korrekt vurdere brudd på vikeplikt og råkjøring. Konseptbeviset og kjørelæreren var enige 83% av tiden. Enigheten kunne derimot være så lav som 42% for vanskelige situasjoner. Foreslåtte forbedringer har blitt identifisert og videre arbeid for et fullverdig system, ment til å gjøre evalueringer og gi tilbakemelding i sanntid, blir foreslått.

Nøkkelord: *situasjonsbevissthet, forklarlighet, trafikksituasjoner, ontologier, virtuell kjørelærer, kunstig intelligens*

Preface

This thesis culminates our studies at the Norwegian University of Science and Technology. It has been written to fulfill the graduation requirements for the Master of Science in Computer Science, undertaken between August 2014 and June 2019.

The project was written in the collaboration with the driving school Way AS, allowing us to work on such an interesting topic. They have accommodated very well for our needs throughout the past year, providing us with help from developers, driving instructors and general help whenever we have asked for it.

We would like to take this chance to express our sincere gratitude towards our supervisor, Odd-Erik Gundersen, having been the best supervisor we could have asked for. He took time out of his own schedule, outside of the usual hours, to help with our work and answer our questions.

Furthermore, we would to express our deep gratitude to our parents and family for always being there and supporting us, from the literal start, and up until this point in our lives.

Finally, our most profound and heartfelt thanks go to our respective partners, for their never-ending belief, support and encouragement. This accomplishment would not have been possible without them.

Thank you.

Sander Aker Christiansen & Ivar Austin Fauske

June 2019

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the thesis, by including a short section on background and motivation, listing the goals of the thesis and research questions to be answered, our method, the contributions of our work, and finally the structure of the thesis.

1.1 Background and Motivation

Although a lot of research has been done in the intersecting fields of traffic representation, autonomous vehicles and providing feedback to drivers, the literature review conducted within this thesis reveals few attempts of providing feedback to drivers on how well they are driving. Of the few approaches in the literature where driver performance is evaluated, none of these utilize situation awareness as a framework to identify why an incorrect action was taken. The novelty of such an approach in itself is a motivating factor behind the research of this thesis.
The scope of our thesis is to look at the feasibility of implementing a virtual driving instructor that uses situation awareness to indirectly identify the flaws of a driver's situation awareness. After identifying the flaws, such a system is to provide constructive, corrective feedback to resolve misinterpretations of traffic laws and facilitate improved situation awareness in future similar situations.
The work in this thesis has been done in cooperation with Way AS, a Norwegian driving school, that provided access to a realistic driving simulator, the raw output data from said simulator, as well as their source code. Way AS see the business value in having an automated assessment system that can evaluate a driver's performance. They believe such a system can support driving instructors and reduce their workload. Thus, the business opportunities of such a system are included as driving forces motivating this research.

1.2 Research Context

The research of this thesis has been conducted at the Department of Computer Science at the Norwegian University of Science and Technology, in collaboration with the driving school Way AS situated in Trondheim.
In addition to the regular services offered by a driving school, Way AS also offers students to practice in a driving simulator. Students operate the simulator from inside a regular car where inputs are applied to a car within a virtual world instead of motoring the actual car. The virtual world is projected on walls around them. Information from the driving simulator is logged to a file that can be read and analyzed later on. The data produced by the simulator will be the primary data source of our work. Figure 1.1 shows the view from within one of the simulator cars.

Figure 1.1: A picture taken from within one of Way AS' driving simulators. There is no physical engine, input actions from the car are instead applied within the virtual simulator world.

1.3 Goals and Research Questions

The underlying assumption of this report is that a system utilizing rule-based AI can be used to solve the problem of generating and providing feedback on a driver's behavior in traffic situations. The system will also take the context into account. Therefore our working hypothesis will be defined as:

"*A Virtual Driving Instructor System, using AI, can be used to evaluate a driver's behavior to give precise, justified, and understandable feedback about a traffic situation the driver can use for improvement.*".

To support this hypothesis, we suggest a set of objectives that must be satisfied, with corresponding research questions to be answered throughout this thesis. Such a hierarchy is able to shed light on the different problems such a system has to solve by imposing explicit requirements on it. Furthermore, it identifies a set of tasks that can be used to evaluate the success of the research documented within this thesis.

First of all the system has to be able to understand the situation and context a driver is operating in. Thus our first objective will be defined as: "*The system must be able to understand the situation and context.*".

A rule-based AI system reasons using rules, which represent domain knowledge. The first research question to be answered is how the system can use these rules to build a knowledge base, i.e. how to represent the situation and context of traffic, as well as traffic rules. This research question is important because good rules and a good knowledge base provide the foundation for the quality of reasoning that can be performed by the system, as well as it's ease of use.

The second research question revolves around implicit information. Much of the key information of a driving situation is not given explicitly but must be inferred. Therefore, a Virtual Driving Instructor must be able to infer implicit information. The aim of the second research question is to understand how a Virtual Driving Instructor can deduce implicit information.

After being able to represent both the traffic situation and it's context, a Virtual Driving Instructor must be able to evaluate a driver's behavior in the given situation. This will be defined as our second objective: "*A Virtual Driving Instructor must be able to evaluate driving behavior*".

To evaluate the behavior of a driver, this behavior must be represented. As a driver's behavior is a result of their actions, a way of converting actions to behavior is required. Therefore the first research question related to this objective is how driving behavior can be recognized and represented, based on driver actions.

After behavior has been represented, the next step is to evaluate it. Behavior is context-sensitive and therefore the corresponding situation, set of traffic rules and context in which the behavior was identified must be included. The second research question will thus be about how driving behavior can be evaluated with respect to a given representation of traffic rules, situation, and context.

While being able to identify flaws in driving behavior is a nice first step in the implementation of a Virtual Driving Instructor that can provide feedback, it is of little use to a driver if the driver is unable to understand the feedback. The value of constructive, understandable feedback is the rationale behind our third objective: "*Feedback generated by a Virtual Driving Instructor must be understandable for humans with little to no understanding of AI*".

The first challenge is when and how the feedback should be generated. What kind of behavior should trigger feedback? How often should one give feedback on the same thing? When do small things add up to justify generating feedback? Does a behavior result in multiple separate feedback or can it be given as joint feedback? These are all questions to look into as part of the first research question related to the third objective.

The last research question revolves around justifying the feedback. It is common to question why one was given some feedback. Most people are driving with the best intentions and commit mistakes unknowingly. Therefore it is important that the Virtual Driving Instructor is able to justify the feedback it provides. This will also ensure trustworthiness.

The research questions identified in this section will form the basis for the experiments to be performed in this thesis. The experiments will be clearly defined in chapter 7. These will be conducted within a driving simulator at the offices of Way AS. The evaluation in chapter 8 will look at each of the research questions, evaluating the success of each objective. The validity of our working hypothesis will then be looked at based on the degree of success for each of its objectives.

Below is a summary of the working hypothesis, objectives, and research questions introduced:

Hypothesis A Virtual Driving Instructor System, using AI, can be used to evaluate a driver's behavior to give precise, justified, and understandable feedback about a traffic situation the driver can use for improvement.

Objective 1 A Virtual Driving Instructor must be able to understand the situation and context.

- **RQ 1** How to represent both the situation and context that applies at a given time in traffic?
- **RQ 2** How can missing information be inferred from domain knowledge, and basic information of situation and context for a given time in traffic?

Objective 2 A Virtual Driving Instructor must be able to evaluate driving behavior.

RQ 3 Based on driver actions, can driving behavior be recognized and represented?

RQ 4 How should driving behavior be evaluated with respect to a given representation of traffic rules, situation, and context?

Objective 3 Feedback generated by a Virtual Driving Instructor must be understandable for humans with little to no understanding of AI.

RQ 5 When and how should a Virtual Driving Instructor generate feedback?

RQ 6 Given the way feedback is generated by a Virtual Driving Instructor, can it also be justified?

1.4 Research Method

Our working hypothesis states that a Virtual Driving Instructor System can evaluate driver behavior and provide feedback. This thesis is concerned with documenting how such a system would be developed, as well as the implementation and evaluation of a proof of concept. Therefore, our research method will be designing a solution and conducting experiments to validate its accuracy. Through aforementioned research method we will provide answers to our research questions.

The literature review of chapter 3 sheds light on research related to the implementation of a virtual driving instructor, where theoretical and abstract solutions have been suggested. However, there is no documented implementation of a virtual driving instructor utilizing situation awareness in the same manner as proposed in this thesis, further supporting our choice of research method. Therefore, it will be important to document the proposed system design.

Two experiments were designed to test whether or not the proof of concept was able to correctly evaluate driver performance. They consisted of a series of scenarios regarding right of way and yielding in different intersections. Our first experiment concerned a broad testing of situations. The second experiment had test subjects drive through a subset of the scenarios in a normal manner. In both experiments feedback provided by the system was compared to expected feedback. While the comparison in the first experiment was done with the authors own evaluation, the second experiment utilized the evaluation provided by a human driving instructor. A satisfactory level of accuracy would provide validity to our proposed solution, and lay the foundation for further development and improvement.

1.5 Contributions

This thesis proposes a novel way of identifying driver flaws and providing corrective feedback, through the framework of situation awareness. Under the assumption that a driver acts with the best intention, incorrect behavior is the result of incorrect situation awareness. Thus, this thesis looks into the issue of indirectly identifying what parts of the situation awareness are incorrect.

Furthermore, this thesis evaluates the applicability of the proposed approach through testing it on interesting intersection scenarios using a proof of concept implementation which considers speeding, right of way and having to yield. Corresponding scenario evaluation given by a human driving instructor served as a measure of correctness.

1.6 Thesis Structure

Chapter 2 Here we introduce relevant background theory related to the different aspects of our system.

Chapter 3 This chapter contains a structured literature review for identifying related research to the scope of this thesis, along with a synthesis of the work having been done in the field. Finally, based on the literature review, the motivation behind the scope of this thesis and an identification of its contributions is described.

Chapter 4 In chapter 4 the proposed system implementing our idea of a virtual driving instructor is explained. The way of representing system and context, the traffic domain, as well as situation awareness is described. Furthermore, an abstract overview of the system is provided, and the whole system design is summarized.

Chapter 5 The concern of this chapter is describing the simulator from which our data origins, as well as what data is output from said simulator and the shortcomings of this data. The extensions made to the simulator, enabling further reasoning about driving, are also documented within this chapter.

Chapter 6 Here, we document the requirements elicitation and implementation of the proof of concept system we have used for evaluation of our novel approach in providing feedback to drivers. The main architectural drivers are identified and discussed, techniques for taking care of these are documented, and logical and process views are included to document the workings of the system. Documentation of how the issues identified in chapter 5 is also included. Furthermore, the most critical agent logic implemented is documented. Finally, a rationale behind the chosen architecture is included.

Chapter 7 The experimental setup, experimental plan and their results are documented in this chapter. The experiments consist of initial acceptance testing making sure that the system is able to handle many legal configurations of an intersection, then user testing is performed to test the system in regular use. A select handful of interesting results are highlighted.

Chapter 8 Finally, an evaluation of the scenarios described in chapter 7 is included. Based on these results, we discuss the applicability of our novel approach in providing justifiable feedback to a driver and identify issues and propose improvements to our system. A final conclusion and suggested future work is also included.

Chapter 2

Background Theory

The goal of this chapter is to introduce the most relevant topics to understanding different aspects of the proposed virtual driving instructor system.

2.1 Context and Situation Awareness

A situation is not fully understood through a snapshot in time with information about all relevant entities within it, but also requires a context. An operational definition from the field of Human-Computer Interaction illustrates the concept of context well;

Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves (Dey, 2001, p. 5).

A framework that describes ones current understanding of a situation, termed "situation awareness" is introduced in Endsley (1988). The formal definition from this paper is: "The perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future."(Endsley, 1988, p. 791). Such a framework is well suited to describe mental models when it comes to decision-making. A clarifying definition building upon Endsley's has been introduced; "Situation awareness is the continuous extraction of environmental information, the integration of this information with previous knowledge to form a coherent mental picture, and the use of that picture in directing further perception and anticipating future events."(Dominguez et al., 1994, p. 11). By looking at these two definitions of situation awareness, situation awareness can be summarized

as the degree to which one is aware of the current context as well as the elements of a situation.

There are several factors that affect one's situation awareness. Some of these are individual, while others depend on the task at hand or the system one is using. Endsley (1995) lists these individual factors into groups, which are displayed in figure 2.1. The complete list of individual factors is abilities, experience, training, automaticity, information processing mechanisms, long term memory stores, goals & objectives, and preconceptions (expectations). An illustration of the entire process associated with acting based on one's situation awareness is given in figure 2.1

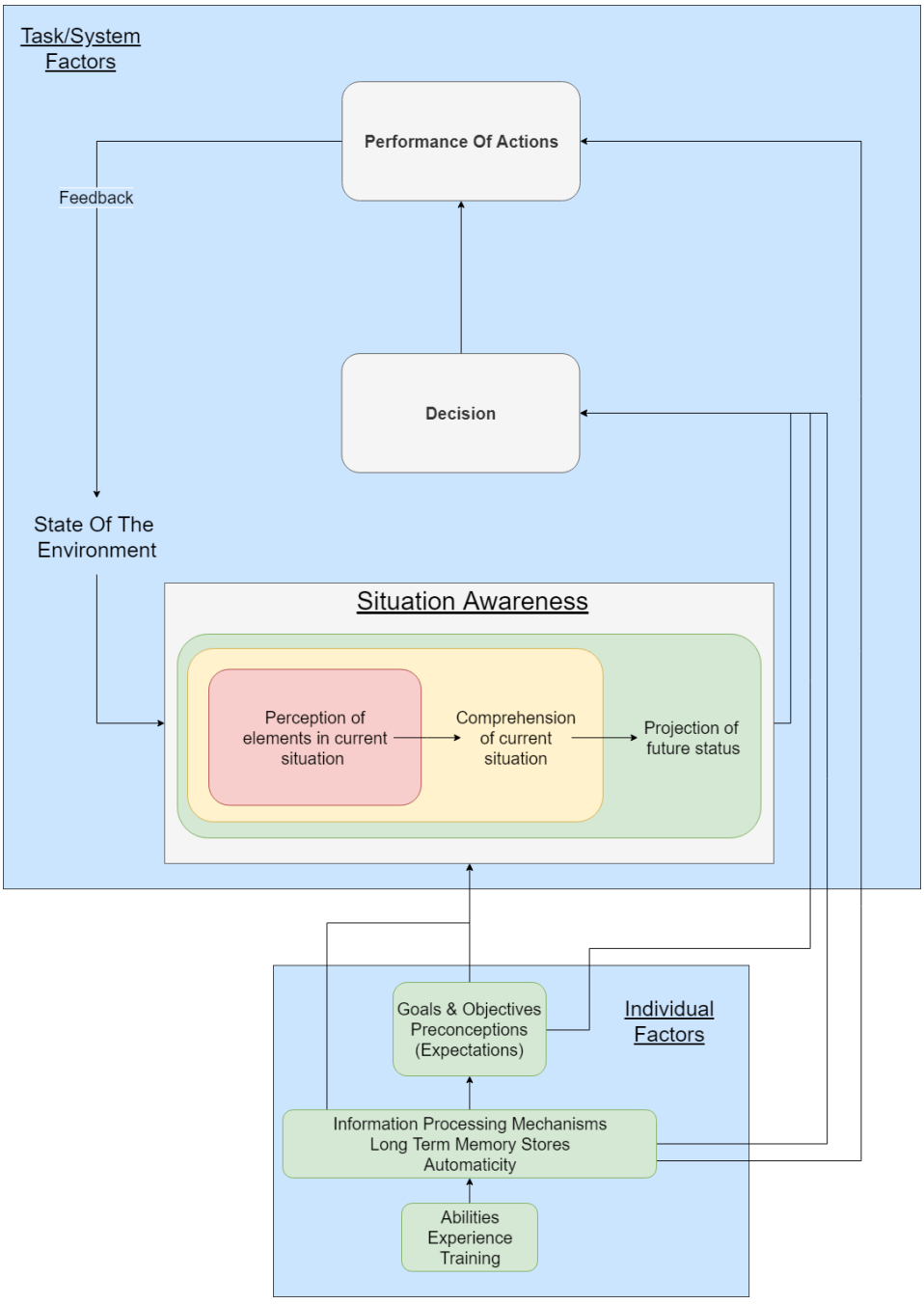


Figure 2.1: Adaptation from Endsley (1995, p.35), illustrating how situation awareness explains the process behind performing actions within an environment. Arrows indicate an effect upon a part of the process. Key features of the illustration include the different levels of situation awareness, as well as the set of individual factors, and how they together decide the performance of actions.

2.2 Ontologies

Ontologies, within the field of information science, serve as representations of some domain. The following definition explains how ontologies tend to describe some hierarchy of related concepts, with optional axioms added to describe relations between concepts:

An ontology is a logical theory accounting for the intended meaning of a formal vocabulary, i.e. its ontological commitment to a particular conceptualization of the world. The intended models of a logical language using such a vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment (and the underlying conceptualization) by approximating these intended models. (Guarino, 1998, p. 5)

Due to its use in this thesis, a short description of the Web Ontology Language, also known as OWL, will be provided. This is a family of knowledge representation languages used for authoring ontologies, that is based on the Resource Description Framework (RDF) standard.
An example of such an ontology description, the minimal one we have used for the proof of concept implementation of our system, can be seen in figure 4.1.
First of all, OWL consists of classes that correspond to concepts within description logics. An example of a class could be Car.
Furthermore, OWL classes are instantiated as individuals, called the same thing in description logics. An example of a Car instance could be *someCar*.
OWL also contains properties (known as roles in description logics), which are further divided into datatype properties and object properties. The former maps directly to attributes of a class individual, while the latter maps to a relationship from one individual to another. An example of a datatype property would be the *hasAge*-property, to state that the *someCar* instance of the Car class has an age of 17 years one could express this as *hasAge(someCar, 17)*. Furthermore, a *mustYieldFor* object property between Car instances could express the relation between one car and another stating that one has to yield for the other as *mustYieldFor(someCar, anotherCar)*.

2.3 Logic and Reasoning

First-order logic is a way to structure and reason with logical sentences. The sentences use constants, predicates, and functions as syntactic elements, where constants map to the objects of the world, predicates maps to relations between the objects, and functions are an alternative way of naming the objects of the world. As an example, given the situation where two cars drive alone on a straight road one after the other, there are five objects, the two cars, the road, and the left and right lane. Constants could be *Car1*=the first car,

Car2=the second car, and *Road*. A predicate can be DrivingBehind(*Car2*, *Car1*) symbolizing the relation that the second car is driving behind the first car. Instead of naming the left and right lanes, one can instead use the functions LeftLane(*Road*) and RightLane(*Road*). Note that these are not functions taking arguments and returning a value, but just another way of naming the objects. First-order logic also allows the use of variables, quantifiers, and logical connectivities to build complex sentences. "There is a car driving behind the first car" can be translated to $\exists x : Car(x) \wedge DrivingBehind(x, Car1)$ using the $\exists$ (*exists*) quantifier, a variable x, and the logical "And" operator; $\wedge$.

Though not explicitly written as first-order logic, OWL has a similar way of expressing domain-specific rules applying within an ontology. Rather these rules are defined through axioms, acting as constraints on individuals of an ontology. For instance, given there exists a *mustYieldFor* object property defined from the Car class to the Car class, and an *inverseMustYieldFor* object property defined from the Car class to the Car class, the axiom of "mustYieldFor owl:inverseOf inverseMustYieldFor" (expressed as an OWL triple here), enforces all occurrences of the *mustYieldFor* relation from individual A to individual B to have an inverse relation, *inverseMustYieldFor*, defined from individual B to individual A as well.
More complicated axioms can be added, expressing things like a Car individual speeding, or violating having to yield within an intersection, which is inferred by some inference engine used by an OWL ontology. The triplet syntax of OWL, borrowed from RDF, allows efficient reasoning on top of instances of an OWL ontology. Hence this language is well suited to the traffic domain. Another advantage of OWL is logical axioms being domain-independent, meaning that general knowledge applying to a certain domain only has to be written once, and inference and reasoning can be performed on any instance of an ontology. Thus, it is system agnostic and only requires the ontology instance to be provided in a valid format.

2.4 Blackboard Pattern

The blackboard pattern is a behavioral design pattern, first identified and used in the architecture of the Hearsay-II project Erman et al. (1980). Their architectural implementation of the pattern is shown in figure 2.2.

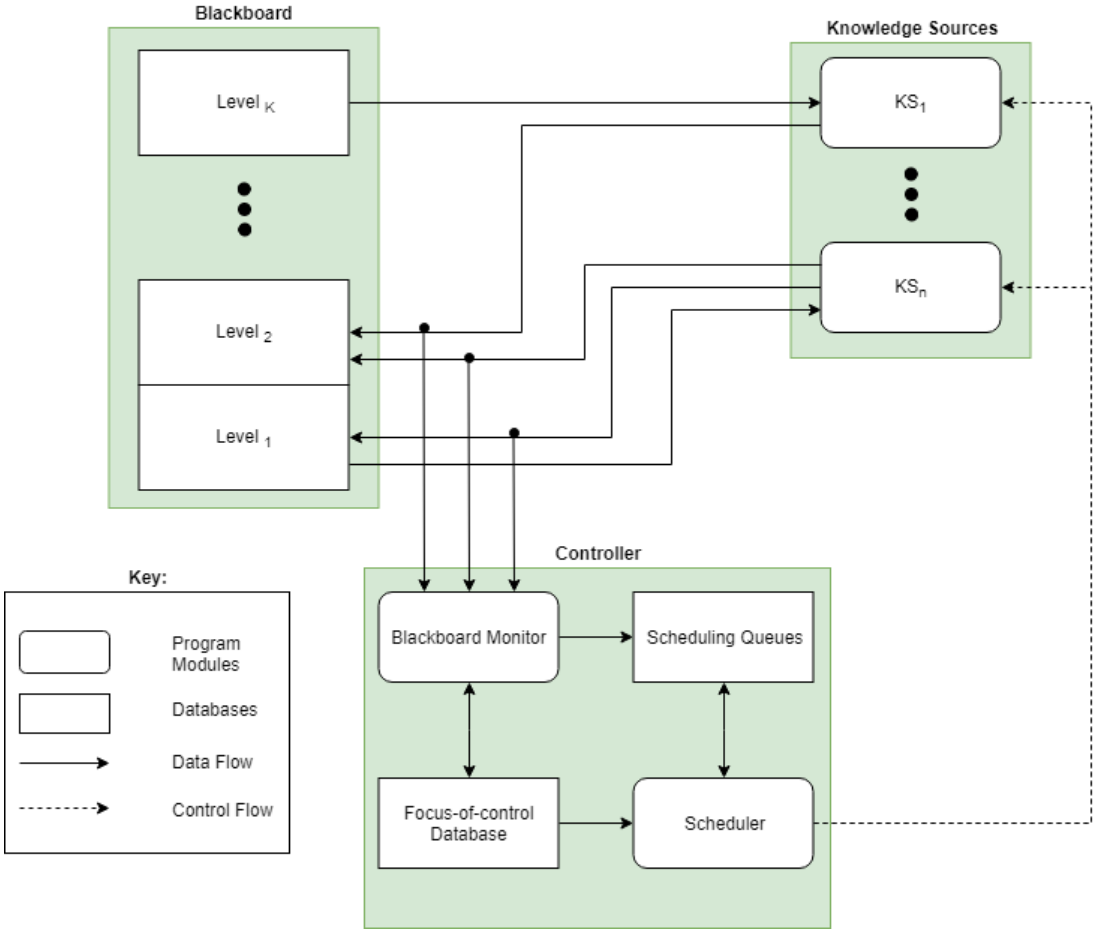


Figure 2.2: Adaptation from Erman et al. (1980, p.222), displaying their implementation of the blackboard pattern. The controller consists of a monitor that communicates with a scheduling queue accessed by a scheduler deciding which knowledge source is to be run next. Upon data flow from a knowledge source (updating state) to a level in the blackboard, any satisfied knowledge sources after this update would be added to the scheduling queue. Furthermore, the figure illustrates how there are several abstraction levels to the state of the blackboard, and how information in one level can be used by a knowledge source to add information to another level.

The pattern has been explained and clarified by both Nii (1986) and Corkill (1991). These authors explain how the pattern revolves around a central artifact; the blackboard. Furthermore, knowledge sources are able to both read and write to this blackboard, updating or adding values. Knowledge sources are schematized as condition-action pairs, whereupon some set of conditions

they are triggered to perform some action. This way a knowledge source only attempts to perform an action once its preconditions are satisfied by the state of the blackboard (i.e. when enough information is available). A controller is also present, responsible for deciding which knowledge source is allowed to write to the blackboard at what time. Such a pattern fits the situation where several different specialized modules share the same state, incrementally updated as computations finish.

Chapter 3

Related Work and Motivation

This chapter concerns identifying and synthesizing the research of fields related to our system through a literature review. Based on this literature review our motivation behind the implementation of a virtual driving instructor is provided. The chapter will also explain in detail the ideas we have been inspired by, or used, from other research.

3.1 Structured Literature Review Protocol

This section concerns documenting the structured literature review protocol used for identifying the relevant research related to the implementation of a virtual driving instructor, or a similar domain. The section describes the sources searched, what search terms were employed, which inclusion criteria had to be satisfied for a paper to be included, a rating on the quality of the included paper, as well as a description and summary of the discovered literature.

3.1.1 Identification of Research

This subsection documents how the research of importance to the implementation of a virtual driving instructor was identified.

Outline

First, an extensive wide search was conducted. Then the results of the initial search were filtered based on a set of inclusion criteria. Finally, the quality of

the remaining results was assessed.

Sources

The following set of digital sources was utilized when conducting the literature review:

- Elsevier
- IEEE Xplore Digital Library
- Springer Link
- ACM Digital Library
- CiteSeer
- Wiley Online Library
- Google Scholar

Search Terms

The search terms of table 3.1 were used to discover the relevant scientific literature.

Group 1	**Group 2**	**Group 3**	**Group 4**
Driving	Traffic	Situation Awareness	Driving Instructor
Car		Context-Aware	Driving Tutor
Road		Ontology	Multi-Agent System
		Ontology-Based	Autonomous Vehicle
		Driver Evaluation	

Table 3.1: The groups of search terms, where terms of similar meaning or within the same scope are grouped together. These groups and terms form the basis of the literature review.

The reasoning behind the following search term groups is as follows:

Group 1 : To restrict the scope of our search to research closely related to the concern of a virtual driving instructor, this group of terms should help limit it to the domains of driving, related to cars, or concerning the road.

Group 2 : A more strict limitation is including the single term of "traffic", as our system will be concerned with looking at the traffic domain. This helps filter out papers who simply mention any of our other keywords.

Group 3 : We are using situation awareness, or being context-aware, as a framework to describe the process of our virtual driving instructor analyzing and providing feedback upon actions taken by the driver, hence these two terms are included. The main job of the system is evaluating the driver, which is why "driver evaluation" was included. The representation of the current situation is to be that of an ontology, hence we would like to view other research which is ontology-based or otherwise takes advantage of ontologies in the traffic domain.

Group 4 : Finally, our system is to realize a virtual driving instructor, so research related to driving instructors or driving tutors is looked at. Furthermore, our preliminary approach is to use a multi-agent system architecture, therefore we are also interested in similar solutions. As there is probably overlap with how situations are represented and reasoned about in the domain of autonomous vehicles, this term is also included to identify any research overlapping with our concerns.

Search Formula and Procedure

The following formula was used to build one search:
$T_1 \wedge T_2 \wedge T_3 \wedge T_4$, where T_x is a term from Group X in table 3.1.

Searching was done using all possible logical conjunctions following the search formula. Furthermore, all of the constructions were run through each of the digital sources.

3.1.2 Selection of Primary Studies

The primary studies are the remaining results after all the results retrieved have been filtered. These will lay the foundation of the related work to our problem and system implementation. The filtering steps are outlined below.

Inclusion Criteria

Although the sets of search terms provide some indication of the relevance of the search results to the implementation of the system, this is not necessarily true. Therefore, a set of inclusion criteria is applied to the results to make sure they are relevant to our problem. These criteria are given in table 3.2.

ID:	Inclusion Criteria:
IC1	The study concerns representing or reasoning about traffic situations or a closely related domain.
IC2	The study concerns situation awareness in traffic or a closely related domain.
IC3	The study concerns a system providing feedback on driving or a closely related task.
IC4	The study concerns an architecture solving a similar problem to the implementation of a virtual driving instructor.

Table 3.2: The set of inclusion criteria that filters out research not closely related to the implementation of our system.

After applying the inclusion criteria of table 3.2, the set of results was substantially reduced to only 35 studies. For clarity, a study does not have to fulfill all of these inclusion criteria but is required to at least fulfill one. The remaining studies after filtering will be regarded as our primary studies.

3.1.3 Quality Assessment

The 35 results remaining after applying the inclusion criteria, will be rated according to the sum of their score over a set of quality criteria.

Quality Criteria

To assess the quality of studies, they have to be evaluated in terms of several criteria. We have chosen a set of criteria that should be general enough to apply to all studies found. Furthermore, they are reasonable expressions of academic qualities that a good study should fulfill, at least to some degree. The quality criteria are given in table 3.3. For each quality criterion, the result will receive a score of either 0, 0.5, or 1, depending on its satisfaction along that dimension.

ID:	Quality Criteria:
QC1	There is a clear statement of the aim of the research.
QC2	The study is put into the context of other studies and research.
QC3	The main contributions of the study are, if possible, validated through well-documented experiments.
QC4	The results of the study are thoroughly discussed and analyzed.

Table 3.3: The set of quality criteria to rate the remaining search results.

Assessment

The goal of assessing each of the results is to get an indication of their academic strength and the degree to which one should trust the findings or results of a given study. The resulting sorted list of results is given in table 3.4. The quality score of a study is not an indication that we should base our solution on their findings, but serve as a guideline to how useful the findings of this study would be in the case they are applicable to our problem. On the contrary, a low score indicates that the study's conclusion should be regarded with criticism.

Table 3.4: The remaining results after applying inclusion criteria, rated according to the quality criteria of our literature review.

Abbreviation	QC1	QC2	QC3	QC4	Rating
sukthankar1998	1	1	1	1	4
regele2008	1	1	1	1	4
konstantopoulos2009	1	1	1	1	4
hulsen2011	1	1	1	1	4
morignot2012	1	1	1	1	4
huelsen2014	1	1	1	1	4
zhao2014	1	1	1	1	4
mohammad2015	1	1	1	1	4
zhao2015	1	1	1	1	4
zhao2016	1	1	1	1	4
buechel2017	1	1	1	1	4
geng2017	1	1	1	1	4
zhao2017	1	1	1	1	4
raptis2018	1	1	1	1	4
tranvouez2013	1	1	1	0.5	3.5
sharon2017	1	1	1	0.5	3.5
provine2004	1	0.5	0.5	1	3

Continued on next page

Table 3.4 – continued from previous page

Abbreviation	QC1	QC2	QC3	QC4	Rating
gusikhin2008	1	1	0	1	3
sipele2018	1	0.5	1	0.5	3
hwang2003	1	0.5	0.5	0.5	2.5
arroyo2006	1	0.5	0.5	0.5	2.5
vacek2007	1	1	0	0.5	2.5
el2016	1	0.5	0.5	0.5	2.5
zamora2017	1	0.5	0.5	0.5	2.5
kappe2003	0.5	1	0	0.5	2
weevers2003	1	1	0	0	2
radecky2008	1	0.5	0	0.5	2
sun2011	1	0.5	0	0.5	2
geyer2013	1	0.5	0	0.5	2
gutierrez2014	1	0.5	0	0.5	2
martelaro2015	0.5	0.5	0	0.5	1.5
vlakveld2005	1	0	0	0	1
marti2009	1	0	0	0	1
fu2010	0.5	0.5	0	0	1
uschold2003	0.5	0	0	0	0.5

3.1.4 Data Extraction from Primary Studies

To provide an overview of the relevant literature discovered, a short summary of each study retained after the filtering from the initial searches is given. The papers considered most relevant will be expanded upon in the next section. The papers are introduced in order of rating, and then by year, as given in table 3.4. A synthesis of the papers is given in the next section.

sukthankar1998 - Multiple Adaptive Agents for Tactical Driving

Sukthankar et al. (1998) present a paper concerning the implementation of a multi-agent system for tactical-level reasoning on the task of driving. By having separate agents with a particular concern, multiple algorithms can be utilized. The system's overall dependence on each agent, is automatically tuned using a novel evolutionary optimization strategy, termed Population-Based Incremental Learning (PBIL). This system, which employs multiple automatically trained agents, can competently drive a vehicle, both in terms of the user-defined evaluation metric, and as measured by their behavior on several driving situations culled from real-life experience. In this article, the authors describe a method

for multiple agent integration which is applied to the automated highway system domain. However, it also generalizes to many complex robotics tasks where multiple interacting modules must simultaneously be configured without individual module feedback.

regele2008 - Using Ontology-based Traffic Models for more efficient decision-making of Autonomous Vehicles

Regele (2008) describes how a high-level abstract world model can be used to support the decision-making process of an autonomous driving system. The approach uses a hierarchical world model and distinguishes between a low-level model for trajectory planning and a high-level model for solving the traffic coordination problem. This initial high-level model suggestion serves as the basis for ontologies developed later on.

konstantopoulos2009 - Investigating drivers' visual search strategies: towards an efficient training intervention

Konstantopoulos (2009) has written a thesis with the aims being to identify some parameters that influence visual search and to develop an efficient training intervention that will improve drivers' visual skills. The difference in eye movements between driving instructors and learner drivers was examined during simulated driving. Results showed that driving instructors had an increased sampling rate, shorter processing time and broader scanning of the road than learner drivers.

hulsen2011 - Traffic Intersection Situation Description Ontology for Advanced Driver Assistance

Hülsen et al. (2011) provide an approach to create a generic situation description for advanced driver assistance systems using logical reasoning on a traffic situation knowledge base. It contains multiple objects of different types such as vehicles and infrastructure elements like roads, lanes, intersections, traffic signs, traffic lights and relations among them. Logical inference is performed to check and extend the situation description and interpret the situation e.g. by reasoning about traffic rules. The ontological representation is successfully applied to complex intersections.

morignot2012 - An Ontology-based approach to relax Traffic Regulation for Autonomous Vehicle Assistance

Morignot and Nashashibi (2012) describe a high-level representation of an automated vehicle, other vehicles, and their environment, which can assist drivers

in taking "illegal" but practical relaxation decisions. This high-level representation (an ontology) includes topological knowledge and inference rules, in order to compute the next high-level motion an automated vehicle should take, as assistance to a driver. The results of practical cases are presented.

huelsen2014 - Knowledge-Based Traffic Situation Description

A thesis from Huelsen (2014) concerns the goal of providing a generic traffic situation description capable of supplying various ADAS (Advanced Driver Assistance Systems) with relevant information about the current driving and traffic situation of the ego vehicle and its environment. With this information, ADAS should be able to perform reasonable functions and actions and approach visionary goals such as injury and accident-free driving, substantial assistance in arbitrary situations up to even autonomous driving. Among other things relative angles between intersections, the process of interpreting a traffic state to an ontology and using this in a rule-based system to reason and perform an action is described, as well as a detailed implementation of a possible collision with other cars approaching intersections. Furthermore, an extensive ontology and ideas of events for entering and leaving lanes, intersections, looking at traffic lights, etc. are described. Different from the paper of 2011, Hülsen also adds the notion of feature selection to handle uncertainty regarding which situation one is currently in. The thesis concludes that it has provided a proof-by-implementation for logic-based situation description for real-time execution of driver assistance functions. An asynchronous real-time framework is used, specially designed for the proposed ontological situation description.

zhao2014 - An Ontology-Based Intelligent Speed Adaptation System for Autonomous Cars

Zhao et al. (2014) present an ontology-based driving decision-making system, which can promptly make safety decisions in real-world driving. This initial work applies the knowledge base of the system to infer whether or not the car is speeding.

zhao2015 Ontology-based decision-making on Uncontrolled Intersections and Narrow Roads

Zhao et al. (2015) use their previously designed ontology-based driving decision-making system, on uncontrolled intersections in Japan (using their traffic rules) and two-way roads only wide enough for one vehicle.

zhao2016 - Fast decision-making using Ontology-based Knowledge Base

Zhao et al. (2016) improve on their earlier work with the process having been sped up by a new architecture having a smaller knowledge base of rules that apply to the current situation, than in their previous system.

zhao2017 - Ontology-Based Driving decision-making: A Feasibility Study at Uncontrolled Intersections

Zhao et al. (2017) test the improved ontology with success in uncontrolled intersections.

mohammad2015 - Ontology-based framework for risk assessment in road scenes using videos

Mohammad et al. (2015) propose a novel ontology tool for assessment of risk in an unpredictable road traffic environment, as it does not assume that the road users always obey the traffic rules. A framework for the video-based assessment of the risk in a road scene encompassing their ontology is also presented in the paper.

buechel2017 - Ontology-based traffic scene modeling, traffic regulations dependent situational awareness and decision-making

Buechel et al. (2017) present a modular framework for traffic regulation based decision-making of automated vehicles. It builds on a semantic traffic scene representation formulated as ontology and includes knowledge about traffic regulations. Situation awareness is used as an explanatory framework. The main contribution is their modularity, which allows for generic representation that can be reused for differing sets of traffic laws. The system has not been used for decision-making in traffic or a simulator yet (not real-time) but has been tested and validated in complex intersections with or without road signs and traffic lights represented as static scenarios.

geng2017 - A scenario-adaptive driving behavior prediction approach to urban autonomous driving

Geng et al. (2017) target the problem of most traditional driving behavior prediction models only working for a specific traffic scenario, and not being adaptable, as well as prior driving knowledge not being considered, the study proposes a novel scenario-adaptive approach to solve these problems. A novel ontology model was developed to model traffic scenarios. Continuous features of driving

behavior were learned by Hidden Markov Models. Then, a knowledge base was constructed to specify the model adaptation strategies and store prior probabilities based on the scenario's characteristics. Finally, the target vehicle's future behavior was predicted considering both a posteriori probabilities and a priori probabilities. The proposed approach was sufficiently evaluated with a real autonomous vehicle.

raptis2018 - DARA: Assisting Drivers to Reflect on How They Hold the Steering Wheel

Raptis et al. (2018) present DARA, the Driving Awareness and Reflection Assistant that makes drivers aware of potentially dangerous practices on how they hold the steering wheel and helps them reflect. One component recognizes how drivers hold the steering wheel and classify their actions through a Leap Motion controller and machine learning. Another is comprised of a mobile application that provides drivers with feedback during and after their drive. The system was successful both in making holding patterns present-at-hand for the drivers and in assisting them to reflect.

tranvouez2013 - A Multi-Agent System for Learner Assessment in Serious Games: Application to Learning processes in Crisis Management

Tranvouez et al. (2013) describe a multi-agent system that implements an Intelligent Tutoring System for assessing performance within crisis management in a simulator environment, where the situation is represented as an ontology.

sharon2017 - Protocol for Mixed Autonomous and Human-Operated Vehicles at Intersections

Sharon and Stone (2017) have made a protocol for intersection management with mixed autonomous and human-operated vehicles which would work even at 1% technology saturation, having improved efficiency over previous ones developed for decreased traffic delay.

provine2004 - Ontology-based methods for enhancing autonomous vehicle path planning

Provine et al. (2004) report the results of a first implementation demonstrating the use of an ontology to support reasoning about obstacles to improve the capabilities and performance of onboard route planning for autonomous vehicles. Experiments reason with whether or not you should change lanes depending

on the obstacle and your vehicle, performing a risk-analysis computation of estimated damage of collision versus that of changing lanes.

gusikhin2008 - Intelligent Vehicle Systems: Applications and New Trends

Gusikhin et al. (2008) discuss the various intelligent vehicle systems that are now being deployed into motor vehicles, where the topics are fuzzy-neural systems control, speech recognition, onboard diagnostics and prognostics, an overview of intelligent vehicle technologies, and driver-aware technologies.

sipele2018 - Advanced Driver's Alarms System through Multi-agent Paradigm

Sipele et al. (2018) propose an architecture based on the multi-agent paradigm for designing driver-centered ADAS that operates through data fusion. The principal goal is to design a hierarchical structure that can manage the knowledge acquisition process of all aspects involved in the driving scene such as the environment as well as the driver's behavior and state, providing support for building and testing reasoning models.

hwang2003 - Hybrid Intelligence for Driver Assistance

Hwang et al. (2003) document the architecture of an adaptive driver support system, that merges various AI techniques; agents, ontology, production systems, and machine learning technologies. The goal is to help drivers by managing their attention and workload, with driver specific adaptations. Their system is tested using a simulator and assumes that all model errors identified are driver errors. The result is an initial prototype to be improved upon.

arroyo2006 - CarCoach: a polite and effective driving coach

Arroyo et al. (2006) describe the design and evaluation of a context-aware driving advisor designed to promote better driving behavior. CarCOACH takes the information gathered from various sensors in the car and identifies common driving mistakes to appropriately commenting on driving behavior. The system presents scheduled feedback controlled in terms of quantity of total feedback and feedback with regards to a specific stimulus, and driver current state. Its goal is to reduce driver stress while maximizing the effectiveness of the feedback presented.

vacek2007 - Using case-based reasoning for autonomous vehicle guidance

Vacek et al. (2007) present an approach for situation interpretation for autonomous vehicles. The approach relies on case-based reasoning in order to predict the evolution of the current situation and to select the appropriate behavior. Case-based reasoning allows utilizing prior experiences in the task of situation assessment.

el2016 - Virtual Reality Driving Simulator Prototype for Teaching Situational Awareness in Traffic

El Aeraky et al. (2016) developed a virtual reality driving school simulator to help students learn how to drive and prepare them for real-world traffic and situations. The users can learn how to behave during various situations while navigating a fictional city and following a special quest. These situations happen at random without warning. The users are accompanied by a virtual computer assistant inside the car which serves as a driving teacher, feedback system and narrative device for storytelling and virtual reality display techniques (like superimposing a path) without breaking immersion.

zamora2017 - Intelligent Agents for Supporting Driving Tasks: An Ontology-based Alarms System

Zamora et al. (2017) discuss a rule-based alarm system as part of an ADAS. The main point of the proposed system is that it takes decisions based on the fusion of the information from the driver, the vehicle status and the state of the road ahead, and it is designed to alert the driver of the car when the system considers that it is necessary. Five dangerous scenarios are defined, analyzed and studied, and a repository of rules is designed to help the driver in those situations. The situation is represented as an OWL ontology, which the multi-agent system reasons on.

kappe2003 - Virtual Instruction in Driving Simulators

Kappé et al. (2003) describe some of the steps that were undertaken in the development of a 'virtual driving instructor'. They describe what it is to drive a car, how to learn that in practice, and how to learn that in a driving simulator. These steps are the basis for a discussion on virtual instruction in driving simulators, in which they present some of the current and future work on a cost-effective driving simulator. However, automated driving instruction is a complex process, requiring an extensive analysis on the selection, timing, and form of instruction and feedback. It also requires insight into the state and the

mental processes of a student. Human instructors are able to evaluate such processes relatively easy, but virtual driving instructors are not. The authors argue that a virtual instructor has to be complemented with a human instructor. The human and the virtual instructor should be able to cooperate, each attending to their own specialties.

weevers2003 - The Virtual Driving Instructor Creating Awareness in a Multiagent System

Weevers et al. (2003) introduce a virtual driving instructor or VDI, being a multi-agent system that provides low cost and integrated controlling functionality to tutor students and create the best training situations. An architecture for the implementation of this intelligent tutoring system is suggested. A tree structure of how different skills are built up is documented, and situation awareness is used to frame the student's understanding of a situation.

radecky2008 - Intelligent Agents for Traffic Simulation

Radeckỳ and Gajdoš (2008) deal with the development of intelligent agents with respect to their process specifications. The development process can be handled and documented by the standard UML tool. UML activity diagrams were extended for their purposes to Agent Behavior Diagrams. Next, the behavior reconfiguration principle is described in more detail. A learning mechanism of agents was specified thanks to the mentioned reconfiguration principle. This approach was implemented in the area of traffic simulation.

sun2011 - SmartAgents: A Scalable Infrastructure for Smart Car

Sun et al. (2011) propose a multi-agent framework: SmartAgent, which is developed specifically for Smart Car. The authors define and implement five agents according to their functionality. These SmartAgents can acquire environment context, make certain decisions based on the predefined policies and the information received from sensors, and react to the driving environment.

geyer2013 - Concept and development of a unified ontology for generating test and use-case catalogs for assisted and automated vehicle guidance

Geyer et al. (2013) present a fundamental ontology that is based on a consistent terminology for application in the field of assistance and automation of vehicles. This ontology allows the analysis of different concepts of cooperative and highly automated vehicle guidance in the early concept phase and ensures that research results can be exchanged and compared.

gutierrez2014 - Agent-Based Framework for Advanced Driver Assistance Systems in Urban Environments

Gutierrez et al. (2014) present a novel agent-based system focused on high-level reasoning as part of the development of Advanced Driver Assistance Systems. This approach focuses on driving safety, in particular, in urban environments in electric urban cars. The main point of the proposed approach is that it takes decisions based on the fusion of the information from the driver, the vehicle status and the state of the road ahead. The proposed system uses an OWL Ontology to represent the concepts and its relation to the urban traffic environment. This system is developed by using a novel multi-agent framework.

martelaro2015 - DAZE: a real-time situation awareness measurement tool for driving

Martelaro et al. (2015) have found a need for real-time SA (situation awareness) measurement designed specifically for both simulation and on-road driving scenarios during the development of interfaces for autonomous vehicles in both simulated and on-road environments. This paper concerns documentation of developing a tool, inspired by the Waze^{TM} driving app, to measure SA through real-time on-road event questions. The system has been tested in a lab and was to be further evaluated against current SA measurement tools.

vlakveld2005 - The use of simulators in basic driver training

Vlakveld (2005) discusses how driver simulators can best be used for basic driver training. The emphasis is not on the technical requirements but on the didactic requirements and the development of so-called coursework. A comparison is made between the use of simulators for training pilots and the use of simulators for drivers. If driver simulators can be used for the training of higher order skills like risk perception and situational awareness, is not clear yet.

marti2009 - A Rule-based Multi-agent System for Road Traffic Management

Martí et al. (2009) introduce a proposed architecture for a multi-agent system that is to control traffic. This system has two working modes: co-ordinately, where all the agents work to solve problems in large networks and locally where due to communications problems little groups of agents work together to inform road users about traffic problems.

fu2010 - Cognitive Awareness of Intelligent Vehicles

Fu and Soeffker (2010) have implemented a system for realizing cognitive awareness in a Java-application for intelligent vehicles. Underlying is the Situation-Operator-Modeling concept, which assumes that changes in the real world can be considered and understood as a sequence of effects modeled by scenes and actions. Data comes in from a driving simulator and is represented internally. Based on the current environment, an operator selection module selects a simple action to take.

uschold2003 - Ontologies for World Modeling in Autonomous Vehicles

Uschold et al. (2003) test the hypothesis that it is beneficial to use ontologies to augment traditional world modeling technologies for autonomous vehicles is explored by developing a theory of obstacles represented as an ontology. This ontology is to provide the basis for identifying and reasoning about potential obstacles in the vehicle environment in order to support navigation. Their work is preliminary, identifying challenges and issues to be investigated further.

3.1.5 Data Synthesis

A synthesis from the studies above is given. The general findings concerning a topic will be formulated, laying the foundation for how we approach solving our problem.

When it comes to representing situations in the traffic domain the overwhelming majority of research discovered in this literature review opt to use *ontologies*. It has both been showed to be flexible enough for several tasks; both autonomous driving, evaluating the rules applying in a traffic situation, or how risky the current environment is. Furthermore, logic expressed as ontology axioms allow for efficient rule-based reasoning on domain-knowledge contained within a knowledge base.

The situations are often framed using *situation awareness*. This framework describes how drivers, or driving instructors, are aware of the current situations, as well as domain knowledge, and how this leads them to take the actions they do. It has the possibility of being efficient means of identifying exactly what an instructor needs to provide feedback on, in the case of faulty actions having been taken in a situation.

Providing support to drivers, evaluating their performance or providing feedback has also been researched. In terms of providing feedback, experiments have

been performed with feedback being tactile, visual, auditory, through vibration, or even a multimodal combination. Furthermore, the visual search and attention of drivers has been used as a metric in predicting performance.

Architectures and implementations for virtual instructors or systems solving similar problems in the traffic domain have been documented and suggested. Some of these refer to the concept of intelligent tutoring systems. Common to these, are multi-agent implementations that allow for extension in the future as well as the separation of implementations concerns.

3.2 Related Work

Based on the synthesis of the previous section, and the structured literature review, this section will detail related work relevant to the topics our system will cover. It will look deeper into these studies, as our solution will reuse ideas or strategies from them. An additional paper, solving how to deal with the temporal aspect of relation events in ontologies, is also included due to our usage of their solution.

3.2.1 Traffic Situation Ontologies

Regele (2008) introduces elements of an intersection, like lanes, as well as relations between them indicating their priority in a situation where two vehicles have to avoid collision by reasoning about the right of way. Although not formalized in an ontology, this paper introduces an abstract topological model for reasoning about traffic rules. This work introduced the "conflicting"-relation between different paths one can take in an intersection. An important aspect of the way roads are represented in this paper is through the use of lanes. Lanes correspond to actual lanes, but there is also the concept of a *virtual* lane. These virtual lanes represent paths through an intersection. The suggested approach explicitly models the relation between any related lanes, meaning that the conflicting relation must be looked at and added between all paths through an intersection before the right of way can be reasoned about. An illustrative example is that for the case of a T intersection, there exist 6 virtual lanes, as each approaching lane can take one of two paths. Furthermore, this number of virtual lanes means there are a total of $6 \cdot 5 = 30$ virtual lane pairs where the conflicting relation must be considered. The amount of computation potentially involved in such a representation, especially for real-time processing of the current situation, makes it seem less than optimal.

Hülsen et al. (2011) formalize the ideas of Regele in an ontology, by introducing coordinates to intersections, as well as relative angles of incoming lanes. The

angle of other incoming lanes is represented relative to that of the ego vehicle. This paper focuses on analyzing the rules that apply to an ego driver. This makes reasoning about the right of way more effective, by not explicitly having to model the relation between each pair of lanes. The focus of this paper was on a more lean representation of traffic situations for logical reasoning. Again roads are built up of lanes, but these lanes are divided into either entering lanes, exiting lanes, or two-way lanes. Furthermore, traffic lights and road signs are also included in the suggested ontology. Rules are added as axioms to the ontology, to reason about whether or not one has to yield. One of their rules is given in equation 3.1.

$$\begin{aligned} CrossingPlain(?cr) \wedge Car(?c1) \wedge Car(?c2) \wedge approachesTo(?c1, ?cr) \\ \wedge approachesTo(?c2, ?cr) \wedge isConflictingFromRight(?c2, ?c1) \\ \wedge neg(same\_as(?c1, ?c2)) \rightarrow hasRightOfWay(?c2, ?c1) \end{aligned} \quad (3.1)$$

Later on, this ontology is applied in experiments of a simulator as reported in Huelsen (2014), with correct behavior seen as results for encountered intersections. Furthermore, the work extends that of the ontology by performing feature selection upon encountering traffic situations with uncertainty in them.

Zhao et al. (2015) utilize an ontology for reasoning about how an autonomous car is to infer whether to stop or yield in narrow roads based on the traffic rules of Japan. Here the authors rely on an incoming collision warning to prompt the inference of what action is to be performed. The inference is based on the current situation, represented in an ontology with a corresponding knowledge base. If the reference vehicle, the ego, is allowed to drive on according to right of way rules before the intersection it will do so. When coming to the intersection, the car will yield if it has a possible collision with any other car. In the case of other cars also waiting a set amount of time, it will start driving through the intersection. In the case of a two-way lane, it will always yield for any car approaching from the opposite direction. These three rules are extremely simple, and rather naive, but do not end up colliding with any other cars in their experiments. The main goal of the paper is a feasible ontology representation for traffic roads that can be used for real-time decision-making, which is achieved. A complete representation of the map they are driving in is initialized within the system, to begin with, thus they only have to keep track of their own position and other nearby cars. Their ontology is similar to that of Regele, explicitly stating whether or not roads are left or right of the others. However, they do not keep track of virtual lanes. The authors make the simplifying assumption that they know the direction other drivers end up taking. Furthermore, due to the simple rules, they do not reason about whether or not to yield for cars within the intersection based on their paths, but simply yield if they are able to collide with them.
In Zhao et al. (2017) their work is further improved, resulting in proof that knowledge-based reasoning on ontologies can be applied to real-time decision

systems of autonomous cars. This is achieved by rather than using the complete knowledge-base as before, only the relevant part of the knowledge base is retrieved and used based on the situation they are currently in. This is similar to how feature selection is used in Huelsen (2014), but a less sophisticated way of doing so by having to explicitly encode the domain-specific information into the application. For instance rather than reasoning across all the intersections within their preprocessed map ontology, they only look at the intersection they are currently in. While showing that reasoning on an intersection is feasible in real-time, the assumption that all intersections, as well as the layout of the road and information of all nearby cars, are available might be a bit too optimistic for a real-world scenario. Zhao et al. mention no drawback related with uncertainty (although their system assumes all information is present), but Hülsen does a great job of discussing and handling this issues, which happens to be very relevant for real-world scenarios.

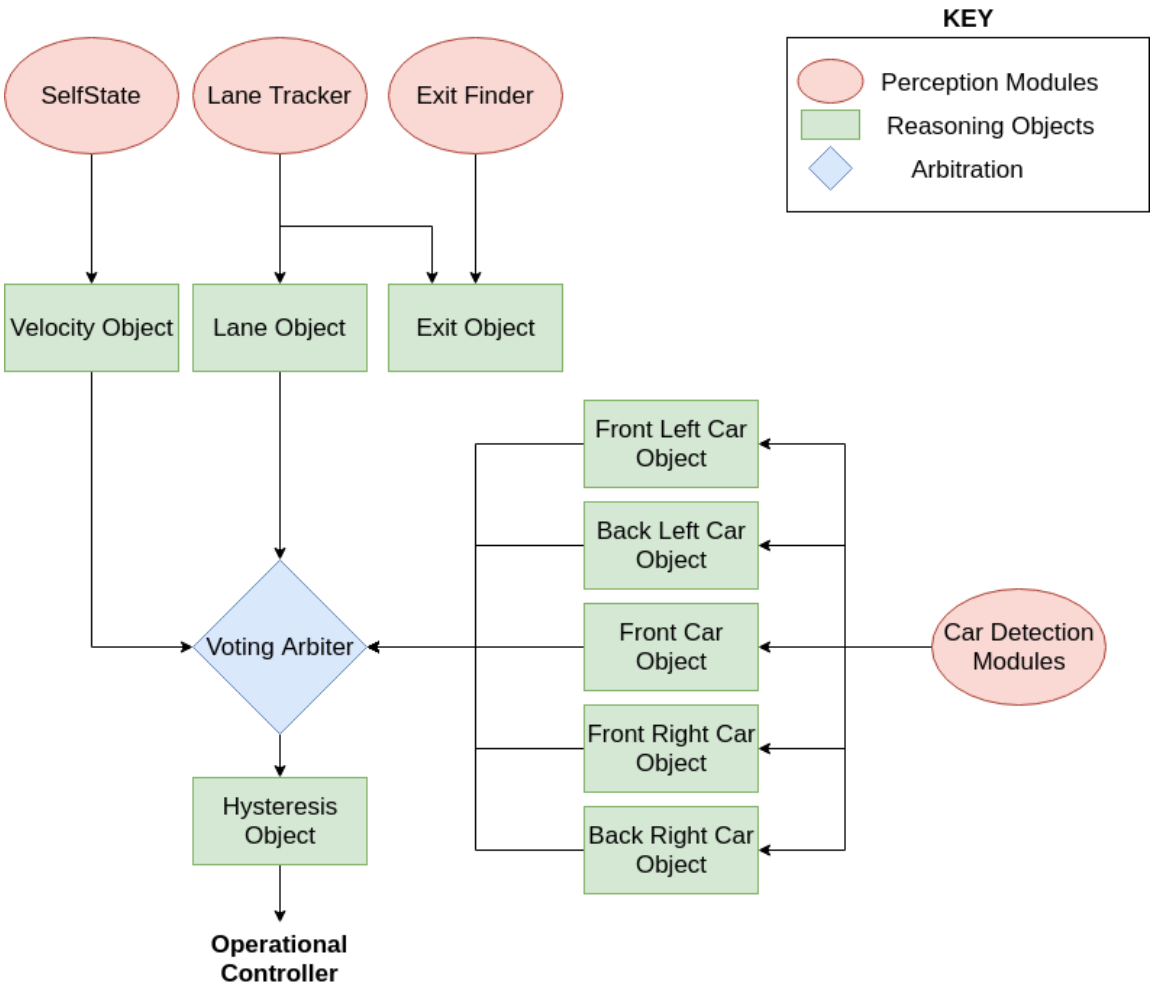


Figure 3.1: Adaptation from Sukthankar et al. (1998, p.94), displaying the collection of reasoning objects recommending actions based on local considerations. Each reasoning object monitors a subset of the perception modules, and vote upon a set of possible actions. A domain-independent voting arbiter chooses the action to take. Finally, the Hysteris Object maintains consistency from one time-step to the next.

Early work has been done when it comes to understanding traffic situations in Sukthankar et al. (1998), within a virtual traffic situation. The authors implement a system that frames the task of driving with situation awareness and implements a tactical driving strategy which is based on weighted voting among experts with different responsibilities. These experts are termed reasoning objects, which based on the current situation, supplied from perception modules,

and their area of specialization, vote for the next action to take. The votes are based on the internal goals of each reasoning object. For instance, all actions that move the speed towards the goal speed would be voted positively for by the speed reasoning object, while all actions maintaining the lateral position if already positioned correctly within the lane would be voted positively for by a lane position reasoning object. These reasoning objects, or experts, are allowed to veto actions. This is useful in the case where the expert responsible for keeping a distance to the car in front, as to not collide, knows that increasing the speed would end in a collision. The most popular, non-vetoed action is suggested as the next action to take by a voting arbiter. An evolutionary algorithm is used to tweak the voting parameters based on a user-defined fitness function. This architecture is shown in figure 3.1. This implementation for understanding and reasoning about a situation has huge benefits in terms of modularity and maintainability. If a change is needed in the reasoning of having to yield, one simply changes the internal logic and/or goals of the reasoning object responsible for yielding.

Reasoning about situations spanning several timesteps has been covered in Matheus et al. (2003). This work introduces several ways of representing the temporal aspect of situational awareness represented as an ontology. Among other things, the notion of an event holding the value of some relation between concepts within some time frame is introduced. By allowing values of relations to span across several timesteps a more efficient representation is achieved in terms of space. A dynamic system is queried for the value of a relation for a given timestep.

3.2.2 Providing Feedback Based on Driving Performance

Arroyo et al. (2006) make use of a blackboard architecture with violations receiving attention scores indicating their relative priority of being assessed, to provide feedback to drivers. When the attention score of a single violation, or a combination of violations, is above a threshold, a central mediator determines that a situation has arisen demanding corresponding feedback to be given. This feedback is put in a feedback queue and delivered when the attention scores indicate a low-stress situation where the driver would be more open to feedback. So far the system has only been utilized to provide simple feedback on things like braking pattern, but the idea of combinations of violations and a threshold providing feedback correspond well with how driving instructors provide feedback in real life.

In Raptis et al. (2018) feedback is given based on how the driver of a car is placing their hands on the steering wheel. The system presented consisted of two components, one using machine learning to analyze their driver's hand placement as either attentive or inattentive, and another providing feedback. The

component responsible for feedback would provide this during the drive, when the car stops, as an average if the car has moved more than 600 meters since it last stopped. The feedback is provided visually on a mobile phone mounted similarly to a GPS, as well as by a female voice to the driver. Furthermore, once the drive is completed, the system provides 5-second segments marked on a map from the drive, with color indicating the score. The authors conclude that such gamification of hand positioning helps motivate to perform better, however, the participants of the study generally preferred the feedback to be given immediately rather than the next time they are at a stop. This preference conflicts the ideas of Arroyo et al., as they insist instant feedback would disturb the attention of the driver.

Based on a multi-agent rule-based ADAS (Advanced Driver Assistance Systems) reasoning on an OWL ontology introduced in Zamora et al. (2017), the authors proceed with testing their system in Sipele et al. (2018). Here the authors actually use situation awareness when providing feedback to a driver, in a driving simulator. This feedback comes in the form of an alert from an ADAS to help the driver react in dangerous situations. The system contains information structured in several abstraction tiers, coming from agents using information in lower tiers, or simply outside sensors. Based on this information the system is able to recognize predefined dangerous situations and provide the driver with an alert so they can react in time. The evaluation of a driver's situation awareness in this paper is based on their visual attention. Thus, although the authors do evaluate the driver performance in terms of situation awareness, they do not directly reason about which part of the situation awareness is flawed. The scenarios in which the system is tested contain visual areas in which the driver should be looking for correct visual awareness. The timing of when a driver has, if at all, looked at this given place indicates how far in the situation awareness process they are. If they are not far enough in this process, the system provides an alert to make them act.

3.2.3 Virtual Driving Instructors

Although little work was found in the literature concerning implementing a virtual driving instructor, one relevant paper was found. An architecture for the implementation of a virtual driving instructor is suggested in Weevers et al. (2003), as an implementation of an Intelligent Tutoring System (ITS). The authors have as a goal to objectively measure student performance in different aspects of driving, and based on this performance create the best possible learning environment. The idea is a multi-agent system consisting of one agent implementing situation awareness, another implementing presentation awareness (presenting feedback based on the situational awareness), and a third implementing curriculum awareness (keeping track of progress within stages of the driving curriculum). Unfortunately, it is unclear whether or not this system

was implemented, as there is no further documentation found in the literature.

3.3 Motivation

This section will recap the proposed system, and then compare aspects of it to the relevant literature. The result of this comparison is a clarification of the contributions of this thesis.
The next chapter, chapter 4, extensively documents the proposed implementation of a virtual driving instructor.

Our proposed implementation of a virtual driving instructor will be utilizing the framework of situation awareness to reason about what information a driver is lacking, or what rules they do not have a correct understanding of. Using a multi-agent system, the implementation will incrementally add information at an increasingly abstract level, starting with basic information coming in from a driving simulator, and ending in projections of the future, for instance, projected collisions based on the path of other vehicles. This information will be contained in an ontology, on which a reasoner will infer whether any violations exist.
The situation awareness of the system is perfect, meaning it has all relevant information about the situation and context, is aware of all domain knowledge, and therefore able to correctly project future states. With the assumption that drivers perform the action they consider correct, given their perception of the current elements in the situation, understanding of traffic rules, and resulting projection of future states, incorrect actions indicate flawed situation awareness and indirectly highlight the fault.
Based on what mistake is made, and the current situation, our system is going to generate feedback as constructive criticism for correcting the situation awareness of the driver, hopefully improving their skill as a driver.

Using a multi-agent system architecture to reason about a traffic situation is not new, Sukthankar et al. (1998) documents a similar architecture to ours where voting among agents is used to select the action of an autonomous vehicle. Each agent has its own specific responsibility within traffic, and together they make up a well-informed understanding of the current situation.

Ontologies have been used to represent the traffic domain before. Regele (2008) documents early work on reasoning of right of way in an intersection encoded in a high-level representation, while Hülsen et al. (2011) proposes a more lean ontology including the concept of relative angles between lanes of an intersection. Huelsen (2014) also looks at the case of how to handle missing information

about the current situation, using feature selection to figure out which situation is more likely. Zhao et al. (2015) and Zhao et al. (2017) document successful reasoning when all information about the intersection, as well as other vehicle's paths, are known in advance. This reasoning was done using an OWL ontology and concluded feasible for real-time decision-making.

The concept of a virtual driving instructor is not new in itself. An early suggestion for an architecture was documented in Weevers et al. (2003). However, no implementation of their proposed architecture was discovered in our literature review, and it is unclear as to whether or not the authors did, in fact, implement the architecture themselves. One example of providing feedback to a driver based on their performance was described in Arroyo et al. (2006). Here a blackboard architecture was used to check if the combination of violations in a current situation exceeds a level, and in this case, feedback is generated and added to a queue. To avoid distracting the driver, feedback is only presented once the stress-level of the situation is considered low. Raptis et al. (2018) show a more recent approach to providing feedback to drivers, looking at the way they hold the steering wheel, and providing feedback audiovisually once the driver comes to a complete stop.

Finally, there is a set of related research very close to our proposed virtual driving instructor implementation. Zamora et al. (2017) and Sipele et al. (2018) use a multi-agent ADAS (Advanced Driver Assistance System) with situation awareness to frame the current state of the driver in their decision process. Their system alerts the driver when necessary to notify them of danger. The system considered five different and potentially dangerous scenarios. One of these scenarios is approaching a pedestrian crossing. Their use of situation awareness is seen in the way they utilize information from an eye tracker. If the user has looked at a predefined region they consider it perceived, and therefore the process of acting as initiated. Furthermore, the eye tracker data was used to decide whether or not the driver is attentive. Whenever the system considers the driver inattentive, or the driver has not perceived a key situation element, their system would alert the user to start their process of acting. An example is a possible collision alert indicating that a car in front is about to collide with you, which should prompt you to start breaking even though you were inattentive or had not seen and perceived said car.

Although many of the ideas described above are reused, our proposed solution separates itself in the way it evaluates driving behavior and provides feedback to a driver. Thus, the novelty and contribution of the solution discussed and tested in this thesis comes from using a multi-agent system and situation awareness to identify flaws in a driver's situation awareness and also give feedback to correct the identified flaws. As a reminder, the work relies on the underlying assumption that a driver acts with the best intention, meaning that any incorrect action

origins from a mistaken understanding or perception of the situation. In other words, incorrect actions are a result of incorrect situation awareness in the driver.

Chapter 4

Proposed Solution

The goal of this chapter is to introduce the design of a system answering the research questions introduced in chapter 1. The rationale behind the decisions taken is all based on the literature identified in chapter 3. First comes an overview of how the Virtual Driving Instructor's implementation will be laid out, then following this are proposed solutions to subproblems of implementation. While this chapter simply outlines the solution, the following chapter (chapter 6) will detail the architecture of the implemented proof of concept system.

4.1 Situation and Context Representation

As is apparent from the scope of the literature review of chapter 2, the de facto standard of representing the situation for the task of driving, is using an ontology. Therefore, we will also use an ontology to represent the situation.

Furthermore, Endsley (1988) describes how an actor within a specific task domain has a certain degree of understanding about the current situation and context, as well as the projected future evolution of it (situation awareness as described in chapter 3). For the task of implementing a virtual driving instructor, the system will have perfect situation awareness, i.e. knowledge of all the elements of the current situation, have perfect comprehension of the situation, and also a correct projection of future status. However, a learner driver being evaluated by our virtual driving instructor could make mistakes due to incorrect situation awareness. With the assumption that all drivers drive as well as they can, according to the traffic rules they are aware of, as well as their comprehension of the current situation, one can evaluate their actions. Imperfect action is the result of an incorrect projection of future status, which, with the preceding assumption, can only be due to an imperfect comprehension of the current situation. This can either be due to misconceptions of traffic laws or

not having perceived all elements of a situation. The virtual driving instructor will, therefore, provide feedback to the driver whenever the projection of future status resulting from actions taken that violate traffic laws. Due to the ontology representation, the virtual driving instructor will also have available all the information about the current situation, which serves as a basis for corrective feedback to correct the faulty situation awareness of the driver.

4.2 The Traffic Domain

Specifically for the subproblem of representing aspects of traffic domains, ideas from a set of papers will be reused. These ideas are described below. The most challenging aspect of modeling and reasoning in the traffic domain concerns intersections. What vehicles have to yield for others is the most challenging aspect of intersection situations, which is the scope of experiments performed to evaluate the performance of our system in chapter 7.

Regele (2008) was the first one to introduce a "conflicting" relation between all the possible ways one is able to travel through an intersection. However, due to the way traffic laws of yielding are designed, one does not have to explicitly model the relation between all possible ways of driving through an intersection. Hülsen et al. (2011) mentions the notion of relative angles of roads to intersections. By introducing relative angles, rather than having to represent the relationship between 30 different ways of driving through a T-intersection with 6 lanes connected to it, one only needs to keep track of the 6 relative angles of the lanes. Furthermore, relative angles allow general functions to be written that capture the left- or right-hand rules applying in the given country one is driving in. These two advantages are enough for us to adopt using relative angles in intersections.

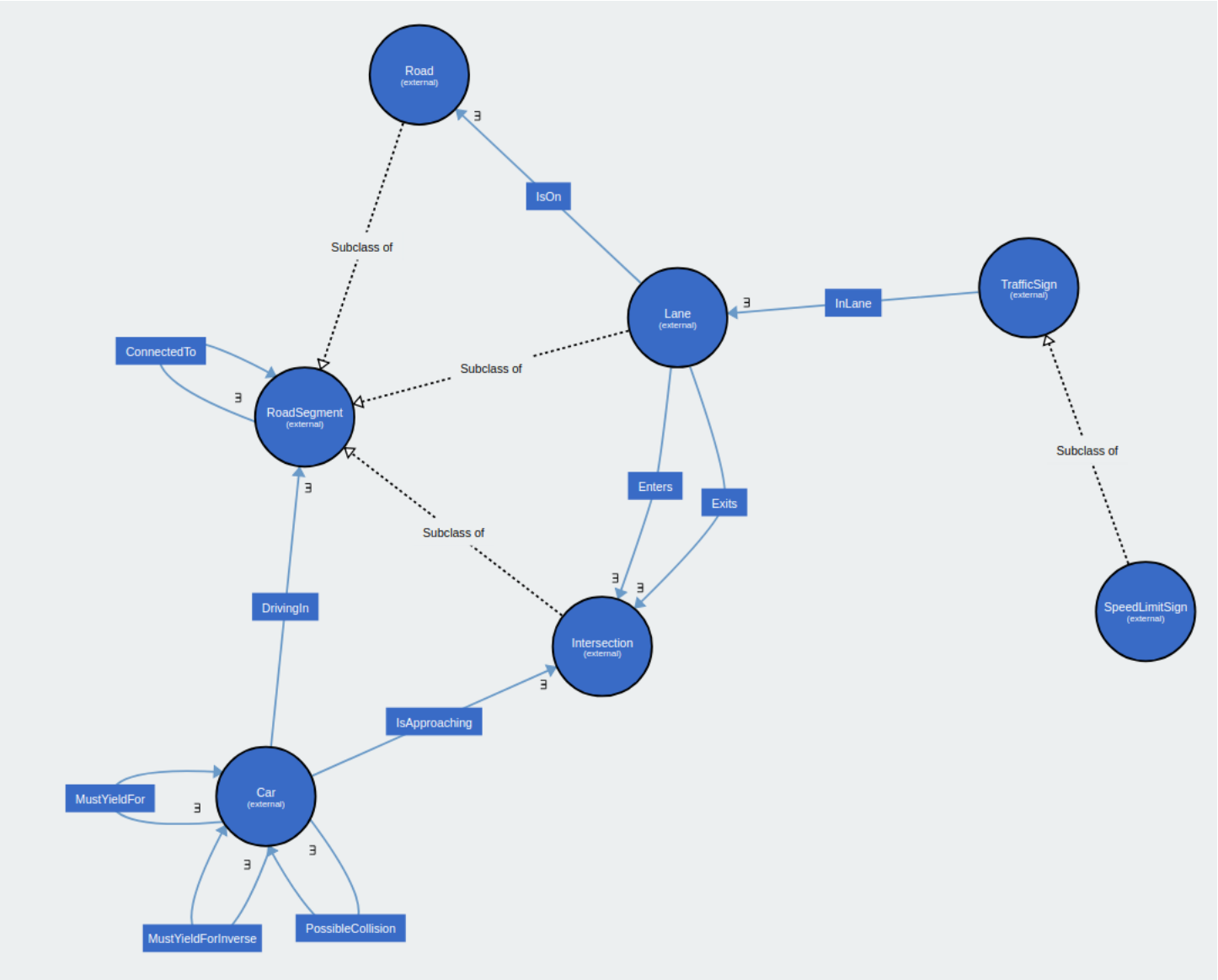


Figure 4.1: The minimal OWL ontology used to describe the traffic situation that our Virtual Driving Instructor uses for reasoning on driver actions. For clarity, only the object properties are included in this figure, while the datatype properties have been omitted. The same goes for the different kinds of road signs, only the speed limit-sign has been included.

Apart from the above-mentioned features, our proposed ontology keeps track of lane connections to intersections, whether or not they are an incoming our

outgoing lane, the cars within the situation, their speed, position, and acceleration, as well as which road segment they are currently in and therefore what intersection they are approaching. Cars also have relations to each other indicating having to yield for the other car, or having right of way. A road segment is a generalization and could either be a lane or an intersection. Road segments are connected to each other. The complete ontology is given in figure 4.1. This is a minimal ontology for reasoning about intersections without traffic lights as of right now, both the ontology and the system are open to extension in the future.

4.3 System Design

As concluded by the literature review of chapter 3, most systems wanting to reason with information from some domain are designed as multi-agent systems. Multi-agent systems have the clear advantage of rather than implementing some general algorithm having to solve all subproblems related to some domain, specific algorithms can be implemented within each agent, whose sole purpose is solving some subproblem. Furthermore, multi-agent systems allow simple extension or modification of an already existing system.

Most interesting to us is the approach shown and thoroughly explained in Sukthankar et al. (1998), where agents have traffic situation specific areas of focus, like maintaining position within one's lane or maintaining a safe distance from the surrounding cars. The set of agents cooperate by voting for a suggested action based on the current situation and projection of future status in their respective driving tasks. In a similar manner, Sipele et al. (2018) show how their system concerns several tiers, with the perception tier being at the bottom taking in raw sensor data, and information tier defined on top of this, further up is a cognitive tier and finally an actuation tier. Inspired by these two approaches, our system will consist of a set of agents with a single responsibility. Each responsibility will be adding information to the current situation based on the information in the same or lower abstraction levels. One example would be adding information regarding what cars have to yield for each other by using their current positions and future paths, where the information of what yield rules apply can be used to reason about whether or not one is violating the traffic laws.

In Arroyo et al. (2006), their system uses a blackboard pattern, which is a pattern that fits well for multi-agent systems. This pattern is described in greater detail in chapter 2, but is basically a way of allowing agents to subscribe to a set of conditions, performing some action once these are present. In our case, this pattern allows us to define the set of conditions required for each agent to be able to reason on the current situation and add new information.

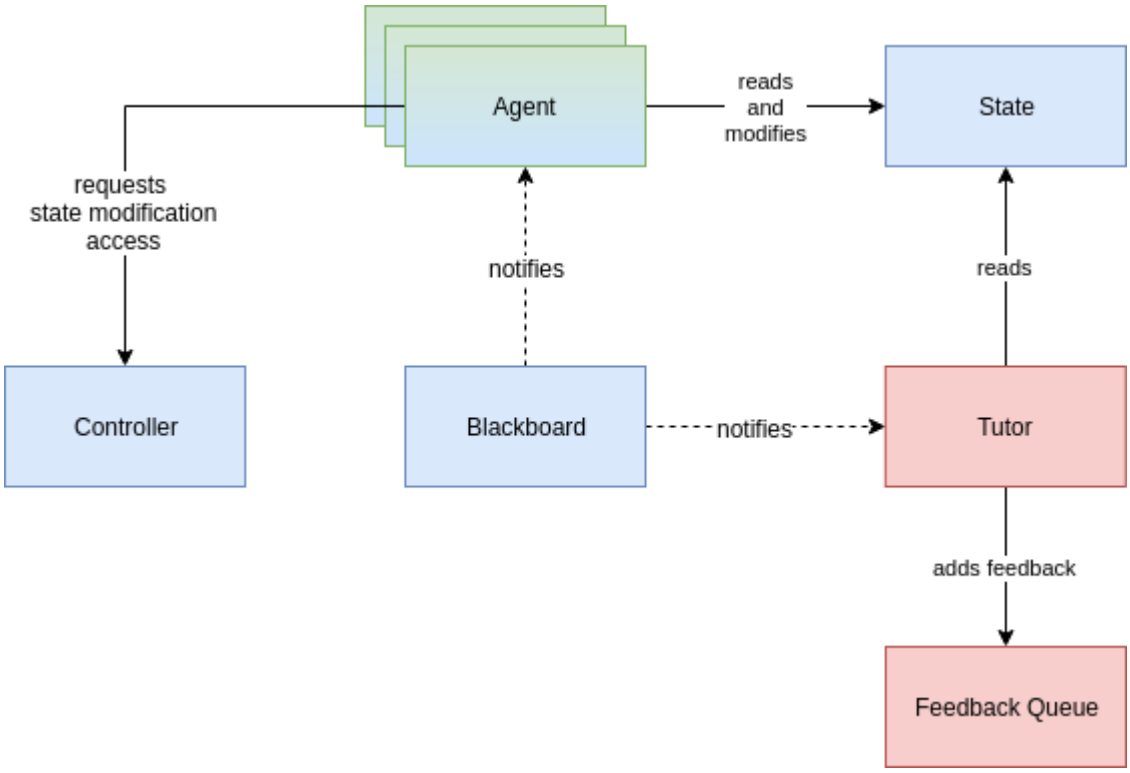


Figure 4.2: The abstract view of our proposed system solution. The proposed solution is designed as a multi-agent system, sharing and adding information using the blackboard pattern, while having a tutor evaluate the current state, once notified of required information being available.

In conclusion, our system is going to utilize a multi-agent system architecture, where conditions of the blackboard pattern are used as a scheduling mechanism to iteratively add increasingly abstract information concerning the given situation.
At a given time, the set of agents with their preconditions satisfied by the current information available in the state (representing the current situation) are evaluated by the controller, where the one with the highest priority is run. The highest priority belongs to the agent that provides the highest information gain to the current situation description.
Once a set of satisfied conditions are added to the blackboard (new information added), the agents interested in this information are notified and can request permission from the controller to add their information to the state. Once enough information is available for the tutor to provide feedback on the actions taken in a given situation, the blackboard notifies the tutor. This tutor is run separately from the rest of the agents and will generate feedback when appro-

priate, not having to ask for permission. This feedback is to be provided to the driver by the system where deemed appropriate and is therefore added to a feedback queue from which it can be consumed. In an appropriate situation of lower stress right after a significant traffic event has occurred, for instance, right after an intersection, the system will consume feedback generated from the given situation and provide it to the driver in real-time during their drive. We agree with the conclusion from Arroyo et al. (2006), that feedback should be presented in lower-stress situations, rather than instantly as requested by subjects of the experiments in Raptis et al. (2018). The abstract description of our proposed system solution is given in figure 4.2, while implementational details of our proof of concept are found in chapter 6.

4.4 Summary

To summarize, our proposed solution implements a virtual driving instructor which provides feedback to a driver based on their potentially incorrect situation awareness in a traffic situation. Due to the assumption that a driver performs actions to follow traffic rules as best as possible, based on their current understanding of the situation and its context (which includes the applying traffic laws), one can assume that an incorrect action corresponds to incorrect situation awareness. The virtual driving instructor itself will possess a perfect situation awareness.

Representing the traffic domain is done using an ontology, which will include all relevant entities (classes) of the domain, relations (object properties) between instances of these entities, as well as datatype properties of each instance.

A set of axioms describing violations of traffic laws are defined in an ontology, having a reasoner which will infer whether a given instance of an ontology contains any violations. In the case of a violation, this is reported to the tutoring part of the system, having encoded expert knowledge into it, which will generate feedback formed as constructive criticism to improve the assumed incorrect part of the driver's situation awareness.

To add the necessary information to the ontology, our system will be composed of multiple agents with a specific responsibility. These have encoded domain-specific knowledge into them. A controller will use priority based on the information gain each satisfied agent can contribute for a given set of information about a situation, and thereby schedule their additions of information. A set of conditions is used to tell whether or not an agent can add information, thus indirectly imposing a dependency of agents adding information at a lower abstraction level to those at a higher abstraction level. These agents will also

realize the projection of future status of the situation awareness.

The system will provide feedback in real-time, which will be presented to the driver in situations where this is appropriate. It will not be provided in high-stress situations, but as close to a relevant traffic event as possible for temporal relevance and the highest amount of learning benefits.

Chapter 5

Simulator Description

This chapter will describe the driving simulator for which our system has been developed, and in which our experiments have been conducted. Not only is the simulator highly relevant as the proof of concept system is based on data coming from it, but the proof of concept is also limited by the degree of detail and amount of information available from the simulator's virtual world. Both the physical architecture of the simulator and the data output from it will be described below. Finally, a list of shortcomings in the available data from the simulator is listed and discussed.

5.1 The Driving Simulator

The simulator of Way AS consists of an actual car, where the controls and engine are connected to a simulator system. Apart from the engine not actually running, and the wheels turning, the car operates like a normal car. The inputs of a driver are forwarded into a local network which the simulator computers are connected to.
Furthermore, the car is situated on top of a motion simulator which is able to simulate acceleration to correspond to the simulator world, and driver inputs, as well as tilt the car. This achieves a higher degree of immersion. However, it is not enough to fully resemble the physical reactions that experienced drivers expect. As a result, many experience motion sickness the first few times they drive in the simulator.
The car itself is placed within a blacked-out room, with projectors projecting the simulated world 360 degrees around it. For extra immersion, an extra monitor has been installed in the rear window. The projectors and this screen are all connected to different computers, rendering their own part of the simulated world. A master-computer accepts input coming in from the car, apply these to the current virtual world state, and emits the results to the rest. This means

that all of the different computers share the state of the simulator world, but render different parts of it. A picture taken from within the simulator room is given in picture 5.1.

Figure 5.1: A picture taken from within one of the rooms in which Way AS has a driving simulator. A car is driven as normal, without the engine running, and wheels turning, but the driver input is emitted onto a local network. The resulting actions are performed in the simulator world, and the world is projected 360 degrees around the car, as well as to a screen in the rear view window.

5.2 Available Simulator Data

The previous section mentioned how the driving simulator shares a simulator world state. This state is comprised of the current position of the car, other cars, as well as their speed. Furthermore, these cars are located within a traffic situation, containing roads, lanes, road markings, road signs, buildings, sidewalks, etc.

To allow reasoning with the state of the simulator, information about the world is required. The simulator is able to output information about the world, currently being as an XML log file summarizing a drive after it is finished. The data within this file restrict the scope and quality of reasoning that is possible to do.
The contents of this log file provide information on the following for each

timestep:

Car Position: Each car within the simulator has the triplet of its coordinates logged for each timestep.

Car Orientation: Each car within the simulator has the quadruple representing its orientation logged for each timestep.

Ego Vector: For each timestep the inputs coming from the Ego is logged as well. This information contains the steering wheel angle, how hard the throttle, brake, and clutch is pressed, whether or not the parking brake is on, their RPM, speed (absolute value) and torque, as well as a light vector representing indicator lights, a custom flag, and what gear they are in.

5.3 Simulator Data Shortcomings

There is a discrepancy between the set of data available within the simulator and that which is contained within the XML log files. This section will describe the shortcomings of the XML file from the view of a system wanting to reason about traffic situations, and end up with a list summarizing what is required for a system to be able to reason about a traffic situation.

While the data of the XML provides the position and orientation of all cars, the speed is only provided for the ego. Furthermore, acceleration is completely missing. To be able to project the future state, a key part of the situation awareness for making decisions, the speed and acceleration is required to tell where cars will be in the near future.

Related to speed, acceleration, and position is also wherein the road network a car is. While the XML file contains the exact coordinates within the simulator of a car, no point of reference other than the origin is provided. What road a car is currently driving on, which lane it is in, whether it is in an intersection or not, etc., are all very relevant parts of the situation description. Without the information of the road network, the amount of reasoning that can be performed is scarce.

Closely related to the road network are road markings and road signs. It is based on previously passed road signs applying to a road segment that a driver context is formed. As described in chapter 2, one's situation awareness is based on both the current situation as well as the context. Without the XML log telling the system which road signs are passed, or what lanes coming into an intersection have a yield sign, it is hard to reason about anything other than performance on driver input and whether or not cars collide provided the travel

in a straight line. Assuming a straight line disregards the road they are driving on, and is, therefore, a naive approach to collision detection.

A less critical part of the information, but still relevant, is being able to tell where other cars are headed. In a perfect world, everyone would know the other car's paths, but in the real world (and in this simulator), indicator lights are used for this purpose. This information is not present in the XML log, and therefore not available to our system. Knowing where other cars are headed can make the traffic flow more efficient, as you can infer whether or not you have to yield for another car's path.

The following list summarizes the shortcomings of the XML log's data, in no particular order of importance. It consists of the information we consider most relevant for reasoning about traffic situations that is not currently available in the simulator logs.

- Car Speed Vectors
- Car Acceleration Vectors
- Road Network Information (roads, lanes, intersections, etc.)
- Road Signs
- Road Markings (broken lines vs. full, crosswalks, etc.)
- Car Indicator Lights

How we overcome these shortcomings within the proof of concept implementation is described in chapter 6, as well as in the additions to the simulator described in the next section.

5.4 Simulator Additions

As described above, several important bits required to reason well about the traffic situation is missing from the simulator log file. This section explains how the simulator world was extended to add logging of some of the information. Extensions were implemented by us to log information regarding both road network information, i.e. car positions in the road network, as well as road sign information.

With these two crucial pieces of information, our system now has data indicating where in the road network cars are. By including prior knowledge of the road network layout into our system initialization, we can tell which cars are approaching the same intersection, and reason about yielding based on their

respective lanes. The prior knowledge of road layout includes which lanes are connected to which intersections, whether they are incoming our outgoing lanes, and their relative angles into the intersection. Furthermore, the system now also includes information on road signs passed stored in the context, which enables more sophisticated reasoning.

The simulator already has an internal event system, which was extended to allow us to log events with specific IDs and value upon colliding with triggers within the Unity world that the simulator is based on. These triggers and events are described below.

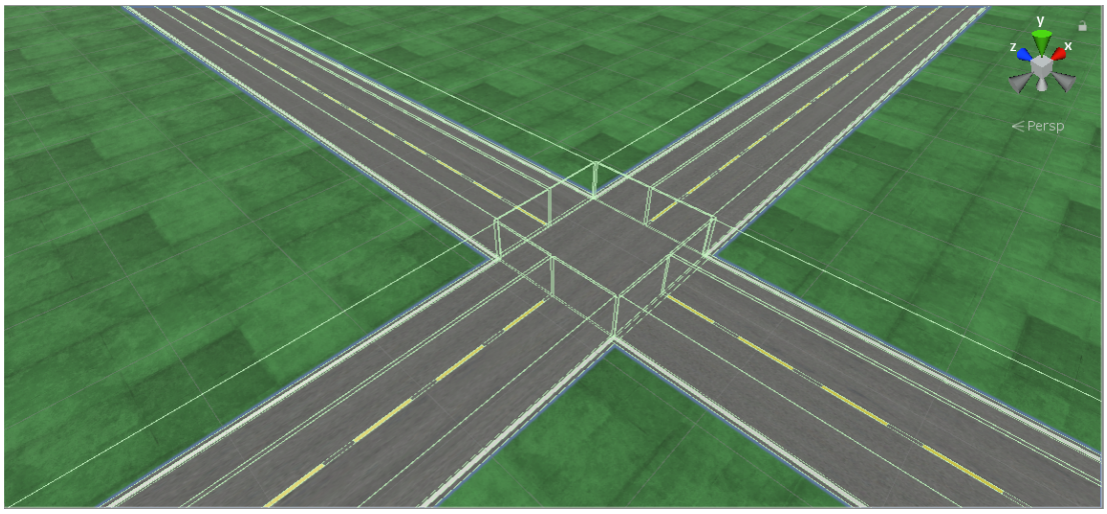


Figure 5.2: A picture from the extension added to the simulator. The transparent boxes with green borders are event triggers spanning lanes and the intersection, generating events upon cars entering or leaving them. Furthermore, each event trigger has a unique ID identifying the road segment.

5.4.1 Events

The first thing done to help with the shortcomings of the simulator information was adding triggers that would log events to the XML. These events consist of an event ID, a flag, a data field, and a value. Two new events were added to the already existing event system of the simulator, the road sign event, and lane change event.
An event for road signs passed, the road sign event, was added. This has an ID indicating what type of road sign was passed, the data contains the ID of the car passing the sign, and the data the value of the sign (if applicable, like for a speed limit sign).
A similar event was added to inform cars entering or exiting road segments, the

lane change event. This event has an ID indicating a lane change. The data field of a lane change event contains the car ID, while the value field contains the road segment ID (road segments are a generalization of different parts of the road network, encompassing lanes, intersections, etc.). The event flag indicates whether or not the car is entering or exiting the road segment.

5.4.2 Road Signs

An event trigger was placed at each road sign of the intersection, as shown in figure 5.3. These triggers are transparent boxes, detecting collision with other cars. Upon collision with such a trigger, a road sign event is fired, which is added to the XML log coming from the simulator.
Given this information, our system is able to update the context applying to a given car at a given time. After a car passes a speed limit sign, the speed limit of their context is updated to reflect the fact that a new speed limit applies to them. Similarly, when a car passes a right of way sign, their context is updated to reflect them having right of way.

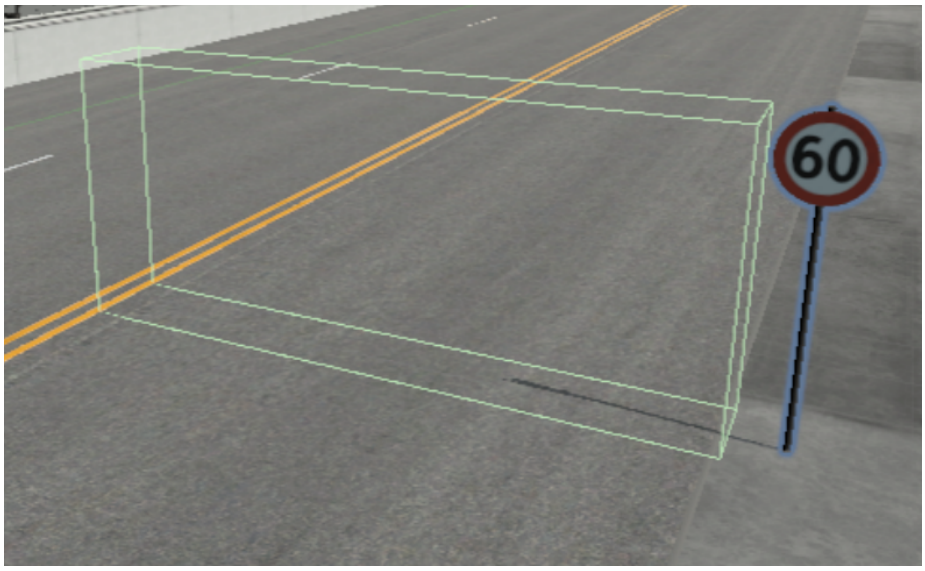


Figure 5.3: A picture from the extension added to the simulator. The transparent box with green borders is a collision detector that will generate a road sign event upon cars passing it. The event has an ID indicating the type of road sign.

5.4.3 Road Network Information

Larger triggers were added to cover whole parts of the road network, i.e. a trigger covering a whole lane, or a trigger spanning a whole intersection. This is shown in the simulator screenshot of figure 5.2. Upon a car entering or exiting such a trigger (i.e. colliding with one of them), a lane change event is emitted . By having access to information about which road segment a car has most recently exited or entered, the system is able to keep track of the time frame where a car is driving in a particular road segment.

Chapter 6

Proof of Concept Implementation

This chapter documents the architecture of our proof of concept system implementation. While this documentation is useful for anyone wanting to understand the working of the proof of concept system, it is also included as the system will be further developed in the future to a functional prototype. Therefore such extensive documentation helps new developers get a good grasp of the working of the current implementation, so they are able to extend and improve the current proof of concept system. Most of the architectural information of this chapter is based on the material of Len Bass (2012).

First of all how we have resolved the shortcomings of the simulator data is described.

Secondly, the set of requirements, both functional and non-functional, of the proposed system will be given.

Afterward, the requirements having a significant impact on the architecture of the system will be identified along with an explanation of why and how. These are so-called architecturally significant requirements.

Important to this elicitation are quality attributes, which are defined as being a measurable and testable property of a system that is used to indicate how well the system satisfies the needs of its stakeholders. It defines the "goodness" of a product along some dimension of interest to a stakeholder. The most relevant quality attributes for this system will be identified, based on the architecturally significant requirements. Next, the set of architectural tactics (techniques to

help achieve a certain quality attribute) utilized for each quality attribute will be detailed.

All architectural patterns (techniques to either help with architecturally significant requirements or the most important quality attributes for this system) utilized within the architecture are listed along with an explanation of where our implementation will use each one.

Next comes a section with architectural views, showcasing the proof of concept system from two viewpoints. Again, this is a proof of concept implementation of our system, not the full implementation of the proposed system. First is the logical view, showing the organization of modules and classes within the system, which helps document the implementation. Next comes process views explaining essential process flows, and the sequence of calls within the implementation. There is also a section containing the potential inconsistencies between the views and the actual implementation for completeness sake.

A section detailing the implementation logic of the more critical agents to our proof of concept system is also included.

Finally comes a rationale behind the overall architecture, relating implementation decisions back to the requirements initially identified and the architectural drivers.

6.1 Resolving Simulator Shortcomings

First of all, for our system to be able to read the information logged from the simulator, an XML parser had to be implemented. This parser needs to convert the data of the XML file to an internal state representation within our system implementation.

Our system adds speed and acceleration vectors for all other cars. This is done through simple interpolation across the previous timesteps. The speed interpolation uses the data actually coming in from the XML file (position data), while the acceleration bases itself on the interpolated speed values computed internally. More accurate values of acceleration and speed would be preferable, yet this will suffice for a proof of concept implementation. Together, these two vectors allow the projection of future positions of other cars. This projection is crucial to the situation awareness framework which again is the way we have decided to explain driver actions. Refer to chapters 2, 3 and 4 for a more in-depth explanation of this.

As mentioned in chapter 5, information on intended paths of other cars contains valuable information for reasoning about having to yield or not. Unfortunately, indicator light information from other cars is not included in the XML file. The solution used in our implementation is propagating path information back after a car has driven through the intersection. This knowledge is added to all timesteps where the car is approaching said intersection. This propagation backward in time is fine as long as the system is used for post-processing of a drive, but for real-time feedback this it is required.
Similarly to how path information propagates backward, road signs passed are also propagated backward. The XML log file has no information about which signs one is approaching, but it is highly relevant to know this ahead of time. Therefore, the information of a yield sign being passed at the end of a lane is propagated back in time for when a car is approaching the intersection from this lane.

The information on road markings has been omitted for the implementation of our proof of concept system. As the scope is restricted to reasoning about speeding and having to yield, road markings do not add any new necessary information for our system to reason with. However, this is something that should be added in the future, once the scope is widened.

6.2 System Requirements

This section lists both the functional and non-functional requirements identified for our system. Functional requirements represent the required functionality of a system, while the non-functional requirements deal with qualitative aspects of implementation. For our proposed solution, the set of functional requirements and non-functional requirements have been elicited. The results are found in table 6.1 and table 6.2.

Table 6.1: Non-Functional Requirements for the implementation of our Proposed Solution

ID	**Requirement**
NFR1	The system should be as modular as possible, meaning any independent functionality should be extracted to its own sub-component.
NFR2	The cost of modifying any sub-component of the system should be as low as possible. .
NFR3	The system implementation shall support generating feedback in near real-time, after receiving and reading incoming simulator data.
NFR4	The feedback produced by the system shall be explainable, so people without an understanding of the inner workings of the system can understand the feedback.
NFR5	The system reasoning shall be deterministic.
NFR6	The system shall be able to appropriately time when it gives feedback.

Table 6.2: Functional Requirements for the implementation of our Proposed Solution

ID	Requirement
FR1	The system shall be able to both represent and reason with traffic rules.
FR2	The system shall be able to both represent and reason with "best practice" of driving, where "best practices" are defined by . domain experts (driving instructors)
FR3	The system shall be able to represent the context of a situation internally, where the context contains relevant information for reasoning with traffic rules and best practices.
FR4	The system shall read time series data provided in a given format by the Way simulator.
FR5	The system shall be able to represent a situation internally.
FR6	The system must maintain the temporal aspect of the data coming from the simulator.
FR7	The internal representation of situation, context and the temporal aspects shall be saved to a file for each timestep.
FR8	The system should be able to reason with a representation of the situation, context and temporal aspects loaded from a stored timestep file.
FR9	The system shall be able to reason with the data read from the simulator.
FR10	The system shall produce feedback when the student makes a driving mistake, being a violation of traffic rules or best practices.
FR11	The system shall be able to provide justification along with the feedback.
FR12	The feedback produced by the system must reference the given violation that made the system produce it.
FR13	The system shall support adding domain-specific knowledge for finding and adding the missing information.
FR14	The system shall be able to find or add missing information to the situation, or it's context, based on its domain-specific knowledge.

6.3 Architectural Drivers and Architecturally Significant Requirements

This section will identify the drivers of our architecture, i.e. what shapes it. These are defined from the set of architecturally significant requirements, and the most important quality attributes inferred from these requirements.

6.3.1 Architecturally Significant Requirements

The most significant requirements, both functional and non-functional, identified in tables 6.1 and 6.2, are listed below, along with an explanation for why they have a significant impact on the architecture.

Table 6.3: Architecturally Significant Requirements

ID	Requirement
NFR1	Having the system be split into parts which interact through interfaces and/or intermediaries, impacts the architecture
NFR2	Low modification costs require the system to be as modifiable as possible. Thus the system requires layers of abstraction, hiding behind interfaces, and mechanisms of indirection which again impacts the architecture.
NFR3	Generating feedback in near real-time puts significant requirements on the performance of the system, and therefore also the architecture.

6.3.2 Architectural Drivers

Modifiability

Modifiability concerns the cost and ease of making modifications to a system.

As both NFR1 and NFR2 almost directly map to the definition of modifiability, this will be the main architectural driver of our system. Being able to replace parts of components, or even full modules, with as low of a cost as possible is valuable. Both the representation of the state received from the simulator, the internal state within the system, or any of the components responsible for reasoning, would be subject to change in the future.

Therefore, all agents should be easily modifiable and making new ones should require little effort.

Performance

Performance is concerned with managing system resources in a particular phase of demand to achieve acceptable timing behavior.

For our system, this will be measured as the time the system requires to reason about a certain timestep of the simulator. The time required should be as low as possible, so the system is able to provide close to real-time feedback to drivers in the simulator. This requirement is expressed by NFR3, which is the last of the architecturally significant requirements of table 6.3.

6.4 Architectural Tactics

This section describes which architectural tactics are being used to meet the architectural driving quality attributes of modifiability and performance identified in the previous section. Again, tactics are techniques that either help with architecturally significant requirements or a system's most important quality attributes.

6.4.1 Modifiability Tactics

Modifiability tactics are being used to reduce the cost of changes made to a system. The tactics that are being implemented help us create a system that is able to accommodate changes with ease. The two main concepts for achieving high modifiability are coupling and cohesion. Our main goal should be to reduce coupling and maximize cohesion in a modular design. In other words; maximizing the cohesion in each module and minimizing the coupling between modules.

Cohesion and coupling can be defined through the concept of responsibilities. Responsibility is an action, a decision, or knowledge that is being preserved by a system or an element of the system. Coupling is the strength of the relationship between responsibilities. By knowing the strength of the relationship between responsibilities, the likelihood of propagating changes can be calculated. To reduce coupling we can either reduce the relationship between modules or maximize the relationships between elements in the same module. Maximizing the relationships between elements in the same module is defined as cohesion. Next, we will define some of the tactics to be used for maximizing cohesion and reducing coupling.

Cohesion

To increase cohesion, we need to get the strongest possible relationship between responsibilities in one module. This can involve moving responsibilities between modules, or splitting responsibilities. By moving or splitting responsibilities we reduce the likelihood of side-effects to other responsibilities in the original module. We will use the following tactics to move or split responsibilities:

1. **Maintaining semantic coherence**
   Collocating responsibilities that are affected by a single modification such that the cost of modifying the new "collocated" module is less than modifying the original one.

2. **Abstracting common services**
   If two or more modules essentially provide a variant of the same service, this service could be implemented in a single module in a more general form. By doing this, any modification to the service would only need to occur once. This tactic is often used when refactoring code.

The implemented architecture has a high cohesion in most of its components. This is especially true for everything related to the blackboard. Each agent reading and modifying the blackboard state has a single responsibility, i.e. high semantic coherence. This is also true for most of the other components. Furthermore, abstractions have been implemented wherever possible. For instance through having a single relation factory for all relations that can be made.

The least cohesive class is the class responsible for reading the XML coming in from the simulator, as well as the controller, as they both tie the application together and hence their responsibilities differ a bit.

Coupling

To reduce coupling we can either reduce the relationship between modules or maximize the relationships between elements in the same module. Some of the best ways to reduce coupling are to generalize, abstract or to break dependencies by adding layers. We have used the following tactics to achieve these effects:

1. **Encapsulation**
   By introducing an explicit interface for a module, the chance of changes propagating to a module is reduced. In other words, the strength of coupling between the modules is reduced by placing an interface working as an API in front of the original module. By doing this one enforces information hiding and limit the ways other modules can interact with the original module.

2. **Intermediates**
   Breaking dependencies by implementing an intermediate layer between

the modules. Intermediates remove the knowledge modules have of each other, and creates a standard way of interaction for all modules requiring the same service from each other.

In the proof of concept system implementation, there is very low coupling between components in general, achieved through the use of these tactics. Both through the use of interfaces and general object-oriented programming practices, but also the tactics of encapsulation and intermediates. For instance, the state is accessed through an intermediary defined in an interface. Another measure has been the introduction of a service locator. Furthermore, the agents accessing the blackboard are all accessed through an intermediary, which implements an interface, meaning the controller can handle them agnostically and has no direct knowledge to the workings of each individual agent. Much of the inter-module communication is through interfaces, and most dependencies are injected through constructors for inversion of control.

6.4.2 Performance Tactics

Performance tactics are implemented to make sure a system is being able to meet all it's timing requirements. Being able to meet timing requirements means being able to generate a response to an event arriving at the system within a given time constraint.
"Events" are the triggers for the computation that needs to happen before a response is sent, and it can be single or a stream of requests.
"Latency" is the time from the event has arrived until a response has been generated.

There are three main categories of ways to handle system resources: demand, management, and arbitration.
Demand is characterized by the frequency of events and how much resources each event consumes. Management related to how resources are managed, and arbitration to how resources are scheduled. The tactics used in our implementation for performance mostly revolve around management, as the demand cannot be changed, and arbitration is not relevant for an application running a single task on a single computer in this case.

In terms of resource management, our main tactic has been to use concurrency. This will reduce the time to process events by processing them in parallel. You can either process similar streams of events on the same threads or by creating new threads to process different types of activities. An example from our application is how the scheduler runs in a separate thread from the reasoning part of the system. Furthermore, some degree of caching has been utilized, as state and values from earlier computations are kept stored within the state. This caching frees up resources from computations already having been done,

thus increasing the performance potential.

6.5 Architectural Patterns

The architecture will combine several different architectural patterns at different abstraction levels, to fulfill the requirements dictated by the quality attributes and architecturally significant attributes. Software design patterns, or architectural patterns, provide general solutions to problems that commonly occur in software development. All design patterns used in our system are described in this section.

There are two types of patterns described below: First of all *creational* design patterns have been utilized, which deal with common mechanisms used when creating objects. Secondly, we have also used *behavioral* patterns. These are patterns that identify and deal with mechanisms for common ways of communication between objects.

6.5.1 Blackboard Pattern

The blackboard pattern is a behavioral design pattern revolving around a common resource, the blackboard, which is read and modified by knowledge sources. Their access to the blackboard is handled by a controller that among other things takes the priority of knowledge sources into account. More details on this pattern are found in chapter 2.
Within our system, a blackboard is used to represent the state of our system, which agents modify as more information is available. For instance, the agents responsible for computing acceleration and possible collision depend on the speed computed by the speed interpolation agent. A controller orchestrates their requested calls based on priority and satisfied conditions. The agents all implement a knowledge source interface, facilitating for modifiability in the case of replacing an agent implementation or adding new ones.

6.5.2 Singleton

The singleton pattern is a creational design pattern, which should be used when there is a need to guarantee that only one object of a certain class will exist at any time. This pattern will be utilized for application-wide configuration, settings, and logging. This may be considered an anti-pattern by some, and usually, it does not contribute much to modifiability, but it does simplify some aspects of the implementation.
In our system, there are several classes only requiring to be instantiated a single time, where the internal state is shared in some sense among the objects of

the system. This is true for the blackboard as well as the factories as they cache already instantiated relations they want to keep track of. By having these factory objects be singletons and use caching, we help the performance aspect of the system. Furthermore, the intermediaries in front of frequently referred to objects like the state are also singletons.

6.5.3 Factory Pattern

The factory pattern is another creational design pattern can be used to create objects based on configuration or external state, without the knowledge of the underlying class.

This pattern will be used to manage and instantiate various modifiable relations of the simulator, differing in which types of entities they relate, and what types of values the relations hold. This pattern will help achieve modifiability, due to new types of relations being easily added to the system.

6.5.4 Observer Pattern

The observer or publish-subscribe pattern is a behavioral pattern that allows an object to notify observers of changes in its state without depending explicitly on the observers. The observer pattern will be used as part of the blackboard implementation to allow systems to observe changes in the state, so the systems can notify listeners to changes in conditions. This will help achieve modifiability, as new agents can be added almost seamlessly through the implementation of our subscriber interface.
In our implementation, this pattern is used to notify agents of when any of the conditions they require to run are updated, for the case where they have already run, but now improved or updated information is available.

6.6 Views

This section contains the included architectural views of our system, with short reasoning for why they are included, as well as a thorough explanation of the view to inform the reader of how our system is implemented.

6.6.1 Logical View

This view documents what features the proof of concept system will provide, and is of value to both developers and end users, as they can see which parts the system is built up of. It is documented as a class diagram in figure 6.1 below,

which illustrates which parts of the system are used by other parts. It includes all the key components and abstractions, along with a thorough explanation of the high-level workings of the system.

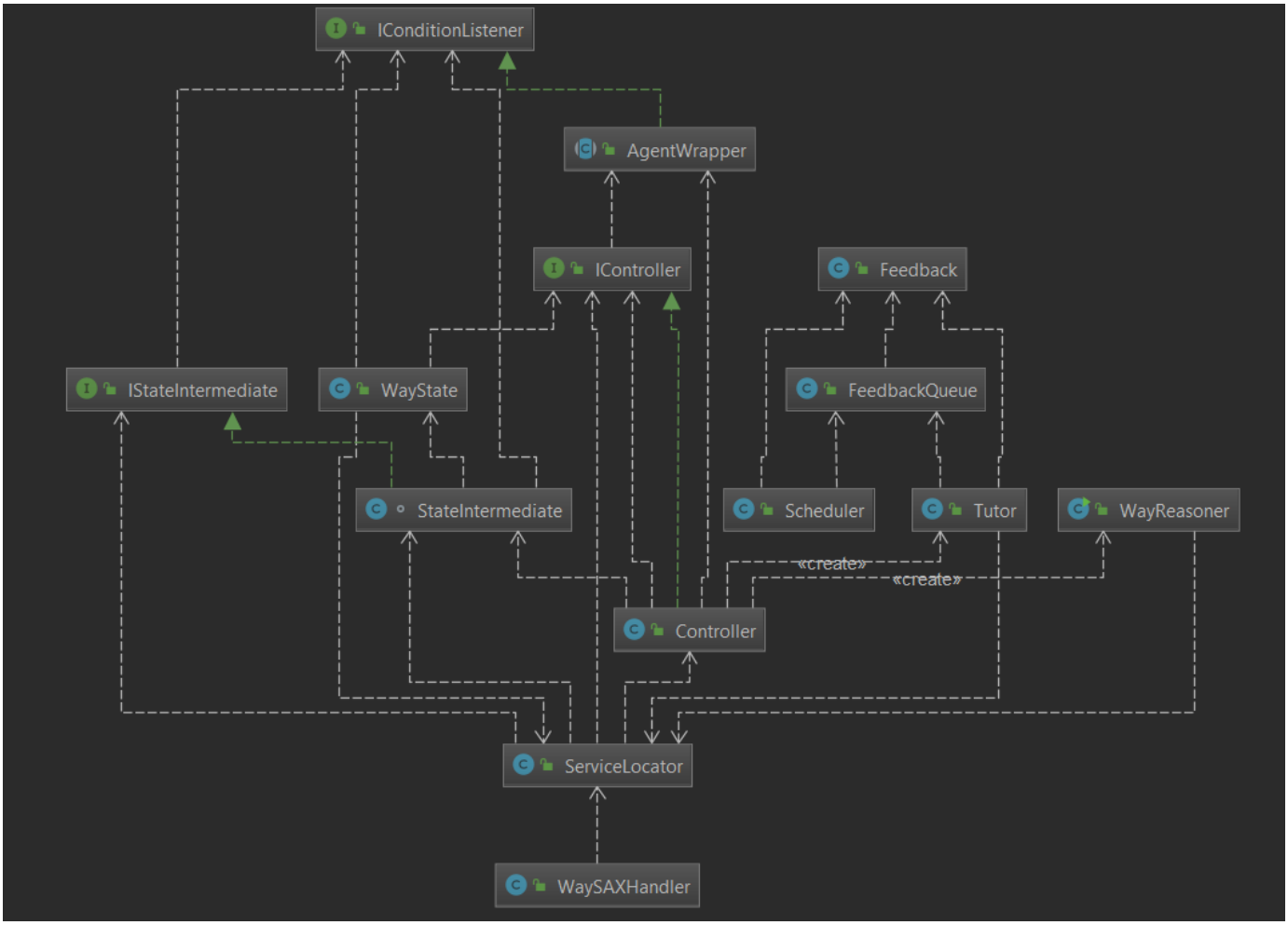


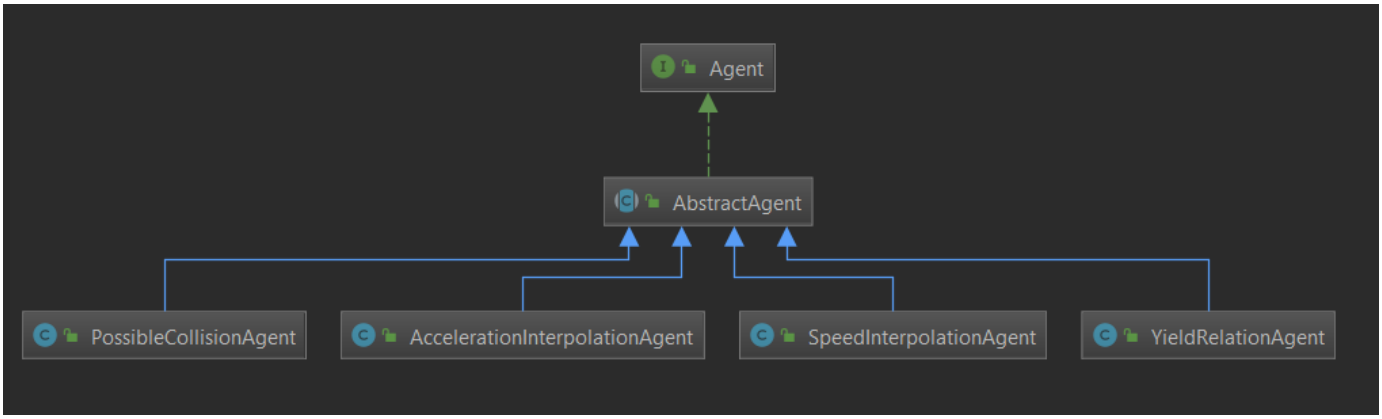
Figure 6.1: The combined class diagram of the proof of concept implementation of our virtual driving instructor system.

The class diagram of figure 6.1 illustrates how our system is built up. At the bottom of the diagram is the class responsible for parsing the XML coming in from the simulator, WaySAXHandler. For each timestep this class reads from the simulator, it adds information to the state for that timestep through the StateIntermediate, which it accesses through the ServiceLocator. This StateIntermediate has a reference to the WayState singleton, which in our system acts

as the blackboard of the blackboard pattern. Furthermore, the methods of this intermediary are specified in the IStateIntermediate interface.

After updating the state, WaySAXHandler notifies the Controller class that the state has been updated. The Controller contains a set of AgentWrappers, each keeping track of the conditions each Agent requires. Furthermore, these AgentWrappers return the priority of the agent contained within them depending on whether or not they have run with the current set of blackboard conditions available. This allows the system to handle the case where conditions could be updated several times during the same timestep, or for the case where previous condition values are used. The agents of our system map almost directly to the abstraction of knowledge sources in the blackboard pattern.

The AgentWrapper is an abstract class, with a method for computing priority being abstract such that new ways of computing priority can be implemented in the future. As of right now the NaivePriorityAgentWrapper always returns it's base priority when the Agent of the AgentWrapper should be run, while OptionalConditionPriorityAgentWrapper returns it's base priority except when any of it's prioritized conditions are present (they can run without these, but prefer to run with them). The RerunnableAgentWrapper allows an agent to run several times, while the other wrappers simply run once for each timestep. This is useful for the case where information needs to propagate backward, like new knowledge of a car's intended path through an intersection. All wrappers extend the abstract class AgentWrapper. The agent package has been omitted from the class diagram of figure 6.1 for clarity, but is given in figure 6.2.



f

Figure 6.2: The class diagram of the agent-package of our proof of concept implementation of our virtual driving instructor system.

When the Controller is notified of an update to the state, it looks at all satisfied conditions, filtering all AgentWrappers based on whether or not their agent's conditions are all satisfied. The satisfied AgentWrapper with the highest priority is then run, possibly updating the state and/or satisfied conditions. As long as the set of satisfied conditions is updated, or there are still satisfied AgentWrappers whose agents are not run, the Controller keeps picking the satisfied Agent whose AgentWrapper returns the highest priority and runs this Agent. AgentWrappers implement the IConditionListener interface, meaning through the observer pattern they are listening to changes in the conditions of a tick from the State. This is particularly useful for Agents that could improve the utility of them running due to this update in a Condition. A particular example of this would be the PossibleCollisionAgent, which first could run without taking acceleration into account, but after being notified of acceleration being added by the AccelerationInterpolationAgent, it would run again with improved computation results which are indicated by an increase in priority.

When all Agents have done their job, the Controller allows the WayReasoner to access the state, which again is done through the StateIntermediate. This reasoner fills an OWL ontology based on the state of the current timestep, using a reasoning engine to infer any violations done. Any violations inferred are added to the current timestep. It is important to note that within this proof of concept system implementation, post-processing of a drive is performed. Therefore any information propagating backward tells the controller that new information is available for all the effected timesteps. It is only once no new information is new to any timestep that the reasoner actually starts to infer violations across the timesteps.

Finally, the Controller makes a call to the Tutor, which looks for violations in any tick. If there are any, it uses Feedback to generate feedback based on these violations and adding it to a FeedbackQueue.

In a separate thread of the application, the Scheduler runs. This Scheduler continuously polls the FeedbackQueue for Feedback and consumes any Feedback added. When consuming Feedback from the queue, the Scheduler provides this to the user at an appropriate time. Currently, it is simply outputted to the console as textual feedback upon being consumed, with a justification and explanation of why the action was a violation in the current situation.

6.6.2 Process View

This view describes the processes and workflows of our system, as well as the communication between objects. It mainly provides value for developers looking to maintain or extend the system, but potentially other stakeholders needing an explanation for how the system works as well.

Below, in figure 6.3, is an overview of the call sequence whenever the XML parser updates the state and notifies the controller of this. This diagram documents the flow of the system as a whole, on a high level, with some simplifications.

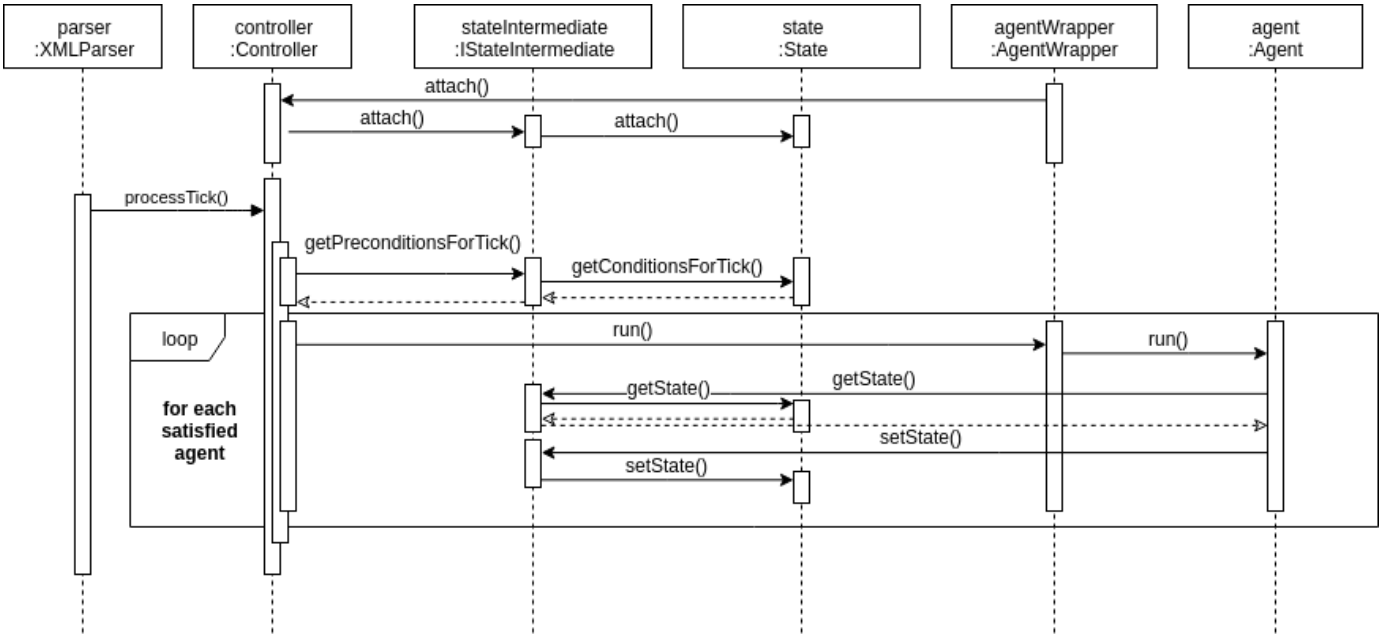


Figure 6.3: Process view showing the overall flow of the proof of concept virtual driving instructor system through a sequence diagram, when initializing a timestep coming in as XML from the simulator.

The sequence diagram in figure 6.3 contains the major workflow within our system, showing how the information propagates throughout the system following the blackboard pattern implementation.

For each timestep, the XMLParser calls the controller to process the tick. The controller will loop over all the AgentWrappers that have been attached to it. First up it retrieves the currently satisfied conditions from the State in the getPreconditionsForTick()-call, which is done through the implementation of the IStateIntermediate interface. Internally, the controller gets all of the priorities for each of the AgentWrappers based on the current set of conditions. A simplification is made to the diagram by avoiding this flow, for clarity.

After getting the satisfied AgentWrapper with the highest priority, this Agent is run by a call from its AgentWrapper. Each call to an Agent requires parts of

the state, represented by the simplified call to getState() (internally it retrieves smaller parts of the state, rather than the entire state). This call again uses the intermediate for access to the state.

Afterward, the agent performs some logic internally and updates the state by a call, simplified for clarity, setState() (internally smaller parts of the state are modified). After the state has been updated, the controller retrieves the new set of conditions.

Whenever any updates are made to conditions, the State will notify any IConditionListeners, marking the timestep as updated. The notification is not shown in this diagram, but in that of figure 6.4.

As long as there are satisfied AgentWrappers, with the set of conditions initially retrieved after the call from the XML parser, where the represented Agent can add information to the state, the system loops over the process described above until the state is filled by each agent of the system.

When the XML parser has initialized this process for all timesteps of the incoming XML, the system prompts another pass across all the timesteps. This is due to the possibility of information having propagated backward (like paths through an intersection). The same process as above is repeated for each updated timestep. In the end, all timesteps contain all the information the proof of concept system is able to add to them. For the future real-time system, rather than doing a pass across all timesteps initially, then another pass as long as any timesteps are marked as updated, the system would process each timestep as it arrives, and keep on adding information as IConditionListeners are notified.

After this, the reasoner retrieves the entire state for each tick, reasons with the information and adds any potential violations. In the future, when the system will not be run post-driving, the reasoner will, of course, be run after each timestep has had all available information be added to it. Again for the future system, the reasoner would reason for each tick as soon as it contains enough information, before moving on to the next one.

After the reasoner has added violations, a tutor will generate feedback and put this into a queue.

A separate thread, the scheduler, consumes feedback from the aforementioned queue, providing it as textual feedback through the console. While the threading is not necessary for the sequential post-processing done in this proof of concept, it will provide useful for the scenario of real-time feedback in the future version

of the system.
The reasoner, tutor, feedback queue, and scheduler thread are all intentionally left out of figure 6.3, for clarity's sake.

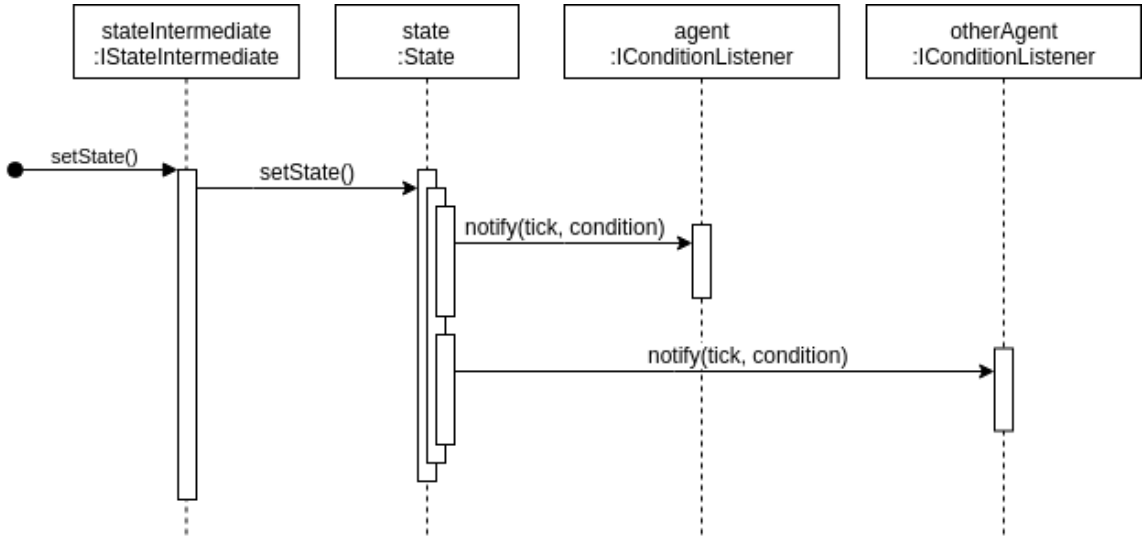


Figure 6.4: Process view of the observer pattern as used in our proof of concept implementation, documented as a sequence diagram.

Figure 6.4 illustrates the flow of calls that are performed whenever the state is updated, meaning that a new condition is satisfied. This was left out of figure 6.3, for the sake of clarity, but is a process that is included in this flow.

Initially, AgentWrapers are attached to the state, as shown in figure 6.3. These implement the IConditionListener interface. Together these two classes implement the observer pattern, as all listeners are notified whenever the state is updated.

As shown in 6.4 upon having the state set, both the "agent" and "otherAgent" instances are notified of this condition update, along with which tick the update holds for. In the real system, there are of course a lot more agents than two. Furthermore, these are attached as listeners to state upon the system being initialized. This is done with the help of the Controller.

6.6.3 Inconsistencies Between Views

The sequence diagram of figure 6.3 contains inconsistencies, as the names of methods, classes, and instances do not correspond to the ones used internally in

the system. The rationale behind this is keeping things clear, and the concern of parameters taking up too much space within the figure.

The same holds for the sequence diagram of figure 6.4, parameters, names of methods, classes, and instances do not correspond to the ones used internally in the system. Specifically, rather than notifying the actual agents, the agent wrappers are notified. Again this simplification is done for the purpose of keeping the pattern as clear as possible. Furthermore, the view of 6.4 does not show the process of attaching the listeners to the observable (State), this is commented on in the corresponding textual description of the view.

Though the mismatches might confuse a reader, the rationale behind this decision is that the simplification of names and calls helps with understanding the pattern and call sequence within our system, which we consider more important than a one-to-one mapping in this case.

6.7 Agent Documentation

This section document the implementation details of the most critical agent implementations for our proof of concept system. Each agent adds significant information to the scope of reasoning about violating ones obligation to yield or speeding.
Again, for each timestep within a drive, each of the agents will attempt to run once all required information has been added to the situation description, computing and adding additional, potentially more abstract information to the situation description. Finally, the reasoner will use this full description to reason about any possible violations.

6.7.1 Speed and Acceleration Interpolation Agents

Due to both speed and acceleration data missing from the received data from the driving simulator we are basing ourselves on, these two agents were implemented to add crucial information to our situation description.
Both agents employ simple interpolation to compute the speed or acceleration of a car, averaging the corresponding values for a given set of timesteps. More precisely, our system uses the estimates of
$speedInterpolation_t = |position_{t-1} - position_t|/|t_1 - t|$ and similarly
$accelerationInterpolation_t = |speed_{t-1} - speed_t|/|t_1 - t|$, both across an interpolation window, averaging the values between each timestep.
This is done for each car present during a timestep, and thus along with their current positions allows us to do prediction about where they will be positioned in the future.

Preconditions required for the speed interpolation agents are simply the positions of other cars, being the raw data coming in from the driving simulator. For the acceleration interpolation agent, however, the speed data is required, hence this agent depends on the information of the speed interpolation agent. This dependency is expressed through the blackboard conditions declared as required in the corresponding agent's wrappers, nicely illustrating the fit of the blackboard pattern to our system.

6.7.2 Possible Collision Agent

Our framework used for describing the process of driving, situation awareness, contains a step of projecting the future state. This is exactly what functionality the possible collision agent implements. Based on the current speed, position, and acceleration (given this information is available), the agent projects where cars will be in the future, and if at any point they will be colliding. If so, there is a possible collision.
However, there is a need to only add this relation whenever a collision is probable. Therefore a restriction to the time in the future has been added. For our proof of concept system, we say that if the time to collision is within twice the time required to both react and brake to a complete stop from the current situation, there is a possible collision.
The distance traveled while reacting is simply $d_r = (speed \cdot reactionTime)/3.6$, where we divide by 3.6 as speed is given in kph. For the distance traveled by braking, the distance is computed as $d_b = speed^2/(250 \cdot f)$, with f being the friction coefficient, estimated to 0.8 for dry asphalt. The sum of these two distances is used to compute the time window in which we consider collision probable.
This agent adds a possible collision relation between any two cars where their position at some time restricted by the time interval imposed by the time it takes to travel twice the computed braking distance, will be within one car length of each other.

6.7.3 Yield Relation Agent

The scope we have restricted our virtual driving instructor system to be able to reason about concerns both violating obligations to yield, as well as speeding. The concern of this agent is to indicate which cars have to yield for others, from the perspective of the ego, being the driver in the simulator.
For our proof of concept system, this agent assumes the right-hand rule applying. Furthermore, it relies on information about one's context to tell whether or not a right of way sign or yield sign applies to the driver.
Based on the relative angles of incoming lanes to an intersection, the right-hand rule, as well as currently applying road sign rules imposed, the agent will add a relation between their driver and other cars it has to yield for.
Furthermore, this agent also looks at the path other vehicles take through an

intersection, to tell whether or not the ego driver has to yield for their path through an intersection. This is done through a special back-propagation of information within the system, due to a lack of this information as no indicator light data is currently coming in from the simulator. In the case of our proof of concept system doing post-analysis of drives, this is fine, but for the proposed solution of chapter 4, real-time data indicating one's path through an intersection, like current lane or indicator lights, has to be present for this kind of sophisticated yield logic to be applied.

6.8 Architectural Rationale

The overall architecture has been built around the blackboard pattern, due to the tasks our system performs. There are several parts that all interact upon an internal representation of the current state of the simulator and the corresponding context. This is what is contained within the blackboard, and upon information being made available the different agents capturing having captured different domain knowledge can expand upon this information. The viewpoint of situation awareness on what the job of a virtual driving instructor is has been outlined in chapter 2, which supports the choice of such an architecture.

Furthermore, as the main architecturally significant requirements concern modifiability, separating responsibilities into knowledge sources (the name of "agents" within the original blackboard pattern) provides high modifiability. The implementation of an agent can simply be modified without having a large effect on the rest of the system. Furthermore, layers of indirection, like through encapsulation, has been introduced through interfaces and intermediaries.

The other architecturally significant requirement of performance has also been addressed, though not to the same degree. The system architecture supports a concurrent implementation in the future, where agents could be run in separate threads, as long as the blackboard has its access atomically secured, and a mechanism to stop reasoning about one timestep and move to the next. Furthermore, caching mechanisms, the singleton pattern, as well as using the factory pattern have been added as a measure to reduce the number of resources required.

Overall the architecture is understandable for new stakeholders, as known technologies and patterns lay the foundation for it. As all architecturally significant requirements of table 6.3 have been addressed, the performance of the system should be up to par with what is required.

Chapter 7

Experiments and Results

In this chapter, we describe the two experiments conducted to test how the proposed solution answers the research questions.

7.1 Experimental Plan

This section details the design of the experiments that were conducted.

7.1.1 Overview

To evaluate the proof of concept, two experiments were conducted. The purpose of the experiments was to evaluate the performance of the system across everyday situations in which yielding applies. Each experiment, therefore, included a subset of scenarios that can occur in a T-intersection or a 4-way intersection. To begin with, a T-intersection serves as a basic intersection where rules of yielding apply, then, to show generalization in the behavior of the proof of concept, the T-intersection is expanded to a 4-way intersection. Furthermore, a 4-way intersection also opens up for new scenarios not possible within a regular T-intersection.

The first experiment was conducted on data generated from testing on an ordinary laptop with the authors as drivers. The results of the first experiment will be based on the Virtual Driving Instructor's evaluation of speeding and yielding behavior and compared to the authors own domain knowledge. The second experiment used data generated by test subjects driving in the Way AS simulator. The results of the second experiment will be based on the Virtual Driving Instructor's evaluation of speeding and yielding behavior and compared to the

evaluation done by a professional driving instructor. The rationale behind the two experiments follows in the subsequent subsections.

7.1.2 Rationale Experiment 1

The purpose of the first experiment is to test a wide range of different scenarios. This means being able to force out both correct and incorrect driving behavior for all types of scenarios through careful configuration, and thereby validating the system's robustness. It serves as a testing ground for the proof of concept to make sure it handles the different situations. The data will be produced by the developers of the proof of concept on a laptop to ensure variety in data and to include data expected to be difficult for the system. It allows observing if expected representations of situation and context are produced for given scenarios, as the developers are able to rapidly configure, drive through and analyze scenarios. Furthermore, it allows seeing if expected information is deduced.

7.1.3 Rationale Experiment 2

The purpose of the second experiment is to test the proof of concept on data from authentic driving. The intention is to provide a larger scale, realistic testing of the system. Therefore, the experiment will be conducted with data from the actual simulator. While the first experiment serves to test edge cases, and check for robustness, the second experiment is more geared toward typical use-cases and evaluates the performance in normal use. Test subjects without knowledge of the internal structure of the proof of concept, and who have to solve the scenarios as they develop, serve as good examples of real users. Furthermore, the more immersive experience of driving in a driving simulator is expected to produce more realistic results, in comparison to those produced from a laptop in experiment 1. Developer knowledge was required to evaluate the results and internal representations of the first experiment, but in the second experiment, professional driving instructors will serve as an evaluation of the results.

7.1.4 Data

All data was gathered from the driving simulator built by Way AS in Unity, described in chapter 5. Data for the first experiment was produced by running the virtual simulator on a laptop. The reason is that it allowed for faster production of data and to reduce the load on Way AS' physical simulators, as stated above. Data for the second experiment was produced by different test subjects driving in Way AS' physical simulators. These people were instructed to drive normally in the simulator.
In both cases, the data output will be the same, an XML file containing the

information from their drive within the simulator. These files will be analyzed independent from the simulator, and any feedback is provided textually in a terminal window on the computer running the Virtual Driving Instructor proof of concept implementation.

7.2 Experimental Setup

This section documents the setup of the experiments conducted to validate the proof of concept system implementation. The different scenarios of the T-intersection and the 4-way intersection are textually described and illustrated. A small rationale behind each individual scenario of the second experiment is given, while a general rationale is provided for the different configurations utilized in the first experiment.

7.2.1 Experiment 1

As stated in the previous section, the experiments of experiment 1 were to be run on a developer laptop. Contrary to in the experiments of experiment 2, each scenario was to be run separately, meaning the virtual world of the simulator is reset between each scenario.
As the rationale behind experiment 1 is testing a wide variety of situations, to check the robustness of the reasoning logic, a decision was made to always let the Ego approach the intersection from the left lane, while the other car approaches from the middle lane. Fixing these variables in the set of possible configurations was done to highlight the other aspects of the scenarios, like the intended path and road signs applying. An illustration of the possible initial positions is given in figure 7.1.
The scenarios to be tested consist of 17 possible legal configurations of the Ego turning to the right or driving straight on, and the other car turning left or right if coming from the middle lane. The combinations also include configurations of the Ego and other car either having right of way, having to yield, or there being no road signs present. Furthermore, where possible, an attempt is made to drive incorrect in the situation as to have the system generate the expected feedback. The expected feedback is given in table 7.1 below.

Ego Destination	Other Path	Ego Yield	Other Yield	Correct	Expected Violation Generation
Right	Middle to Left	RH	RH	Yes	No
Right	Middle to Left	RH	RH	No	Yes
Right	Middle to Left	ROW	Y	Yes	No
Right	Middle to Left	Y	ROW	Yes	No
Right	Middle to Left	Y	ROW	No	Yes
Right	Middle to Right	RH	RH	Yes	No
Right	Middle to Right	Y	ROW	Yes	No
Right	Middle to Right	Y	ROW	No	Yes
Straight	Middle to Left	RH	RH	Yes	No
Straight	Middle to Left	RH	RH	No	Yes
Straight	Middle to Left	Y	ROW	Yes	No
Straight	Middle to Left	Y	ROW	No	Yes
Straight	Middle to Right	RH	RH	Yes	No
Straight	Middle to Right	RH	RH	No	Yes
Straight	Middle to Right	Y	ROW	Yes	No
Straight	Middle to Right	Y	ROW	No	Yes

Table 7.1: The expected results of running the configurations of experiment 1 locally. The destination of the ego, the path of the other car, as well as their respective right of way privileges: RH = right-hand rule, ROW = right of way, Y = yield. Whether or not the scenario was driven in a correct manner is indicated, as well as whether or not a generated feedback is expected.

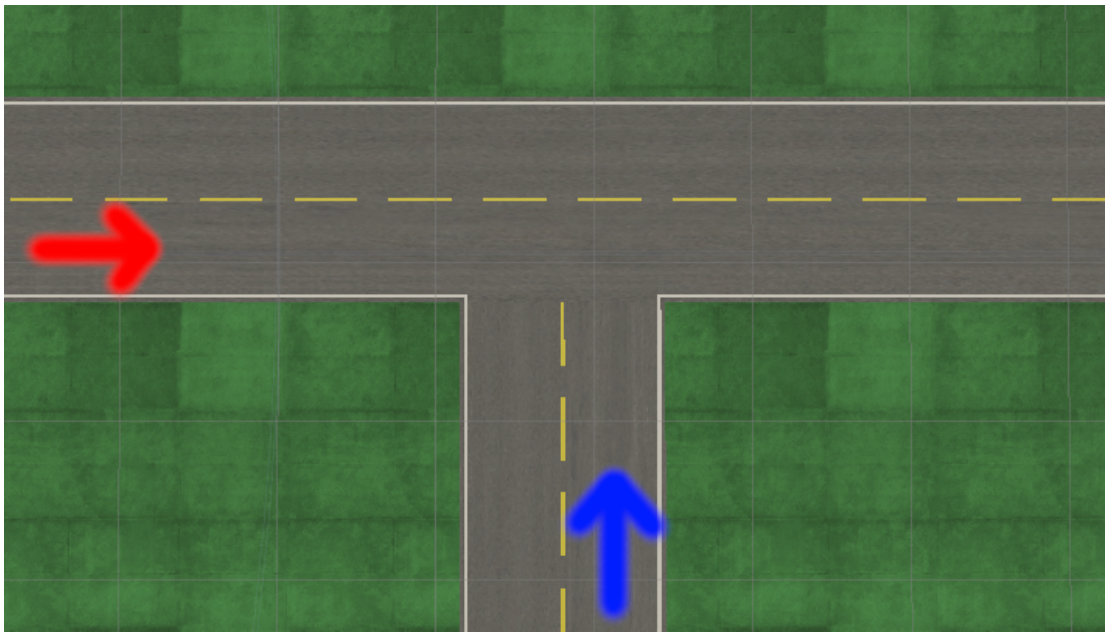


Figure 7.1: The start positions of the T-intersection scenarios tested in experiment 1. 17 possible legal paths through the intersection (i.e. no u-turns), were tested, with different legal combinations of road signs. The Ego always starts from the left, as indicated by the red arrow, while the other car starts from the middle as indicated by the blue arrow.

7.2.2 Experiment 2, T-Intersection

The scenarios for experiment 2 included six scenarios of driving in a T-intersection. The scenarios are displayed below in figure 7.2. A description of the T-intersection scenarios is given below. Each scenario is described textually, with the rules applying, as well as the rationale behind the given scenario.

Scenario 1 Here, the Ego approaches the intersection from the left lane, while the other car approaches from the bottom lane (seen from above). In this case, no road signs impose any yield-related traffic rules, hence the basic right-hand rule applies. This means that the Ego has to yield for the other car. The rationale behind this scenario is testing the system's capability of evaluating the basic right-hand rule.

Scenario 2 The second scenario has Ego approach from the right lane, while the other car approaches from the left lane. Again, no road signs impose any yield-related traffic rules so the basic right-hand rule applies. The path of the Ego has the other car be to the right of the Ego, hence Ego has to yield for the other car. The rationale behind this scenario is testing

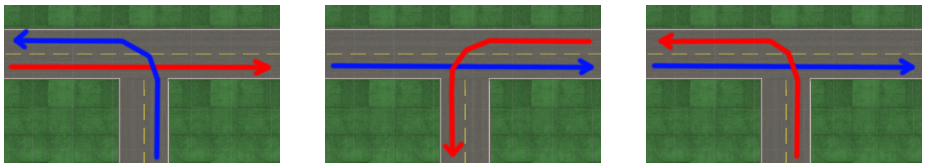
the system's capability of evaluating the right-hand rule in terms of paths through an intersection.

Scenario 3 In this scenario, Ego approaches the intersection from the bottom lane, while the other car approaches from the left. No road signs impose any yield-related traffic rules, meaning the right-hand rule applies. In this case, the Ego has the other car to the left, meaning it does not have to yield for it. The rationale behind this scenario is testing the system's capability of evaluating the basic right-hand rule by being aware of it not applying.

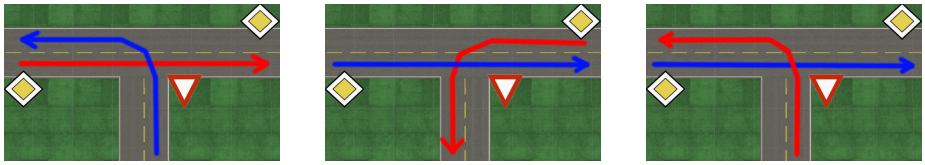
Scenario 4 This fourth scenario has the Ego approach the intersection from the left lane, while the other car approaches from the bottom lane, like in scenario 1. However, this time there are road signs imposing yield-related traffic laws. The left and right lanes have right of way, while the bottom lane has to yield for both of them. As the Ego vehicle has right of way, it should not have to yield for the other car. The rationale behind this scenario is testing the system's capability of evaluating right of way.

Scenario 5 This scenario has the Ego approach from the right lane, while the other car approaches from the left lane. The Ego vehicle is to turn to the left, meaning that the other car will be to the right of it in the intersection. Therefore, the Ego has to yield for the other car. The rationale behind this scenario is testing the system's capability of evaluating the right-hand rule although road sign rules related to yielding are present.

Scenario 6 The final T-intersection scenario has the Ego approach from the middle lane, with the other car approaching from the left. In this case, there is a yield sign applying to the Ego, meaning it has to yield for the other car. The rationale behind this scenario is testing the system's capability of evaluating having to yield when the yield sign is present.



(a) The T-intersection scenarios (1 through 3 from left to right) without any road signs.



(b) The T-intersection scenarios (4 through 6 from left to right) with road signs, the horizontal lanes having right of way, the vertical lane having to yield, as indicated by the road signs.

Figure 7.2: The set of T-intersection scenarios as seen from a top-down view within the simulator. The red line indicates the path of the Ego, while the blue line indicates the path of the other car.

7.2.3 Experiment 2, 4-way Intersection

The scenarios of experiment 2 included four scenarios when driving in a 4-way intersection. The scenarios are displayed below in figure 7.3. As for the T-intersection, a description of the scenarios is provided below. A notable difference between this and the previous set of scenarios is the addition of a fourth lane, as well the left and right lane now having a yield sign, while the bottom and top lanes have right of way.

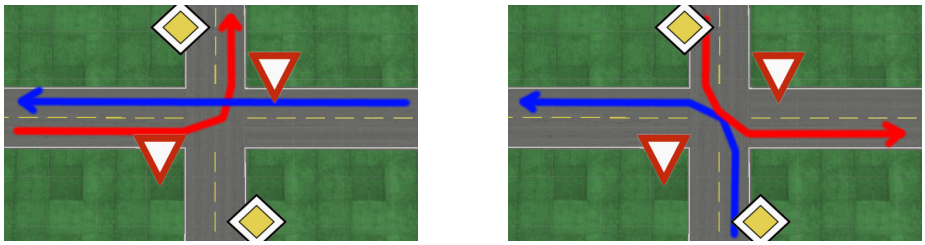
Scenario 7 The first scenario of the 4-way intersection has the Ego approach from the left with an intended path to the top lane, while the other car approaches from the right with an intended path to the left lane. In this case, as a having to yield sign is present in the lane of the Ego, the Ego must yield for the other car. The rationale behind this scenario is testing the system's capability of evaluating having to yield with signs present, as well as for the system to be able to handle the extended intersection.

Scenario 8 This scenario has the Ego approach from the top lane with an intended path to the right lane, and the other car approaching from the bottom lane with an intended path to the left lane. Furthermore, both of the cars have right of way when entering the intersection. In theory, both cars should be able to drive through the intersection without interfering with each other due to their paths. The rationale behind this scenario is testing the system's capability of evaluating reasoning related to right of

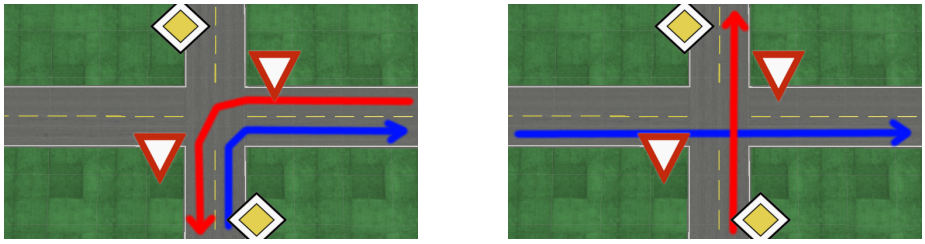
way, as well as the system being able to handle the extended intersection.

Scenario 9 In the third scenario of the 4-way intersection, the Ego approaches from the right lane with an intended path to the bottom lane, and the other car approaches from the right lane with an intended path to the bottom lane. Here the Ego has right of way, while the other car has to yield. The rationale behind this scenario is further testing the system's capability of evaluating reasoning related to right of way, as well as the system being able to handle the extended intersection.

Scenario 10 Finally, in the fourth scenario of the 4-way intersection, the Ego approaches from the bottom lane with an intended path to the top lane, and the other car approaches from the left lane with an intended path to the right. In this case, the Ego has right of way, while the other car has to yield. Again, the rationale behind this scenario is testing the system's capability of evaluating reasoning related to right of way, as well as the system being able to handle the extended intersection.



(a) The two first 4-way intersection scenarios, scenario 7 to the left and scenario 8 to the right.



(b) The two last 4-way intersection scenarios, scenario 9 to the left and scenario 10 to the right.

Figure 7.3: The set of 4-way intersection scenarios as seen from a top-down view within the simulator. The red line indicates the path of the Ego, while the blue line indicates the path of the other car. The horizontal lanes have right of way, while the vertical lanes have to yield, as indicated by the road signs

7.3 Experimental Results

This section documents general results seen in the testing of our proof of concept system, by looking at different scenarios of driving through an intersection. Both the results from experiment 1 and experiment 2 are provided below.

7.3.1 Experiment 1

The table of 7.2 documents whether a violation was generated or not for each of the configurations of the T-intersection that were run as acceptance tests in experiment 1.

Ego Destination	Other Path	Ego Yield	Other Yield	Correct	Violation Generated
Right	Middle to Left	RH	RH	Yes	No
Right	Middle to Left	RH	RH	No	**Yes**
Right	Middle to Left	ROW	Y	Yes	No
Right	Middle to Left	Y	ROW	Yes	No
Right	Middle to Left	Y	ROW	No	**Yes**
Right	Middle to Right	RH	RH	Yes	No
Right	Middle to Right	Y	ROW	Yes	No
Right	Middle to Right	Y	ROW	No	**Yes**
Straight	Middle to Left	RH	RH	Yes	No
Straight	Middle to Left	RH	RH	No	**Yes**
Straight	Middle to Left	Y	ROW	Yes	No
Straight	Middle to Left	Y	ROW	No	**Yes**
Straight	Middle to Right	RH	RH	Yes	No
Straight	Middle to Right	RH	RH	No	**Yes**
Straight	Middle to Right	Y	ROW	Yes	No
Straight	Middle to Right	Y	ROW	No	**Yes**

Table 7.2: The results of running the configurations of experiment 1 locally. The destination of the ego, the path of the other car, as well as their respective right of way privileges: RH = right-hand rule, ROW = right of way, Y = yield. Whether or not the scenario was driven in a correct manner is indicated, as well as whether or not the system generated a feedback. The scenarios where the system output a violation and it was expected have their text made bold.

As for speeding, this behavior was recognized by the proof of concept implementation several times throughout the scenarios. Each time the simulator log file values resulted in a computed speed for the Ego that was above the current speed limit in the Ego context, the system would recognize it being a speeding violation.

7.3.2 Experiment 2

General trends were observed across the scenarios of experiment 2. Concrete examples of these trends are illustrated through examples. These examples are described textually, as well as displayed in three-part pictures capturing how the scenario evolved over time.
The overall results are given in table 7.3 below. This table contains the total number of yield violations across all participants of experiment 2, per scenario. Unfortunately, we had to exclude the data from test subject 11 due to a corrupt video making it impossible to label or validate the results.

Scenario	Number Yield Violations
1	7
2	5
3	0
4	0
5	3
6	11
7	8
8	3
9	13
10	0

Table 7.3: The total number of yield violations discovered by the proof of concept across all participants of experiment 2 per scenario.

There was discovered a pattern where the proof of concept gave feedback on violations of yielding at the very beginning of a scenario in some instances. These instances were suspected to be caused by a bug. In total, the bug was identified 24 times across all of the scenarios, and it is illustrated in figure 7.8 in a later subsection. Below, in table 7.4, follow the results with the bug filtered out. The table also includes the percentage of agreement when compared with the evaluation done by a human driving instructor. In the case of the system not providing feedback, and the human driving instructor considering the handling of yielding to be sufficient, they are said to agree. In the case where the system recognizes a violation in terms of having to yield, and a driving instructor does the same thing, they are said to agree. The table serves as an indication of the system's correctness, and the total agreement was 83%.

Scenario	Number Yield Violations	Expert Agreement
1	6	69%
2	1	92%
3	0	100%
4	0	100%
5	1	92%
6	10	42%
7	0	100%
8	1	91%
9	7	45%
10	0	100%

Table 7.4: The total number yield violations across all participants of experiment 2 per scenario, along with the percentage of violations a human driving instructor agrees with.

Again, for speeding, this behavior was recognized by the proof of concept implementation several times throughout the scenarios. Each time the simulator log file values resulted in a computed speed for the Ego that was above the current speed limit in the Ego context, the system would recognize it being a speeding violation.

Scenario 6 - Not Adjusting Speed

This first situation, shown in figure 7.4 as an instance of scenario 6 (figure 7.2), illustrates the general situation where Ego has to yield and does not sufficiently adjust their speed. Consequently, the car not having to yield for Ego all of a sudden is required to brake to avoid a collision. In the situation of figure 7.4, as well as similar situations where Ego has to yield and has a possible collision with another car approaching the intersection, our system does register violations and provides feedback.

Figure 7.4: This example of scenario 6 shows the Ego not yielding when supposed to and having to compensate by braking hard. In this case, the proof of concept system provides feedback about violation of having to yield.

Scenario 6 - Yielding Properly

The second situation, shown in figure 7.5 as an instance of scenario 6 (figure 7.2), illustrates the general situation where Ego has to yield and properly adjusts their speed to allow other cars to drive by without interference. In this situation, as well as similar situations, where the Ego has to yield for other cars, but Ego speed is adjusted to avoid a collision, no possible collision relation is generated by our system. Therefore, per definition, having to yield is not violated and our system does not generate any violations or feedback.

Figure 7.5: This example of scenario 6 shows the Ego yielding when supposed to, by adjusting speed appropriately beforehand. In this case, the proof of concept system provides no feedback as there were no violations of having to yield.

Scenario 4 - Enforcing Right of Way

The third situation, shown in figure 7.6 as an instance of scenario 4 (figure 7.2), illustrates the general situation where Ego has right of way and disregards any approaching vehicles to the intersection. In this situation, as well as similar situations, the Ego does not technically have to adjust their speed for any other car, as they should yield for the Ego. Furthermore, there is no violation of having to yield as the ego has right of way. Therefore, even though a possible collision relation is present, our system produces no violations and no feedback.

Figure 7.6: This example of scenario 4 illustrates driving on in the case of having right of way when another car is approaching the intersection. In this case, the proof of concept system provides no feedback as there were no violations of having to yield.

Scenario 4 - Driving Passively

The third situation, shown in figure 7.6 as an instance of scenario 4 (figure 7.2), illustrates the general situation where Ego has right of way, but considers the speed of other approaching cars and ends up slowing down to let them pass. In this situation, as well as similar situations, the Ego does not technically have to adjust their speed for any other car, as they should yield for the Ego, but decide to do so as they consider the other car approaching the intersection too fast. In this situation, as well as similar situations where the Ego has right of way, there is no violation of having to yield. Therefore, although the possible collision relation is present, our system produces no violations and no feedback.

Figure 7.7: This example of scenario 4 illustrates stopping to let another car having to yield for you pass, in the case of having right of way. In this case, the proof of concept system provides no feedback as there were no violations of having to yield.

Scenario 2 - Initial Acceleration

The very beginning of scenario 2 illustrates the general situation where the Ego and another car approach the intersection. At this moment, the other car accelerates to reach a predefined speed within the scenario, being arbitrarily high for the very first moments of the scenario. In the cases where the straight path from the other car and the Ego intersected, this resulted in a projected collision and, therefore, a generated feedback from the system of Ego violating having to yield in the cases where Ego had to yield for the other car. This general scenario is illustrated in figure 7.8.

Figure 7.8: Initial acceleration is very high initially when cars spawn within the simulator, resulting in a possible collision and the system saying the Ego violates having to yield. In this case, the car is not seen, but is at the end of the road, in the lane opposite to the Ego.

Scenario 7 - Lane Adjustment

Another behavior observed was that when adjusting ones position lateral within the lane, typically done before turning to indicate the intended path, this action would result in feedback of violating having to yield for other cars in the opposing lane directly in front. Similarly to the initial acceleration example in scenario 2, this adjustment within the lane would project momentarily possible collision in the future. This general situation is shown with an example from scenario 7 in figure 7.9.

Figure 7.9: Adjustment of ones position in the lane, due to the intended path, is an over-sensitivity within the system that results in the system saying the Ego violates having to yield.

Scenario 9 - Other Car Collision Course With Ego

Finally, in scenario 9, a recurring situation was seen. Here, Ego has to yield for the other car, with Ego starting in the right lane with an intended path to the bottom lane, and the other car approaching from the bottom lane with an intended path to the right lane. As Ego has to yield for the other car, participants in the experiment would wait for the other car to drive through the intersection. However, after exiting the intersection, the other car has a projected collision in the future. Due to the Ego having to yield for the other car, the system would generate feedback saying Ego violated having to yield in this case. An example of this is shown in figure 7.10.

Figure 7.10: In scenario 9, the other car is on a collision path with the Ego, but the system says the Ego violates having to yield.

Example of Speeding Feedback

Here we have included a picture from the proof of concept output after running it on the data from test subject 11. In addition, to show what the system output during a period of speeding would look like, it was also chosen to highlight two other matters.

First, it highlights a trend of speed values having unexpected momentary variations. Although the actual driving simulator of Way AS provides speed values for the Ego car, all speeds were calculated by the speed interpolation agent based on the positional data coming from the simulator log files. As can be seen in picture 7.11, the momentary variation in speed values would differ by up to 4 km/t from one tick to the next, before returning to about the same value as before.

Second, picture 7.11 shows the frequency of which the system can give feedback when violations occur. One can see that feedback is given with approximately 35 milliseconds intervals.

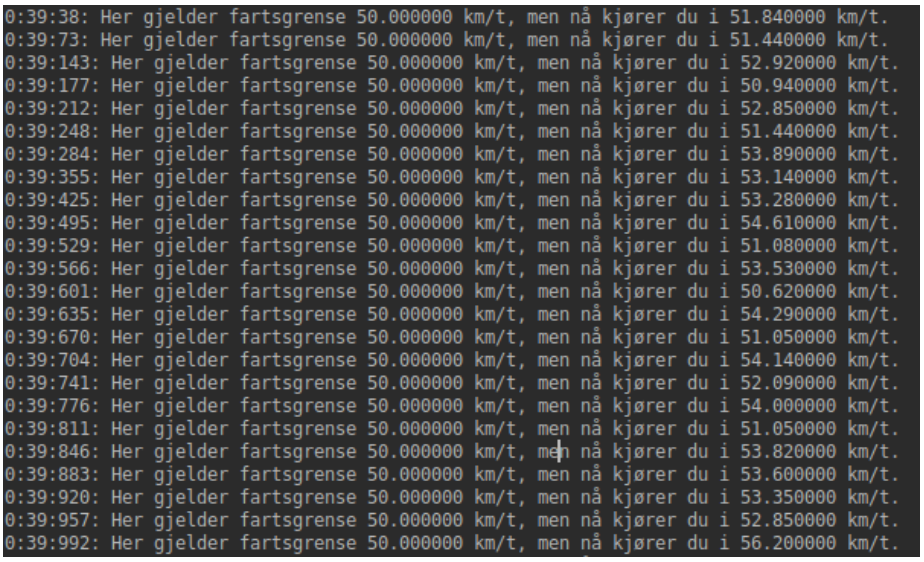


Figure 7.11: This figure shows part of the output for test subject number 6, provided in Norwegian. The picture shows speeding violations with the corresponding speed across a single second within the simulator. One can observe the general trend of speed having unexpected variations within a short interval, as displayed here, the speed values can vary quite drastically momentarily. Also, it is worth noting the frequency at which the system outputs feedback. The numbers xx:yy:zzz is the time in minutes, seconds and milliseconds from when the test drive started.

Test subject 11

Unfortunately, the video file recorded for the drive of test subject 11 was corrupt. Hence, even though the results were recorded we were not able to evaluate the results from the system, and the driving instructor was not able to comment on the scenarios of test subject 11.

Simulator Sickness

Some people experience motion sickness from driving in the simulator, as mentioned in the first section of chapter 5. Test subjects 4 and 14 experienced motion sickness to the degree where we had to interrupt their drive. Consequently, test person 4 only finished the first 4 scenarios and test person 14 only finished the first 6.

Chapter 8

Evaluation and Conclusion

This chapter contains the evaluation and conclusion of the work performed in this thesis. The evaluation synthesizes the results of the experiments and follows up with a discussion of them. The set of contributions from this thesis is identified, and finally, a conclusion and future work are discussed.

8.1 Evaluation

This section will evaluate the results of each of the experiments conducted and then proceed with a subsection on a general evaluation of the proof of concept implementation.

8.1.1 Experiment 1

The initial results showed that the proof of concept sometimes gave very early feedback of the driver not yielding properly, way too far away from the intersection. Some configurations of the first experiment initially showed that this feedback was mainly produced when the other car would be oncoming to the Ego, and is the result of predicting too far into the future. Therefore, a limit to how far the possible collision agent would look into the future was added. As described in chapter 6, this is based on the braking length given the current speed of the Ego. It is noteworthy that the experiments themselves facilitated looking too far into the future. Since each scenario started from a standstill and began with rapid acceleration then the early prediction would use unrealistic acceleration, as illustrated in a situation from experiment 2 in figure 7.8 of the

previous chapter. On the other hand, it may also have allowed for earlier detection of problems regarding the possible collision assessment used in the proof of concept.

However, after adding this restriction of time in the future, we were pleased to compare the feedback expected from table 7.1, and the actual output documented in table 7.2. In each of the scenarios where the authors attempted to drive incorrectly, our system did provide feedback saying that having to yield was violated. Furthermore, this took the context into account, saying that it was due to the basic right of hand rule in the cases where no road signs applied, and it was due to a yield sign in the case where yield signs imposed having to yield.

An important note about these experiments of experiment 1, as mentioned in the previous chapter, is that they were driven by the developers of the proof of concept implementation. Therefore, they are aware of what is required for the system to provide feedback on yielding having been violated and can adapt their driving accordingly. Though a subjective evaluation by the authors, we do consider the "correct" scenarios to represent a normal way of driving through the intersection in a correct manner. Similarly, for the incorrect examples, we consider these representatives of situations where a driver does not drive according to having to yield for the other car.

8.1.2 Experiment 2, T-intersection

The T-intersection of experiment 2 encompassed scenarios 1 through 6. As can be seen from the results of table 7.4 in the previous chapter, the violations recognized by our proof of concept implementation were in scenarios 1, 5 and 6. There was only a single violation recognized in scenario 5.
Further investigation showed that the single violation in scenario 5, where a test subject actually drove outside of the road thereby missing the trigger providing them with right of way. This meant that technically, with regards to yielding, the scenario was the exact same as scenario 2. However, the violation was identified closer to the intersection, being regarded as an example of the general situation where the Ego adjusts its lateral lane position as shown in figure 7.9. As expected, the human driving instructor did not consider this a valid feedback. The lane adjustment leading to a projected collision in the near future is discussed further in the next section.
For the scenarios where several violations were recognized by our systems, however, the human driving instructor agreed with 69% of the violations in scenario 1, and only 42% of the violations in scenario 6. The result from scenario 1 indicates a decent degree of correctness in our system, while 42% in scenario 6 is terrible. For scenario 6, guesswork would outperform the proof of concept

system. These scenarios will be subject of the discussion later in in this chapter.

Though accuracy of 100% is ideal, an overall accuracy for the scenarios of experiment 2 at 83% is quite decent. Scenarios 2 through 5 were considered clear cut by the human driving instructor. The high agreement percentage of these scenarios indicates that the system performs well in clear cut scenarios. While the difficult situations may be the most interesting, it is also important that the system masters the simple situations.

8.1.3 Experiment 2, 4-way intersection

The proof of concept did not alter its behavior or performance when adapted for the 4-way intersection. Although moving from a T-intersection to a 4-way intersection may seem like a small change, the fact that the proof of concept kept its performance proved that it was able to generalize yielding concepts.
The 4-way intersection encompassed scenarios 7 through 10. Again referring to the results of table 7.4, the system recognized violations in scenarios 7, 8 and 9.

The single violation of scenario 8 proved to be an example of the driver again missing the trigger providing them with right of way, again being subject to the possible collision as a result of lane adjustment.
It turned out that the single violation of scenario 7 was due to the same reason. In fact this is the exact scenario in figure 7.9. As this was not a single occurrence, but actually proved to be a general problem across scenarios, ways of handling a driver not passing through the trigger imposing right of way or having to yield will be discussed below.
For the 7 violations recognized in scenario 9, the system would say the Ego violated having to yield. In only 45% of these situations, the human driving instructor agreed. This indicates a severe flaw in the logic of our application, especially as the human driving instructor was able to see the entire situation in his field of view due to it occurring very close the intersection, being independent of the time when the cars are far from it. The example of this scenario was illustrated in figure 7.10 of the previous chapter. Clearly, the Ego is not at fault here, but rather the other car having a momentary collision course with the ego due to high speed when taking a turning. However, as the Ego has to yield for another car, and has a possible collision with this car, which is currently in or approaching the same intersection as the Ego, our system recognizes this as a violation by Ego. This will be discussed below.

8.1.4 General Evaluation

Video and Log Offset

We used context updates to track the time offset between the video and the log file. A context update would be made whenever the test subject passed a sign or left the intersection. These are clear and frequent moments during the drive of a test subject. When investigating the log file produced from experiments, to validate the violations recognized by the system, we expected that the offset would remain constant, thus only looked at the first context update as a reference. However, it was discovered that the offset in time between the video and the log file changed over the course of a test drive. For example for test subject 7 the video was 7 seconds behind the log when he passed the first speed limit sign, meaning the time 0:30 in the video was 0:37 in the log. Three minutes later, at 3:28, the video and the log were in sync.

Yielding

From the overall results, it seems that the proof of concept's notion of violating having to yield corresponds well with that of human driving instructors. The human driving instructor was mainly considering whether or not the Ego car was disturbing the other car and considered any disturbance as a violation of having to yield. The results confirm our initial beliefs that the possible collision relation is a good indication, due to the threat of possible collision disturbing the other car.

Speeding

Looking at speeding is trivial. The fact is that the system is able to keep track of the currently applying speed limit on a road. Furthermore, the speed interpolation agent of the proof of concept implementation keeps an interpolated value of the speed of all cars. Therefore, whenever the Ego has a speed higher than the speed limit of the context, they are per definition speeding. The system would recognize this as an evaluation and provide feedback. This behavior worked as intended 100% of the time.

8.2 Discussion

This section discusses important matters from the above evaluation and the results of the previous chapter, to highlight areas of importance for future work

on the system and similar systems. Initially, the results of the experiments are discussed in the context of the hypothesis, objectives, research questions identified in chapter 1. Next, specific issues are discussed, and finally, a general discussion is included.

8.2.1 Hypothesis, Objectives and Research Questions

Below is the summary copied from chapter 1. This subsection will consider each of the objectives in turn, and look at their respective research questions, to see if they have been answered to a satisfying degree through the implementation and experimental results. An evaluation of our hypothesis will be left for the conclusion.

Hypothesis A Virtual Driving Instructor System, using AI, can be used to evaluate a driver's behavior to give precise, justified, and understandable feedback about a traffic situation the driver can use for improvement.

Objective 1 A Virtual Driving Instructor must be able to understand the situation and context.

RQ 1 How to represent both the situation and context that applies at a given time in traffic?

RQ 2 How can missing information be inferred from domain knowledge, and basic information of situation and context for a given time in traffic?

Objective 2 A Virtual Driving Instructor must be able to evaluate driving behavior.

RQ 3 Based on driver actions, can driving behavior be recognized and represented?

RQ 4 How should driving behavior be evaluated with respect to a given representation of traffic rules, situation, and context?

Objective 3 Feedback generated by a Virtual Driving Instructor must be understandable for humans with little to no understanding of AI.

RQ 5 When and how should a Virtual Driving Instructor generate feedback?

RQ 6 Given the way feedback is generated by a Virtual Driving Instructor, can it also be justified?

Objective 1

As the results of the experiments show a fairly high degree of correctness, the proof of concept implementation seems to be able to represent the situation

properly. The rationale behind this argument is that a system not able to understand the situation and context will not have any basis for feedback, but as our implementation provided the correct feedback in most situations the system understands both to an appropriate degree.
Research question RQ 1 has been answered quite clearly, the way to represent the situation and context is through an ontology. The ontology consists of the relevant parts of the situation and context and was able to infer violations when they occurred.
Research question RQ 2 was answered by the implementation of a multi-agent system. Each agent is responsible for adding additional information for a more extensive context and situation representation. Using basic information like other car positions and which lanes they are in, as well as what road signs apply, the system is able to infer which cars have a collision course with other cars, as well as which cars are obligated to yield for others.

Objective 2

The system produces feedback, being the result of an evaluation based on how a driver is behaving in the simulator. Hence the proof of concept was able to evaluate driving behavior.
When it comes to research question RQ 3, the driver actions or inputs are realized as behavior within the system. The system has rules defined for recognizing behavior that violates the currently applying traffic laws, like violating having to yield or driving at a speed above the current speed limit. These violating behaviors have been defined within the agent system and ontology, which is how they are recognized.
Research question RQ 4 has been answered by taking into account the currently applying context and situation of the Ego. Ego behavior is recognized by higher-level agents in combination with the definition of behavior through ontology violations. The sum of traffic rules, situation elements and context serve to produce a set of relations that indicate which cars the Ego is obligated to yield for, and if a possible collision relation is also generated, a violation is produced.

Objective 3

The textual feedback from the system is shown in the example of figure 7.11, using simple sentences in the proof of concept system.
Research question RQ 5 has been answered to a certain degree. When it comes to *how* the Virtual Driving Instructor should generate feedback we have provided it as textual feedback in natural language sentences for the proof of concept implementation. However, it is provided as soon as it is generated in this proof of concept. This contradicts the suggested solution in chapter 4, where we propose giving feedback in situations of low stress, with a much lower frequency than

what was done in the proof of concept.
When it comes to whether the feedback can be justified, asked in research question RQ 6, the answer is yes. As our system keeps track of the current context along with the situation, and all relations between the cars in the current situation, all information to provide justification is present. For now, in the case of violating having to yield, we have settled with identifying how you are violating having to yield (whether due to not respecting road signs or the basic right-hand rule), as well as which car you should have yielded for in our proof of concept implementation. However, the feedback could contain even more justification if required due to the extensive internal representation.

8.2.2 Speeding

The success of providing feedback on speeding should be taken with a grain of salt, as figure 7.11 of chapter 7 illustrated that the interpolated values of speed can have significant momentary variations. One could argue that as the tick at which speed is computed within the simulator is below 100 ms, the effect of such variations will not lead to speeding violations being omitted, as the value is likely to return to its actual value. Still, this has propagating effects throughout the system as both the possible collision agent and acceleration interpolation agent depends on these speed values. Again, as stated in chapter 5, the optimal solution to such a problem would be having direct access to the actual speed (and acceleration for the possible collision agent) values from the simulator rather than having to compute them within the proof of concept, and the values computed by our speed interpolation agent are possibly inaccurate due to them being results of interpolation. Furthermore, having the speed values coming directly from the simulator would lighten the computational load of the system, thereby improving the performance.

The problem of providing feedback on speeding opens for more interesting challenges. Among these are how severe a violation has to be for a virtual driving instructor to provide feedback on it. When displaying the experiment drives to human driving instructors they would react when a person was either speeding severely momentary or speeding slightly over a larger period of time. Therefore how severe a violation must be for a virtual driving instructor to provide feedback should be investigated.
Based on this, a new issue arises, which is how often one should provide feedback. A suggestion based on Arroyo et al. (2006) was given in chapter 4, of based on all violations or recognized behavior up until some point is given as soon as possible when being in a low-stress situation. This feedback should highlight the most important aspects of driving to provide feedback on, given the individual driver.
Another aspect of speeding is our solution treating it as trivially as it does. This proves that our proof of concept does not over-complicate aspects of driv-

ing that do not need to be over-complicated. The rationale behind this relates to Ockham's razor, solving a problem with the simplest solution possible. Furthermore, it illustrated how the proof of concept can easily be extended to reason about other more or less trivial behavior.

8.2.3 Having to Yield

When it comes to reasoning about having to yield, the experiments have revealed three issues that need to be investigated further and be improved.

First of all, is the issue related to the initial acceleration of other cars within the simulator. An example of such a scenario is shown in figure 7.8. As stated previously, the possible collision relation is central to our reasoning about having to yield, and the projected future state from this agent is affected by the initial acceleration. There are several ways that could improve this, for instance, the possible collision agent could have its computation split into several parts that rather than trying to solve the general problem take the situation and context into account. Furthermore, it could take into account the other car probably not having constant acceleration, but stopping to accelerate once their speed is at or slightly above their current speed limit. Another factor that could help this logic is having access to the computes values from the simulator itself rather than having an agent be responsible for interpolating the values of acceleration (again the acceleration values are based on the interpolated values for speed).
The rationale behind why a change is needed is the following: feedback on something predicted too far into the future would confuse students. While some weird or wrong feedback can be filtered by the student, it will still weaken the authority of the system. Furthermore, a student will likely not be able to filter out improper feedback as that would require expert knowledge that the student has yet to learn. Even worse, the student is expecting to learn this expert knowledge from the virtual driving tutor.

The second issue identified with reasoning about having to yield is that of scenario 9 (shown in figure 7.10). Due to the other car being within the intersection that the Ego is approaching, the Ego has to yield, and the other car having a projected collision with the Ego. In this situation, it is clear that the other car is crashing into the Ego, yet the system says Ego violates having to yield due to a possible collision with a car Ego has to yield for. Clearly the logic is not robust enough, and improvements must be made. However, we do not consider this an indication of the suggested solution of chapter 4 being bad, but rather a result of shortcomings in our proof of concept. As pointed out, possible collision is a very complex concept to model and requires a more advanced agent, else, other supporting concepts have to be included to capture more of the nuances of the situation.
One suggestion, is making the possible collision agent more sophisticated, by

distinguishing between whether the Ego will collide with another car, if the cars both collide with each other, or if the other car is colliding with the ego (as here), and then disregard all scenarios where the other car is responsible for the collision. The possible collision agent is discussed further, later on in this section.

The third issue if that of lane adjustment leading to a projected collision in the future, as illustrated in figure 7.9. This momentary change in the acceleration due to positioning in lane leads to a projected collision in the future, and a violation of having to yield being violated in our proof of concept system. Again this must be improved, whether with a more sophisticated agent responsible for computing the possible collision, or some other way. For instance, a momentary violation of a traffic law lasting only one timestep within the simulator could be disregarded, while a more clear violation of traffic laws persisting over several timesteps should be recognized.

8.2.4 Driving Instructor Evaluation

Referring to the results of chapter 7, the feedback from the system mostly followed that which a human driving instructor would give. Still, there were examples of the system providing feedback and the human driving instructor not doing so. A notable difference was that the proof of concept implementation proved to be a more defensive driver than the human driving instructor. A defensive driver is typically extra careful and passive when interacting with other entities in traffic. This tendency is illustrated through the worst agreement percentage of 42% for scenario 6 where inspection showed that the proof of concept wanted clearer, earlier retardation than the human driving instructor. We believe that the percentage of the agreement does not serve the proof of concept justice, as three factors were identified as potential causes for the disagreement, apart from the possibility that the system actually outputs wrong feedback.

The first factor is that in these scenarios, the human driving instructor himself told us that there was quite a bit of uncertainty due to the field of view and information available in the video recording. The human driving instructor would not be able to see the other car approaching the intersection until either car was very close to the intersection. The human driving instructor would have liked to see the other approaching car much further away from the intersection to be able to evaluate if the car interferes with the Ego or the Ego interferes with the other car. A top-down overview of the entire situation was also suggested.
After this uncertainty was discovered, we asked whether or not the judgment could have been different if the other car could have been seen earlier in the situation, to which the human driving instructor said yes. Asking specific questions where we would inform him of the actual speed values and locations of the car in the scenario indicated at just the edge of the field of view at a current timestep made the human driving instructor provide the same feedback as the

system did in that situation.
This made us conclude that although there is a significant difference in the feedback of the system and the human driving instructor in scenario 6, and to some degree in scenario 1, we should regard these result with some skepticism.

The second factor pertains to human driving instructor discretion. Often times there is not a hard line between right and wrong. An example illustrating this is that a momentary potential collision is not necessarily going to happen in the future if both cars brake significantly. However, our implementation of the possible collision agent assumes that speed and acceleration at a given moment will remain constant, and therefore has a tendency to consider the collision as probable in the cases where a driving instructor might not.

Thirdly, the human driving instructor was fond of discussing driving actions in the context of hypothetical future situations, i.e. in terms of best general practices. Even though there were no negative consequences of driver action within a given scenario, the human driving instructor would still consider their actions a violation of having to yield. This illustrates the need for adding a more sophisticated approach taking best practices into account when evaluating yield situations, either through introducing more agents, more sophisticated logic to the responsible current agents or by allowing a higher abstraction level or the tutor of the system make these considerations.

8.2.5 Feedback Frequency

The feedback shown in figure 7.11 shows how the frequency of speeding feedback is extremely high. This proof of concept implementation has focused on showing that our approach is able to provide feedback on traffic situations, but one cannot give feedback whenever there is something wrong. The feedback has to be spaced out, as a student can only take so much input at a time. The picture shows feedback on speeding being given more than 20 times within the course of a second. Also, giving the same feedback for each timestep it is generated could cause the same feedback to be repeated way too often, turning the feedback into spam. This could make a driver disregard it completely, filtering it out as noise. On the other hand, seeing that the system is able to catch violations at such a fine granularity is promising for further development. Still, careful thought should be put into the frequency and manner of providing feedback.

8.2.6 General Topics

Being able to generalize is a key attribute of good artificial intelligence systems. We consider the ability to generalize as one of the key attributes of a virtual driving instructor. Therefore, we were pleased to see that the system handled

the change of intersection structure. It shows that the proposed solution supports generalization.

From the preliminary work, it was expected that the possible collision agent would be key to the performance of any virtual driving instructor or an autonomous car, hence also the proof of concept. While the agent developed for the proof of concept proved adequate, we still believe there could be done some improvements. However, we do not believe that making a stacked possible collision agent is the way to go. There are simply too many and too complex concepts within the assessment of possible collision. As an example, we predicted if the position of two cars would overlap at some point in the future given their momentous speed and acceleration. Only slight positional adjustments within their lane may cause oncoming cars to hold true for the calculation, i.e. lateral lane position adjustment. However, one is usually to assume that a car will try to stay within their lane. Therefore, we think that a possible collision agent has to take this assumption into account and such inclusion would improve feedback regarding oncoming cars.

Based on the discussion of the issues seen with yield violations, we proposed capturing the different aspects of possible collision should be handled by individual agents. Then a higher level agent should use these agents to assess the likelihood of a collision. Having to make changes like that shed light on the high degree of modifiability and suitability of the proposed solution. No other changes are needed than the decomposition of the possible collision agent, the rest of the agents remain unchanged. As a side note, another consideration generated by decomposing the possible collision agent would be to move onto a numerical representation of possible collision.

Another aspect of the proof of concept system we would like to highlight is that some traffic situations require discretion, whereas computers and systems like our proposed solution usually fare better with absolutes. This was briefly discussed in the subsection of driving instructor evaluation. The need for discretion is clear in the case of a possible collision. In our proof of concept implementation, this agent implements a general algorithm checking if the collision will happen in the near future, where the near future is restricted by twice the time it takes for the car to stop with its current speed and acceleration. In some situations this computation works very well, as illustrated by the accuracy of table 7.4 in which the human driving instructor agrees with the proof of concept, but the fact this it is not 100% (like in scenarios 1 and 6) could be due to the fact that this computation is not specific enough for all scenarios. There is clear discretion present here, which in the words of the human driving instructors depends on several factors, like the speed of the cars, the speed limit, the road surface, the weather, and probably even more factors. This type of discretion is relevant for other agents to be implemented in a more full system,

but shows up most clearly in our possible collision agent in the proof of concept implementation.

Finally, an overview of the status quo of the proof of concept implementation is discussed. As of right now, the proof of concept should not be used as a standalone system providing feedback to drivers, but rather be used as a supplementing tool for a human expert (a driving instructor). There are several reasons for this, as identified previously in this chapter. Among these reasons are discretion, as well as the frequency feedback is provided at and the sensitivity of our possible collision agent in identified scenarios.
Weird or wrong feedback, which is currently present, would confuse a learner driver. Furthermore, invalid feedback that can be recognized by drivers would have the system lose credibility and the driver would not care to learn from it. Therefore, the system could, in theory, be used to flag certain times in the drive where it has recognized mistakes, and a human driving instructor could then manually look at these situations to see if the feedback is valid or not. In this scenario, the system would work more like a decision support system.
Another aspect of the status quo and rationale as to why it should be used as a tool in its current state is that human driving instructors make drivers reflect over their driving. The current proof of concept simply outputs a sentence saying that a mistake was made in the current situation, as well as why it was a mistake. This does not allow reflection. To be able to work as a standalone system, the implementation could aim for a reflective dialogue with the driver rather than providing direct feedback as it does now.

8.3 Conclusion

This thesis outlines a proposed solution for a virtual driving instructor. Moreover, it documents the implementation of a proof of concept system based on the proposed solution of a multi-agent system using a blackboard architecture, representing the situation, context, and traffic violations within an ontology.
Based on the assumption that a driver is driving according to the traffic laws they are aware of, as well as the elements of the situation they have observed, improper actions and behavior result from flaws in their situation awareness. Furthermore, what mistakes are made should indirectly indicate the flaws in the driver's situation awareness.
Analysis and performance of the proof of concept suggest that keeping track of driver actions and behavior, as well as having perfect situation awareness, makes the system able to provide corrective feedback when appropriate to a driver on the topics of violating having to yield and speeding.
The experimental results suggest that this proof of concept implementation has some merit to it, but considerable work remains for a fully fledged virtual driving instructor to be implemented. An overall feedback agreement percentage

of 83% between the proof of concept implementation and a human driving instructor was observed. However, the agreement percentage could go as low as 42% in difficult situations.
Issues with the robustness of the proof of concept agent implementations were also identified, especially regarding the sensitivity of the possible collision agent. Furthermore, an issue with the video recordings of the experiments was identified. The human driving instructor requested a broader field of view and reflected that he would probably have recognized more of the scenarios as violations if he was able to see the approaching cars further out from the intersection.

8.4 Future Work

The proof of concept implementation and the testing of it through the scenarios of experiments 1 and 2 revealed a lot of information that can be used for further improvement of this or similar systems. The information relating to future work is categorized and discussed below. Both identified flaws, as well as proposed additions of functionality, are documented.

8.4.1 General Work

First and foremost, we propose improving the video recording taken of a driver driving within the simulator. There was a clear issue for human driving instructors to evaluate the scenarios presented within the experiments, as they were unable to retrieve all the information they would like. We propose either having a wider field of view, such that the information further to the left and right of the car is available, or to also include a top-down view of each intersection clearly showing any approaching cars, or other relevant elements.
The field of view solution resembles the task more closely as the view is the same the driver has within the simulator. However, presenting a large field of view video could skew the perception of speed and therefore reduce the quality of evaluation. A top-down view of the entire traffic situation is able to maintain the perception of speed, but issues could arise with it being a different point of view than what a human driving instructor is used to.

Another issue identified early on was that of a memory leak within the ontology representing the situation. While this was handled by proper garbage collection within the application, problems could arise with more complicated scenarios than that of a single intersection with only two cars present. Inspiration from the solution in Zhao et al. (2017), utilizing a temporary ontology, should lay the foundation for solving such an issue if present in the future.

8.4.2 Simulator Data

There are several shortcomings in terms of the data our proof of concept system was able to utilize in our experiments. First and foremost is the issue of not having access to information about what indicator lights are currently indicating for any car. This information allows drivers to infer other cars intended paths ahead of time (given that they do use their indicator lights to indicate where they are going). By integrating this information into the proof of concept the foundation for reasoning about which cars in an intersection will have an intersecting path with the Ego's path is made available. Furthermore, indicators solve the problem where our system had to propagate this information backward in time, only having access to path information once cars actually have driven through the intersection. Thus, indicators allow better reasoning about having to yield in real-time and is something that should be implemented in the future. Another detail related to ones intended path in a traffic situation is the lane positioning, this can provide some of the same information as indicator lights, and is another feature that a system could evaluate and provide feedback on. We propose investigating lane positioning further.

Being able to provide feedback in real-time is significantly more likely to be achieved once indicator light data is available, but another improvement that could be made to the system would be parallelization. Currently, all agents satisfied with a given set of information add their information to the blackboard sequentially. In the future, all independent agents should be run in parallel to better achieve timing requirements. We propose either running all agents until all information has been computed for a given timestep, or providing a cut-off when enough information has been added to infer a violation that requires immediate feedback. The timing required to process each timestep should be investigated in the future, along with the relative performance improvement of parallelizing the set of agents.

8.4.3 System Improvements

We would like to start by discussing how and why to include new agents within the system. As mentioned several times before, a virtual driving instructor is only able to reason with what data it has access to. The perfect situation awareness of the system is constrained by what aspects of the situation the simulator data allows it to keep track of. Thus, if the system was to reason about something like the driver's gear usage, an agent responsible for evaluating the gear use would have to be added. Of course, this agent would require information on which gear the driver currently uses, and probably other information like their current RPM. Finally, the addition of axioms evaluating the performance measure on gear-use by the agent would have to be added to the ontology.

This example illustrates the ease of extending the system, provided that the relevant basic information provided from the simulator is available. When present, one simply extends the system with the necessary agents able to add a more rich description of the given information to the ontology and adding axioms to the ontology to describe how to evaluate the given aspect of driving.
However, as the system grows the challenge of the relative priority of agents, and their hierarchical organization could become a challenge.

The next point addresses the issue of possible collisions being computed on lateral lane adjustments in scenarios of our experiments. In general, one expects cars to follow their lanes. This knowledge was not utilized when it came to predicting collisions. Cars probably following their lane should somehow be included, either as part of predicting collision or as a separate notion that can be included to determine if we care about the possible collision or not. This improvement should make the system more robust for small alignment adjustments inside a lane, identified as a general problem from the results of chapter 7. Our suggestion is to leave possible collision as is and include the lane assumption as a part of the blackboard through a higher level agent, able to moderate the signal of the possible collision based on the current situation.

Another side to yielding violations was also identified through the driving instructor evaluation; yielding when one has right of way. This is challenging to evaluate, and thus not included in this proof of concept due to time constraints. However, a human driving instructor would make a learner driver aware that such behavior slows down the flow of traffic, and is consequently a violation in itself. Such behavior is hard to evaluate, as the driver could be slowing down because the other driver does not show signs of yielding. This aspect of yielding relates to driving safely, not directly to violating having to yield, which is another reason why it was left out of this proof of concept. We do propose pursuing solutions on how to evaluate approaching an intersection in a safe manner, apart from the simple violation of having to yield, as this was something the human driving instructor brought up when evaluating the scenarios of our experiments.

Something not addressed, as we were only considering a single intersection within our scenarios, was what happens with the context after exiting the intersection. There are several possible cases here, where continuing on the road with right of way should maintain this aspect of the driving context, but making a turn should reset the context in the sense of right of way. Again, as our scenarios only tested a single intersection we were not able to observe problems with this lack of behavior, but for a later consistent and correct system, potentially changing or maintaining parts of the driving context based on the incoming and outgoing lanes one take in an intersection is required.

As a final note, if one intends to build a complete system based on the proposed

solution in this thesis, we think that incrementally playing with and adding agents is the way to go.

8.4.4 Situation Awareness Assumption

This thesis assumed that incorrect actions were caused by incorrect situation awareness in the driver. The validity of this assumption should, however, be investigated further. Moreover, one should look into how precisely one can identify the flaws in the situation awareness given the driver's actions. Furthermore, we propose designing experiments to test to what degree corrective feedback improves a driver's situation awareness.

8.4.5 Feedback and Driver Profiling

One of the challenges needing further investigation is the form of feedback. For example, the feedback can be given as text on the screen. Another way is by audio. One could even go to the length of making video reconstructions of particular situations and show it to the student from a different view. The opportunities are many and all have their merits. We propose to look into the strength and weaknesses of the different forms of feedback that can be given.

Another aspect is that of when to give feedback. Even the best advice is futile if given at the wrong time. If the student is too focused on the driving situation, then the student may not pay attention to the feedback. An interesting thought would be to track the stress level of the student and try to time feedback with less stressful situations, as suggested in Arroyo et al. (2006). On the other side, if you wait too long to give feedback, then the student may have forgotten too much of the situation that spiked the feedback, thus making the feedback less effective as it is given out of context. However, this is just relevant for real-time feedback during a driving session. It could just as well be that giving feedback along with a video at the end of a session is the best time to give feedback.

Based on the discussion of feedback frequency in the previous section, we recommend looking into how to aggregate feedback over multiple timesteps to build higher-level feedback.

A simple suggestion is to look at changes like saying "you are breaking the speed limit" for each timestep to "You have been driving over the speed limit for the last minute".

The Tutor-part of our proposed system should look into feedback generated over time, and based on this provide feedback on more overarching tendencies of the driving, like "You tend be hitting the brakes too aggressively". Methods of

aggregating single instances of violations from a drive, to such feedback should be looked at.

Human driving instructors do not only give feedback to students when they do mistakes. Positive reinforcement is just as important as negative reinforcement for some driving instructors. Our work has only been focused on detecting mistakes and giving feedback on those. There should also be conducted research into giving appraise on good driving behavior. Then the results of this research should be compared to the results of our research in providing corrective feedback on improper behavior. Witnessing professional driving instructors in action we suspect that a combination of positive and negative reinforcement would yield the highest learning outcome for the average student.

The job of a driving instructor goes beyond teaching a student to drive legally. Their job is to help students become the best drivers possible. This includes teaching students in best practice driving. Therefore, we think it would be interesting to investigate how to also be able to give feedback intended to just improve a student's driving skills. One could, for instance, look for tendencies in a student's driving such as the student often having sudden changes in brake pressure and give feedback that one should usually brake smoothly or look further ahead in traffic to adjust speed earlier. This would required performance measures for each aspect of driving, like how early a student is able to locate dangers within an intersection, their patterns in using the clutch, brakes, throttle and gear, their RPM and gear-use, etc.

The sum of the above subsection is proposing that a sort of profiling of each student is to be looked into. This matches what driving instructors continuously evaluates during driver's education, as they attempt to evaluate whether a driver is in the very fresh state still learning how to drive the car, if they are in a more intermediate state and should start looking into more advanced aspects of driving, or if they are about ready to take their driver's test. This categorization of the student is based on performance measures across the dimensions mentioned above, as well as many more, and is something that should be considered in the future for a fully autonomous virtual driving instructor system implementation.

8.4.6 Instructor-Student Interaction

This thesis has been focusing on understanding traffic situations and detecting when there are mismatches between expected and actual situations. In other words on the world - virtual driving instructor interaction. There is however another dimension of virtual driving instructors that need to be investigated, the instructor-student interaction. While we looked into how to provide expla-

nations with our feedback, we did not go nearly enough into the depth necessary to convey the feedback properly. Multiple challenges were identified in order to ensure proper interaction with driving students. All of these should be subject to further investigation if one is to implement a successful full-scale virtual driving instructor.

Bibliography

Arroyo, E., Sullivan, S., and Selker, T. (2006). Carcoach: a polite and effective driving coach. In *CHI'06 Extended Abstracts on Human Factors in Computing Systems*, pages 357–362. ACM.

Buechel, M., Hinz, G., Ruehl, F., Schroth, H., Gyoeri, C., and Knoll, A. (2017). Ontology-based traffic scene modeling, traffic regulations dependent situational awareness and decision-making for automated vehicles. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1471–1476. IEEE.

Corkill, D. D. (1991). Blackboard systems. *AI expert*, 6(9):40–47.

Dey, A. K. (2001). Understanding and using context. *Personal and ubiquitous computing*, 5(1):4–7.

Dominguez, C., Vidulich, M., Vogel, E., McMillan, G., et al. (1994). Situation awareness: Papers and annotated bibliography. *Armstrong Laboratory, Human System Centre*.

El Aeraky, S., Dollfuss, M., Kopciak, P. A., Kolar, P., and Daniela, H. (2016). Virtual reality driving simulator prototype for teaching situational awareness in traffic.

Endsley, M. R. (1988). Situation awareness global assessment technique (sagat). In *Aerospace and Electronics Conference, 1988. NAECON 1988., Proceedings of the IEEE 1988 National*, pages 789–795. IEEE.

Endsley, M. R. (1995). Toward a theory of situation awareness in dynamic systems. *Human factors*, 37(1):32–64.

Erman, L. D., Hayes-Roth, F., Lesser, V. R., and Reddy, D. R. (1980). The hearsay-ii speech-understanding system: Integrating knowledge to resolve uncertainty. *ACM Computing Surveys (CSUR)*, 12(2):213–253.

Fu, X. and Soeffker, D. (2010). Cognitive awareness of intelligent vehicles. Technical report, SAE Technical Paper.

Geng, X., Liang, H., Yu, B., Zhao, P., He, L., and Huang, R. (2017). A scenario-adaptive driving behavior prediction approach to urban autonomous driving. *Applied Sciences*, 7(4):426.

Geyer, S., Baltzer, M., Franz, B., Hakuli, S., Kauer, M., Kienle, M., Meier, S., Weißgerber, T., Bengler, K., Bruder, R., et al. (2013). Concept and development of a unified ontology for generating test and use-case catalogues for assisted and automated vehicle guidance. *IET Intelligent Transport Systems*, 8(3):183–189.

Guarino, N. (1998). *Formal ontology in information systems: Proceedings of the first international conference (FOIS'98), June 6-8, Trento, Italy*, volume 46. IOS press.

Gusikhin, O., Filev, D., and Rychtyckyj, N. (2008). Intelligent vehicle systems: applications and new trends. In *Informatics in Control Automation and Robotics*, pages 3–14. Springer.

Gutierrez, G., Iglesias, J. A., Ordoñez, F. J., Ledezma, A., and Sanchis, A. (2014). Agent-based framework for advanced driver assistance systems in urban environments. In *17th International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE.

Huelsen, M. (2014). *Knowledge-Based Traffic Situation Description*, pages 93–138. Springer Fachmedien Wiesbaden, Wiesbaden.

Hülsen, M., Zöllner, J. M., and Weiss, C. (2011). Traffic intersection situation description ontology for advanced driver assistance. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 993–999. IEEE.

Hwang, C. H., Massey, N., Miller, B. W., and Torkkola, K. (2003). Hybrid intelligence for driver assistance. In *FLAIRS Conference*, pages 281–285.

Kappé, B., van Emmerik, M., van Winsum, W., and Rozendom, A. (2003). Virtual instruction in driving simulators. In *Driving Simulator Conference, Dearborn, MI*.

Konstantopoulos, P. (2009). *Investigating drivers' visual search strategies: Towards an efficient training intervention*. PhD thesis, University of Nottingham.

Len Bass, Paul Clements, R. K. (2012). *Software Architecture in Practice - Third Edition*. Addison-Wesley.

Martelaro, N., Sirkin, D., and Ju, W. (2015). Daze: a real-time situation awareness measurement tool for driving. In *Adjunct Proceedings of the 7th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pages 158–163. ACM.

Martí, I., Tomás, V. R., Saez, A., and Martinez, J. J. (2009). A rule-based multi-agent system for road traffic management. In *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, volume 3, pages 595–598. IEEE.

Matheus, C. J., Kokar, M. M., and Baclawski, K. (2003). A core ontology for situation awareness. In *Proceedings of the Sixth International Conference on Information Fusion*, volume 1, pages 545–552.

Mohammad, M. A., Kaloskampis, I., Hicks, Y., and Setchi, R. (2015). Ontology-based framework for risk assessment in road scenes using videos. *Procedia Computer Science*, 60:1532–1541.

Morignot, P. and Nashashibi, F. (2012). An ontology-based approach to relax traffic regulation for autonomous vehicle assistance. *arXiv preprint arXiv:1212.0768*.

Nii, H. P. (1986). Blackboard systems. Technical report, STANFORD UNIV CA KNOWLEDGE SYSTEMS LAB.

Provine, R., Schlenoff, C., Balakirsky, S., Smith, S., and Uschold, M. (2004). Ontology-based methods for enhancing autonomous vehicle path planning. *Robotics and Autonomous Systems*, 49(1-2):123–133.

Radeckỳ, M. and Gajdoš, P. (2008). Intelligent agents for traffic simulation. In *Proceedings of the 2008 Spring simulation multiconference*, pages 109–115. Society for Computer Simulation International.

Raptis, D., Iversen, J., Mølbak, T. H., and Skov, M. B. (2018). Dara: assisting drivers to reflect on how they hold the steering wheel. In *Proceedings of the 10th Nordic Conference on Human-Computer Interaction*, pages 1–12. ACM.

Regele, R. (2008). Using ontology-based traffic models for more efficient decision making of autonomous vehicles. In *Fourth International Conference on Autonomic and Autonomous Systems (ICAS'08)*, pages 94–99. IEEE.

Sharon, G. and Stone, P. (2017). A protocol for mixed autonomous and human-operated vehicles at intersections. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 151–167. Springer.

Sipele, O., Zamora, V., Ledezma, A., and Sanchis, A. (2018). Advanced driver's alarms system through multi-agent paradigm. In *2018 3rd IEEE International Conference on Intelligent Transportation Engineering (ICITE)*, pages 269–275. IEEE.

Sukthankar, R., Baluja, S., and Hancock, J. (1998). Multiple adaptive agents for tactical driving. *Applied Intelligence*, 9(1):7–23.

Sun, J., Zhang, Y., and Fan, J. (2011). Smartagents: A scalable infrastructure for smart car. In *2011 12th International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 99–103. IEEE.

Tranvouez, E., Fournier, S., Espinasse, B., et al. (2013). A multi-agent system for learner assessment in serious games: Application to learning processes in crisis management. In *IEEE 7th International Conference on Research Challenges in Information Science (RCIS)*, pages 1–12. IEEE.

Uschold, M., Provine, R., Smith, S., Schlenoff, C., and Balikirsky, S. (2003). Ontologies for world modeling in autonomous vehicles. In *18Th International Joint Conference on Artificial Intelligence, IJCAI*, volume 3.

Vacek, S., Gindele, T., Zollner, J. M., and Dillmann, R. (2007). Using case-based reasoning for autonomous vehicle guidance. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4271–4276. IEEE.

Vlakveld, W. P. (2005). The use of simulators in basic driver training. In *Humanist TFG Workshop on the Application of New Technologies to Driver Training, Brno, Czech Republic. Available at: www. escope. info/download/research_and_development/HUMANISTA_13Use. pdf.*

Weevers, I., Kuipers, J., Brugman, A. O., Zwiers, J., Van Dijk, E. M., and Nijholt, A. (2003). The virtual driving instructor creating awareness in a multiagent system. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 596–602. Springer.

Zamora, V., Sipele, O., Ledezma, A., and Sanchis, A. (2017). Intelligent agents for supporting driving tasks: An ontology-based alarms system. In *VEHITS*, pages 165–172.

Zhao, L., Ichise, R., Liu, Z., Mita, S., and Sasaki, Y. (2017). Ontology-based driving decision making: a feasibility study at uncontrolled intersections. *IEICE TRANSACTIONS on Information and Systems*, 100(7):1425–1439.

Zhao, L., Ichise, R., Mita, S., and Sasaki, Y. (2014). An ontology-based intelligent speed adaptation system for autonomous cars. In *Joint International Semantic Technology Conference*, pages 397–413. Springer.

Zhao, L., Ichise, R., Sasaki, Y., Liu, Z., and Yoshikawa, T. (2016). Fast decision making using ontology-based knowledge base. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 173–178. IEEE.

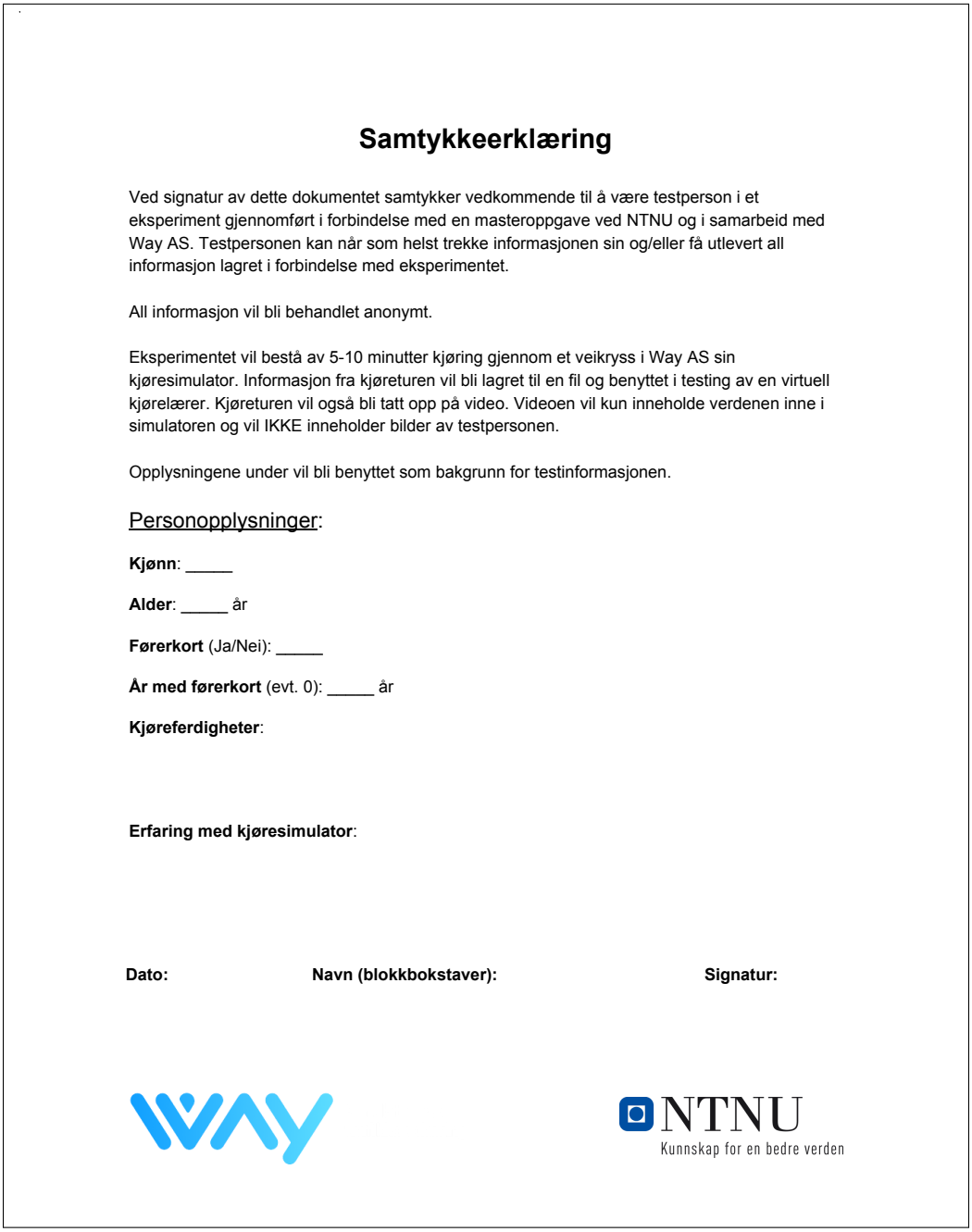
Zhao, L., Ichise, R., Yoshikawa, T., Naito, T., Kakinami, T., and Sasaki, Y. (2015). Ontology-based decision making on uncontrolled intersections and narrow roads. In *2015 IEEE intelligent vehicles symposium (IV)*, pages 83–88. IEEE.

Appendices

Appendix A

Experiment Consent Form

Below, in figure A.1, the schema test subjects would fill out and sign before participating in our experiment is given. It includes some information about the driver like their age, sex, simulator experience, number of years with a drivers license as well as how they perceive their own driving skills.



Samtykkeerklæring

Ved signatur av dette dokumentet samtykker vedkommende til å være testperson i et eksperiment gjennomført i forbindelse med en masteroppgave ved NTNU og i samarbeid med Way AS. Testpersonen kan når som helst trekke informasjonen sin og/eller få utlevert all informasjon lagret i forbindelse med eksperimentet.

All informasjon vil bli behandlet anonymt.

Eksperimentet vil bestå av 5-10 minutter kjøring gjennom et veikryss i Way AS sin kjøresimulator. Informasjon fra kjøreturen vil bli lagret til en fil og benyttet i testing av en virtuell kjørelærer. Kjøreturen vil også bli tatt opp på video. Videoen vil kun inneholde verdenen inne i simulatoren og vil IKKE inneholder bilder av testpersonen.

Opplysningene under vil bli benyttet som bakgrunn for testinformasjonen.

Personopplysninger:

Kjønn: _____

Alder: _____ år

Førerkort (Ja/Nei): _____

År med førerkort (evt. 0): _____ år

Kjøreferdigheter:

Erfaring med kjøresimulator:

Dato: **Navn (blokkbokstaver):** **Signatur:**

Figure A.1: The schema (in Norwegian) test subjects would fill out confirming that we can analyze their data from the simulator test drive, along with some attributes related to their driving, like experience with simulators and how long they have held a drivers licence.

Appendix B

Video and Log Files

Below is the URL to the data files from Experiment 2. It contains a set of folders within Dropbox (Link to Dropbox folder) with the video recording of the test subjects drive, as well as the corresponding XML log file being the basis for our proof of concept evaluation of the drive.

URL to Dropbox folder:
`https://www.dropbox.com/sh/v1z82ppo8ii83px/AABcS-sqnaZgMbImCOIz4qpea?dl=0`

Remark: Due to an oversight the driver's camera was left on for test subject number 12. Although it was specified in the contract that the videos would not include images of the test subject, he has agreed to let us use the video.

Appendix C

Proof of Concept Source Code

The code for the proof of concept can be found at GitHub. We have made a dummy user added as a collaborator to the repository containing the source code of our proof of concept implementation.
This dummy user is accessed with the credentials below:
Username: WayMasterThesis
Password: VirtualDrivingInstructor

We as the authors cannot guarantee that this dummy-user still has access to the source code developed in collaboration with Way AS in the future, as they might restrict access to it from the public. However, for now, this user should have access to the VirtualDrivingInstructor repository containing the source code of the proof of concept implementation used for the experiments in this thesis.

Christiansen & Fauske

Implementing a Virtual Driving Instructor

NTNU
Norwegian University of
Science and Technology