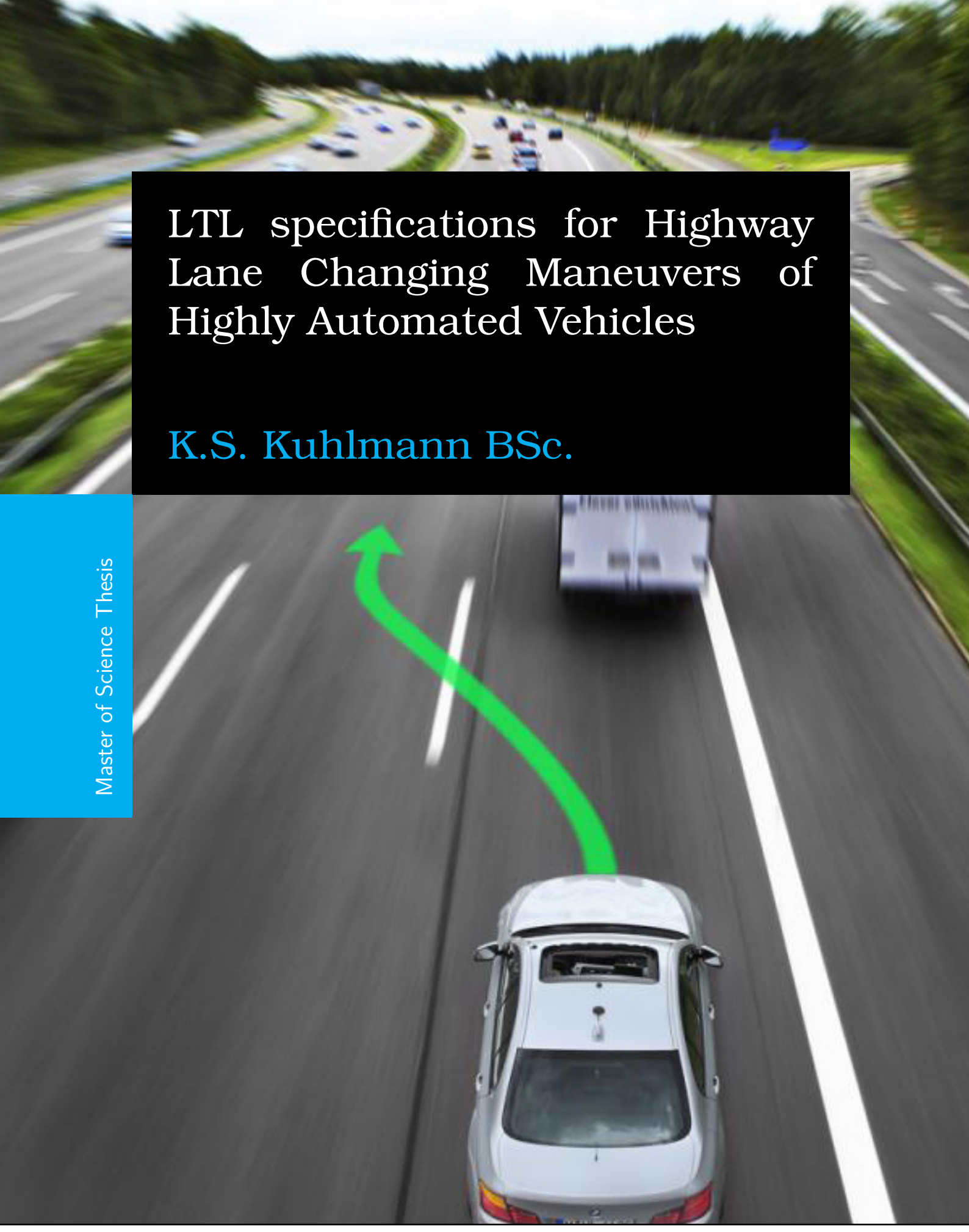

LTL specifications for Highway Lane Changing Maneuvers of Highly Automated Vehicles

K.S. Kuhlmann BSc.

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

LTL specifications for Highway Lane Changing Maneuvers of Highly Automated Vehicles

Master of Science Thesis

For the degree of Master of Science in Mechanical Engineering at Delft University of Technology

K.S. Kuhlmann BSc.

December 1, 2014

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

This research is part of the DAVI-project. For more information visit http://davi.connekt.nl/

Abstract

Since the interest in autonomous driving solutions is massively increasing, the need for good and reliable control algorithms is growing everyday. This project studies the performance of safe lane changes of a highly autonomous vehicle given the currently available perception of the environment, vehicle dynamics and desired comfort and speed requirements from the user. Also focus will be on when the vehicle decides to overtake other vehicles to move closer to its desired prescribed speed, while respecting the "rules of the road", i.e. not causing unexpected actions in relation to the other road participants. These requirements will then be converted into linear temporal logic statements for the purpose of automated synthesis of a receding horizon controller for longitudinal and lateral control of the vehicle. Thereby allowing it to make adjustments to the desired system behavior and computing a new control strategy, relatively easy and by definition, the resulting controller is formally guaranteed to meet the safety specifications at all times. Besides this search for formal specifications, a comparison is made with more conventional control techniques by reviewing a model predictive controller that was developed parallel to this project, showing its capabilities and discussing possible safety issues.

Table of Contents

Acknowledgements

As I am writing these last words to finish my thesis, the realization that this document will be the very last milestone for graduating from my study at the TU Delft, is slowly sicking in. It has been an exciting, sometimes difficult, but always very interesting journey, and my thesis project was no different.

After a long search for a graduation project, both suitable and interesting enough to work on for the long period of time, the DAVI project provided me with a chance to participate in their creation of an autonomous vehicle. Although I had no experience what so ever with formal methodology or LTL, I took the chance of exploring this completely new and maybe more computer science based approach in trying to solve a control problem. For this reason, the help of my supervisor, dr.ir. M. Mazo Espinosa Jr. was sometimes maybe even more appreciated. Therefore I want to thank Manuel for not only providing me with new insights in the sometimes very technical matter, but also for always being available for a discussion on how to proceed. Or helping to motivate me again and suggesting other possible options after we did encounter another disappointment, e.g. caused by the software issues.

I want to give my special thanks to Simone Stefano Manazza for providing me with the necessary files of their MPC algorithm, needed for comparison and even more for the late night discussions on possible safety issues and other possible flaws in their approach. And also to V. Raman and M. Fält for helping me trying to solve the software issues with TuLiP.

Of course I want to show my appreciation to all of the other colleagues at the DAVI project. And last but absolutely not least my parents, other family and friends, for not only supporting me during the complete time of my study in Delft, but also for being understanding whenever I was stressed out or easily irritable, especially during the final weeks of my thesis project.

Delft, University of Technology — K.S. Kuhlmann BSc.
December 1, 2014

Chapter 1

Introduction

The implementation of automation in ground vehicular systems has taken big steps forward in the last couple of years as a result of growing interest in both academic research as well as the automotive industry itself. Advanced driver assistance systems (ADAS), such a parking assist, adaptive cruise control and lane departure warning, already make our lives easier and safer on a daily basis. At the same time, the concept of Intelligent Transport Systems (ITS) is rapidly gaining popularity. The prospect of increased road capacity and safety, while at the same time saving fuel and increasing driving comfort are the key factors that drive the research in this area. Standardization of vehicle-to-vehicle and vehicle-to-infrastructure communication protocols and recent developments in the field of sensor and computational units, increasing performance and reducing costs, together with the increasing receptiveness of the general public, enable the next step towards fully autonomous vehicles.

Together with its partners, TU Delft initiated the Dutch Automated Vehicle Initiative (DAVI) [1] [2], to investigate and demonstrate automated driving on public roads, with automated cars sharing the road with manually driven vehicles. Not only the technical challenges and human interaction, but also the legal aspects of an autonomous vehicle, are taking within the scope of the DAVI-project. This research is part of the DAVI-project and its outcomes will be used to further improve the behavior of the DAVI-car.

1-1 Scope & Purpose

The literature survey corresponding to this thesis was the first step in the search for a control strategy for a fully automated vehicle that takes into account traffic laws, user comfort and the (most common) "rules of the road". The latter can be explained as the expectations human drivers have in a certain situations about the behavior of other vehicles. These criteria will be transformed to system design requirements in a formal language that allows for automated synthesis of embedded controller software in a way that ensures safety and liveness. In other words, the controller must be robust and stable in every possible scenario, in a constantly

changing environment, but also be able to perform the desired tasks inside that environment instead of getting stuck in a certain situation.

The focus will be on highway overtaking and merging, i.e. lane changing, scenarios. This project assumes the presence of a sensing and perception layer which provides a good representation of the environment over the full planning horizon. Hence, object detection and tracking are not in the scope of this project. Moreover, only normal non-emergency maneuvers are considered, i.e. no evasive maneuvers and off-road driving. With the knowledge acquired during the literature survey, the aim is to propose a feasible solution to the decision making problem on when the host vehicle is allowed to change lanes. The corresponding low-level path following controller that is required to perform the requested lane change is also not in the scope of this work.

1-2 Correct-by-Design

The term *correct-by-design* represents the techniques of system verification, i.e. the process of checking that a system meets its requirements, by formalizing the desired properties and allowed behaviors of a system and constructing a control strategy to enforce this behavior. Chapter 2 will introduce existing approaches to system verification providing this formal guarantee of satisfying the desired properties. These approaches use a mathematical model of the original system and analyze its correctness with respect to the requirements. Formal Methods, used in computer science as well as control oriented implementations, are discussed in section 2-1, after which a formal language that can be used to formalize system requirements is introduced in section 2-2.

Formal methods are mathematically-based techniques that provide a guarantee of system correctness and enable the user to construct a system that operates reliably, despite its complexity. The main steps include the construction of a mathematical model of the original system and then proving that the model respects the system requirements. The key elements in these procedures are specification and verification.

With the use of formal methods, the desired properties of the high level controller for lane changing maneuvers of the autonomous vehicle, as mentioned in section 1-1, can be satisfied. An automated process, developed by Wongpiromsarn et al. (more details in section 2-3), can be used to simplify the extraction of the control strategy. This will not only result in a robust and failsafe, but also an easily adaptable, controller if more or improved functionality is demanded in the future.

1-3 Problem Statement

With the above information we can summarize the general goal of this project; try to find a way to implement all of the necessary functionality needed to execute a safe lane change, given the currently available perception of the environment, vehicle dynamics and desired comfort and speed requirements from the user.

This can be formulated by the following problem statement:

Problem Statement 1. *Find formal specifications (including speed requirements, comfort and social behavior) for the purpose of automated synthesis of a controller for longitudinal and lateral control of an highly autonomous vehicle, during highway lane changing maneuvers.*

This general goal can be reached by taking the smaller steps of first introducing the necessary background information, then determining the needed capabilities of such a controller and finally expressing those functionalities as formal system specifications. The found specification could then be used to actually synthesize and test a controller using simulations. The organization of the work presented, corresponding to the mentioned approach, is given in the next section.

1-4 Outline report

After this introduction, some technical preliminaries used a basis for this project are given in chapter 2. This includes more background information on the correct-by-design methodology, an introduction to a formal logical language that will be used to express the specifications of the system, and a receding horizon framework used to reduce computational demands. This is followed by a discussion on the necessary capabilities of an overtaking controller, with special focus on highway situations, in chapter 3, which will be completed by an overview of the desired controller properties. These properties will then be converted in chapter 4, wherein the full formal specification will be presented. Chapter 5 will give the reader insight in the currently state of the art controllers that use more conventional control techniques and discusses the pros and cons of these techniques. Finally, chapter 6 will summarize the work presented, as well as discuss some recommendations and possible future work.

Chapter 2

Technical Preliminaries

This chapter will describe several technical foundations on which the current work is building. More background information and details of the correct-by-design methodology parts needed for this thesis, are discussed, as well as previous implementations. Next a brief introduction to Linear Temporal Logic (LTL) and its semantics is given in section 2-2. Section 2-3 introduces a receding horizon framework for temporal logic specifications that acts as a big inspiration and starting point for this project.

2-1 Formal Methods

Formal specification is a precise mathematical representation of a system and its requirements. Examples of such representations are differential equations, finite state machine and hybrid automata. Formal verification relies on a series of proof techniques by which the correctness of the abstracted model, subject to its specifications, can be analyzed. A formal guarantee is given, that the desired system properties hold for all possible executions of the system, provided that the actual execution of the system respects its model. This problem can be defined as the equivalence problem as follows:

Problem 2.1 (Equivalence)**.** *Given systems S_a and S_b and a notion of equivalence between systems, when is S_a equivalent to S_b, denoted by $S_a \cong S_b$?*

Many different analysis and verification problems in the design of complex systems can be formulated as instances of Problem 2.1. However, in the most important one S_a is an already designed model of the original system and S_b is a model of the specification. A positive answer to Problem 2.1 would imply that the design conforms to the specification. Another instance could be that two different models S_a and S_b have been constructed for the same original problem, one much simpler than the other. Then, a positive answer to the equivalence problem would imply that any of the two models could be used to complete the design. Hence, when S_b is much simpler to construct than S_a, it would be guaranteed that the remaining design procedure could be accomplished with greater ease by working with the simpler model

S_b. This observation places restrictions on the notions of equivalence as they need to treat as equivalent system S_a and the much simpler system S_b. In [3], several notions of equivalence are described.

System correctness can be formally verified by hand with the use of mathematical proofs, but due to the required high level of mathematical expertise, the slowness of the approach and the vulnerability to errors, the interest in automation of such proofs is growing rapidly.

2-1-1 Computer-Science Oriented Approaches

The two main categories of computer-science oriented approaches for the automated proofs are algorithmic and deductive. The algorithmic approach relies on extensively exploring the state space to check whether the desired properties of the system are satisfied. Model checkers based on the techniques described in [4] provide for such explorations. Based on the formal system requirements, stated in a precise mathematical language, every possible as well as invalid behavior of a system can be derived. Next, the model checker will check if the intersection of all the possible and invalid behaviors of the system is empty and, if this is not the case, provide an error trace. If the intersection is indeed an empty space, the model checker terminates with a positive answer. The benefits of this approach are that it is fast, completely automated and requires no human interaction. Downsides of the approach are the limitation to finite systems, as a result of the need for decidability, and that it suffers from the state explosion problem [5]. Various software toolkits using different specification languages have been developed. Chapter 4 of ...literature survey... will go into more detail about these automated model checkers.

The deductive approach uses axioms and proof rules to prove the correctness of a system. The basic idea of this approach is presenting a proof based on inductive invariants:

Definition 2.1. *If at some initial state of a system, specification φ holds and all legal successors of every φ-state are also φ-states, then φ always holds and the system is correct according to its specification φ.*

For the (partial) automation of such proofs, tools like Prototype Verification System (PVS) can be used. This packages make use of *theorem proving* and has the benefit, compared to the algorithmic model checkers, that it is not limited to finite state systems. However, it does require a skilled human interaction.

2-1-2 Control Oriented Approaches

Parallel to the studies in computer science, control scientist have developed a methodology for verifying that a control system of the form $\dot{x}(t) = f(x(t), u(t))$ or $x[k+1] = f(x[k], u[k])$ stays within a certain *safe set*. The dual of this *safety problem* is the *reachability problem* that concerns proving the existence of a trajectory that starts from an initial set and reaches another given (goal) set. The two main approaches to solve both problems are direct reachability analysis and Lyapunov-type methods. Direct reachability techniques seek to compute either a set of all states that can be reached by trajectories starting from a certain initial

set, or to compute a set of all initial states from which trajectories to a certain set of final states can be computed. The former case is called *forward reachability* and e.g. aims to prove that the unsafe set is not reachable from the initial states of the system. The latter case is called *backward reachability* and could be used to proof the existence of a trajectory to a desired final state that originates from the initial state set of the system. More details about these methods can be found in [6]. Lyapunov-type methods do not require explicit computation of reachable sets and have the ability to handle non-linearity, uncertainty and constraints directly. A Lyapunov function, satisfying certain algebraic conditions is searched which guarantees that all possible trajectories from an initial set remain in a safe set. More details about Lyapunov-type methods, e.g. using barrier certificates, can be found in [7] and references therein.

For the use of formal methods in the design of control systems, the *control problem of equivalence* formalizes the essence of design, from [3]:

Problem 2.2 (Control for equivalence)**.** *Given systems S_a and S_b and a notion of equivalence between systems, when does it exist and how can a system S_c and an interconnection relation $\mathcal{I}$ be constructed such that $S_c \times_{\mathcal{I}} S_a \cong S_b$?*

System S_b is typically a model of the specification that is to be enforced, on a given platform modeled by S_a, though the design of S_c. The main advantage of using formal methods in controller design to enforce 2.2 is that formal verification is not necessary to prove the equivalence between the designed system $S_c \times_{\mathcal{I}} S_a$ and the specification S_b. Again, in [3] methods and techniques to solve Problem 2.2 are described in more detail.

2-1-3 Hybrid Systems

The hybrid system formalism [3] provides a rich mathematical language for specification of embedded systems where computing and control components interact with physical processes. In this framework, a hybrid system is characterized by (a) a set of continuous states, (b) a finite set of locations or discrete states, (c) the set of initial states, (d) an invariant set associated with each location, (e) a set of vector fields, and (f) a set of discrete transitions between two locations. A guard set and a reset map can be derived from the set of discrete transitions between two locations. Reference [8] adds continuous and discrete input sets to this description.

Reachability analysis, Lyapunov-type methods and constraint-based approaches can be applied to verify safety properties of systems modeled in this hybrid automata framework. Model checkers such as HyTech and PHAVer, provide automated forward reachability analysis. Back reachability analysis has been applied in [9] to analyze the safety of aircraft auto land systems. Another set of frameworks, that explicitly capture the concurrency and asynchronous characteristics of distributed systems, was introduced by Lynch, based on input/output (I/O) automata and interacting infinite state machines. These automata frameworks allow for the composition of automata to make larger ones, thereby enabling modular specification of individual system components, which can be composed to describe the whole system. The most interesting, are the hybrid I/O automata (HIOA), which add a set of trajectories to describe the evolution of system states over intervals of time.

This technique is described in [10] as follows:

In the HIOA framework, the discrete behavior of a system is described by a set of discrete state transitions (*actions*). The continuous behavior is described by a set of *trajectories* that specify the behavior of the variables of an automaton with time. An execution of HIOA is described by a finite or infinite alternating sequence of trajectories and actions. A safety or invariant property $\mathcal{I}$ of an HIOA $\mathcal{A}$ is typically deduced by finding a stronger inductive invariant $\mathcal{I}' \subseteq \mathcal{I}$ and checking, through case analysis, that all the actions and trajectories of $\mathcal{A}$ preserve $\mathcal{I}'$. Specifically, the set $\mathcal{I}$ of states is an invariant of a HIOA $\mathcal{A}$ if

- (Start condition) any initial state of the system $x_0 \in \mathcal{I}$,
- (Transition condition) For any action a, if $x \xrightarrow{a} x'$ and $x \in \mathcal{I}$, then $x' \in \mathcal{I}$,
- (Trajectory condition) For any trajectory τ, if the first state of τ, $\tau_{fstate} \in \mathcal{I}$,then the last state of τ, $\tau_{lstat} \in \mathcal{I}$.

HIOA has been successfully applied in various implementations such as the safety verification of the automated highway system of the California PATH project [11] and the Traffic Alert and Collision Avoidance System (TCAS) that is used by aircrafts to avoid midair collisions [12].

2-2 Linear Temporal Logic

Temporal logic is a logical language which incorporates temporal aspects and can be used to reason about certain events in time. This time dependency could be linear, where at each moment in time there is only a single successor moment, or branching, where it has a tree-like structure where time may split into alternative courses. This section will focus on Linear Temporal Logic (LTL), which is found appropriate to formally specify various kinds of systems, especially those of concurrent software programs.

2-2-1 Semantics of LTL

Before describing LTL, we first need to give two definitions, also found in [7] and [4] to define an atomic proposition, which is LTL's main building block.

Definition 2.2. *A system consists of a set V of variables. The domain of V, denoted by $dom(V)$, is a set of valuations of V. A state of the system is an element $v \in dom(V)$.*

Definition 2.3. *An atomic proposition is a statement on system variables v that has a unique truth value ($True$ or $False$) for a given value of v. Let $v \in dom(V)$ be a state of the system and p be an atomic proposition. We write $v \Vdash p$ if p is $True$ at the state v. Otherwise, we write $v \nVdash p$.*

Also, we describe an *execution* of a system by an infinite sequence of its states. For a discrete time system, its execution can be written as $\sigma = v_0 v_1 v_2 ...$ where for each $t \geq 0$, $v_t \in dom(V)$

is the state of the system at time t.

A LTL-formula (φ) is build from atomic propositions, the Boolean connectors like *conjunction* $\wedge$, *disjunction* $\vee$, *negation* $\neg$ and *material implication* $\Longrightarrow$, and so called temporal modal operators *always* $\Box$, *eventually* $\Diamond$, *next* $\bigcirc$ and *until* $\mathcal{U}$.

Semantics of LTL: An LTL-formula is interpreted over an infinite sequence of states, i.e. a *path*, which means a path can either fulfill an LTL-formula or not. Given an execution $\sigma = v_0 v_1 v_2 ...$ and an LTL-formula φ, we say that φ holds at position $i \geq 0$ of σ, denoted by $v_i \models \varphi$ iff φ holds for the remainder of σ starting at position i. The semantics of LTL are defined as follows:

1. For an atomic proposition p, $v_i \models p$ iff $v_i \Vdash p$;
2. $v_i \models \neg\varphi$ iff $v_i \not\models \varphi$;
3. $v_i \models \varphi \vee \psi$ iff $v_i \models \varphi$ or $v_i \models \psi$;
4. $v_i \models \bigcirc\varphi$ iff $v_{i+1} \models \varphi$;
5. $v_i \models \varphi \mathcal{U} \psi$ iff there exists $j \geq i$ such that $v_j \models \psi$ and $\forall k \in [i, j), v_k \models \varphi$.

From this definition, $\bigcirc\varphi$ ("next") holds at position i iff φ holds at position $i+1$ and $\Diamond\varphi$ ("eventually") holds at position i iff φ holds at some position $j \geq i$, i.e. $\Diamond\varphi \equiv True \mathcal{U} \varphi$. Also, $\Box\varphi$ ("always") holds at position i iff φ holds at every position in σ starting at position i, i.e. $\Box\varphi \equiv \neg\Diamond\neg\varphi$. The modal operators can be combined to form new modalities. For example, $\Box\Diamond a$ ("always eventually") describes the property stating that at any moment i there is a moment $j \geq i$ at which an a-state is visited, i.e. the a-state is visited infinitely often. The dual modality $\Diamond\Box a$ with the same reasoning leads to a being "eventually forever" true, i.e. from some moment j only a-states are visited. For the set of all executions of a system, the following can be defined:

Definition 2.4. *Let Σ be the set of all executions of a system. The system is said to be correct with respect to its specification φ, denoted by $\Sigma \models \varphi$, if all executions satisfy φ, i.e. $(\Sigma \models \varphi)$ iff $(\forall\sigma, (\sigma \in \Sigma) \Longrightarrow (\sigma \models \varphi))$.*

LTL formulas can be used to define important and widely used properties of a system, such as **safety**, **reachability**, **obligation**, **progress**, **response** and **stability**. The most important ones subject to this project are safety and reachability. Safety properties are of the form $\Box\varphi$. For example, a safety property could be that the host vehicle can never go off-road; $\Box\neg$*offroad*, with *offroad* being a state with the position of the vehicle off the road. Another safety property could be that the host vehicle may never occupy the same space as another vehicle or an obstacle; $\Box\neg(v_{pos} \in O)$, with v_{pos} being the position of the host vehicle and O being a set of position states that contain obstacles. Reachability (also referred to as "guarantee") is always of the form $\Diamond\varphi$. This property ensures that φ becomes true at least once in an execution. Reaching a certain goal state is an example of this property.

Similarly as in classical logic we can define the notion of *equivalence* between two LTL formulas:

Definition 2.5. *Let $\mathcal{M}$ be the representation of the structure of a discrete, linear model of time of the form $\mathcal{M} = \langle \mathbb{N}, I \rangle$, with $I : \mathbb{N} \longmapsto 2^{AP}$ maps each Natural number (representing a moment in time) to a set of propositions AP. Then the LTL-formulas φ_1 and φ_2 are equivalent, denoted by $\varphi_1 \equiv \varphi_2$, iff $\forall \mathcal{M}, \forall i \in \mathbb{N}. \langle \mathcal{M}, i \rangle \Vdash \varphi_1 \Leftrightarrow \langle \mathcal{M}, i \rangle \Vdash \varphi_2$.*

As LTL subsumes propositional logic, equivalences of propositional logic also hold for LTL, e.g. $\neg\neg\varphi \equiv \varphi$ and $\varphi \wedge \varphi \equiv \varphi$. The temporal modalities introduce a number of additional equivalence rules, as given in figure 5.7 of [4], which make use of the duality relation between the $\Box$ and $\Diamond$ operators: $\neg\Box\varphi \equiv \Diamond\neg\varphi$ and the fact that $\Diamond$ (and thus $\Box$) can be rewritten in terms of $\mathcal{U}$, as shown above: $\Diamond \equiv True\mathcal{U}\varphi$. Hence, all temporal operators can be rewritten in terms of $\mathcal{U}$ and $\bigcirc$, as proven by [4].

2-3 Receding Horizon Temporal Planning

This section introduces a receding horizon framework for temporal logic specifications that is sufficient to describe a wide range of properties including safety, stability, progress, obligation, response and guarantee. The framework is introduced in the work of T. Wongpiromsarn [7] and will be used as a basis for the automated synthesis part of this project, with the additional capability of handling *moving obstacles* and with system specifications specifically designed for *highway situations*.

2-3-1 Overview and Preliminaries

The increasing frequency in the use of systems with strong links between computational and physical elements has caused a strong increase in the attracted attention of synthesis of correct-by-design embedded control software. The main challenge in this approach is the abstraction of systems evolving on a continuous domain into equivalent (see problem 2.1) finite state models. But maybe even more important is the challenge of dealing with the computational complexity in this synthesis of finite state automata. In particular, the synthesis problem becomes significantly harder when the interaction with the (potentially dynamic and a priori unknown) environment has to be taken into account (due to the state explosion problem [5]). A common used approach is proposed by Piterman et al. [13], who treated the problem as a two-player game between the system and the environment and proposed an algorithm for the synthesis of a finite state automaton that satisfies its specification regardless of the environment in which it operates (subject to certain assumptions on the environment that need to be stated in the specification). Piterman et al. showed, that for a certain class of specifications of the form

$$\left(\bigwedge_{i \in I} \Box\Diamond\varphi_i\right) \Longrightarrow \left(\bigwedge_{j \in J} \Box\Diamond\varphi_j\right), \tag{2-1}$$

known as the Generalized Reactivity(1) (GR(1)), such an automaton can be computed in polynomial time. However, the application of the synthesis tool is still limited to small problems. Successful applications, all with specific additions to the basic proposed approach, can be found in [14], [15], etc.

To tackle the problems with computational limitations, an approach that has earned his roots in the area of constrained optimal control can be used. This *receding horizon* approach optimizes the problem over a shorter horizon, starting from the currently observed state, implements the computed control action needed, and then optimizes again for the new observed state with the horizon shifted one time step ahead. Thereby, reducing the computational complexity by solving a sequence of smaller optimization problems. Wongpiromsarn [7] proposed an extension to this receding horizon framework to handle linear temporal logic specifications that can reduce computational complexity of the synthesis problem, while ensuring system correctness with respect to the given overall temporal logic specifications. A short summary of this framework is given in section 2-3-2 and 2-3-3. For more details and underlying proofs, the reader is referred to the original work of Wongpiromsarn [7].

2-3-2 Problem Formulation

The problem formulation considers the computational complexity issue of the hierarchical approach to attack the Planner-Controller Synthesis Problem, as given in section 6.3 and 6.4 from [7]. From the same sections, the following definitions are taken and repeated below for convenience:

Definition 2.6. ***System Model:*** *Consider a system model S with a set $V = S \cup E$ of variables where S and E are disjoint sets that represent, respectively, the set of plant variables that are regulated by the planner-controller subsystem and the set of environment variables whose values may change arbitrarily throughout an execution. The domain of V is given by $dom(V) = dom(S) \times dom(E)$ and the state of the system can be written as $v = (s, e)$ where $s \in dom(S) \subseteq \mathbb{R}^n$ and $e \in dom(E)$. We call s the controlled state and e the environment state.*

Assume that the controlled state evolves according to the following discrete-time linear time-invariant state space model: for $t \in \{0, 1, 2, ...\}$,

$$\begin{aligned} s[t+1] &= As[t] + Bu[t] + Ed[t], \\ u[t] &\in U, \\ d[t] &\in D, \\ s[0] &\in dom(S), \end{aligned} \tag{2-2}$$

where $U \subseteq \mathbb{R}^m$ is the set of admissible control inputs, $D \subseteq \mathbb{R}^p$ is the set of exogenous disturbances and $s[t]$, $u[t]$ and $d[t]$ are the controlled state, the control signal and the exogenous disturbance, respectively, at time t.

Definition 2.7. ***System Specification:*** *Assume that the specification φ consists of the following components:*

1. *the assumption φ_{init} on the initial condition of the system,*
2. *the assumption φ_e on the environment, and*
3. *the desired behavior φ_s of the system.*

Specifically, we assume that φ can be written as

$$\varphi = (\varphi_{init} \wedge \varphi_e) \Longrightarrow \varphi_s. \tag{2-3}$$

From [7] and reference therein, we find that the specification of the form 2-3 can be reduced to a subclass of GR(1) formula of the form

$$\left(\psi_{init} \wedge \Box\psi_e^e \bigwedge_{i\in I_f} \Box\Diamond\psi_{f,i}^e\right) \Longrightarrow \left(\bigwedge_{i\in I_s} \Box\psi_{s,i} \wedge \bigwedge_{i\in I_g} \Box\Diamond\psi_{g,i}\right) \tag{2-4}$$

where ψ_{init} and $\psi_{g,i}$ are propositional formulas of variables from V; $\psi_{s,i}$ is a Boolean combination of propositional formulas of variables from V and expressions of the form $\bigcirc\psi_s^t$ where ψ_s^t is a propositional formula of variables from V that describes the constraints on the transitions of controlled states; and ψ_e^e and $\psi_{f,i}^e$ are propositional formulas of variables from E.

The following aspects of specification (2-4) are noteworthy:

- The desired properties include the safety properties, $\bigwedge_{i\in I_s} \Box\psi_{s,i}$, and the progress properties, $\bigwedge_{i\in I_g} \Box\Diamond\psi_{g,i}$
- Each progress property, $\Box\Diamond\psi_{g,i}$,$i \in I_g$, specifies the set of states that the system needs to visit infinitely often, i.e. the system goal. The conjunction of these progress properties allows for multiple goals to be specified. Although, all of them need to be achieved infinitely often, the order in which they are satisfied is irrelevant.

All of the above, leads to the problem statement of the receding horizon framework that Wongpironsarn proposes, which can be defined as follows:

Problem 2.3. ***Discrete Planner Synthesis Problem:*** *Given a finite state abstraction D of a physical system and the system specification φ of the form (2-4), synthesize a discrete planner that computes a discrete plan to ensure that starting from any initial condition, φ is satisfied for any sequence of environment states. And, find a sufficient condition and receding horizon strategy that allows the synthesis to be performed on a smaller domain; thereby reducing the number of states of the automaton, while still ensuring system correctness with respect to the original LTL specifications.*

2-3-3 Receding Horizon Framework

Wongpiromsarn uses the notion of partial order to provide a measure of closeness to the goal states.

Definition 2.8. *A partially ordered set $(V, \preceq)$ consists of a set V and a binary relation $\preceq$ over the set V satisfying the following properties: for any $v_1, v_2, v_3 \in V$, (a) $v_1 \preceq v_1$; (b) if $v_1 \preceq v_2$ and $v_2 \preceq v_1$, then $v_1 = v_2$; and (c) if $v_1 \preceq v_2$ and $v_2 \preceq v_3$, then $v_1 \preceq v_3$.*

First, for each progress property $\square\lozenge\psi_{g,i}$, $i \in I_g$, a collection of subsets $\mathcal{W}_0^i, \ldots, \mathcal{W}_p^i$ of $\mathcal{V}$ is constructed such that:

- $\mathcal{W}_0^i \cup \mathcal{W}_1^i \cup \ldots \cup \mathcal{W}_p^i = \mathcal{V}$;
- $\psi_{g,i}$ is satisfied for any $\nu \in \mathcal{W}_0^i$, or in words: $\mathcal{W}_0^i$ is the set of goal states associated with the progress property $\square\lozenge\psi_{g,i}$;
- And $\mathcal{P}^i \equiv \left(\left\{\mathcal{W}_0^i, \ldots, \mathcal{W}_p^i\right\}, \preceq_{\psi_{g,i}}\right)$ is a partially ordered set defined such that $\mathcal{W}_0^i \prec_{\psi_{g,i}} \mathcal{W}_j^i$, $\forall j \neq 0$.

Then define a function $\mathcal{F}^i : \left\{\mathcal{W}_0^i, \ldots, \mathcal{W}_p^i\right\} \to \left\{\mathcal{W}_0^i, \ldots, \mathcal{W}_p^i\right\}$ such that $\mathcal{F}^i(\mathcal{W}_j^i) \prec_{\psi_{g,i}} \mathcal{W}_j^i$, $\forall j \neq 0$, that defines an intermediate goal for starting from a state in $\mathcal{W}_j^i$.

As an example, consider a simple case where $\{\nu_1, \ldots, \nu_{10}\}$ is the set $\mathcal{V}$ of states, ν_{10} satisfies $\psi_{g,i}$ and the states in $\mathcal{V}$ are organized into five subsets $\mathcal{W}_0^i, \ldots, \mathcal{W}_4^i$. The relationships between the states in $\mathcal{V}$ and the subsets $\mathcal{W}_0^i, \ldots, \mathcal{W}_4^i$ are illustrated in Figure 2-1, taken from [7], pp. 126.

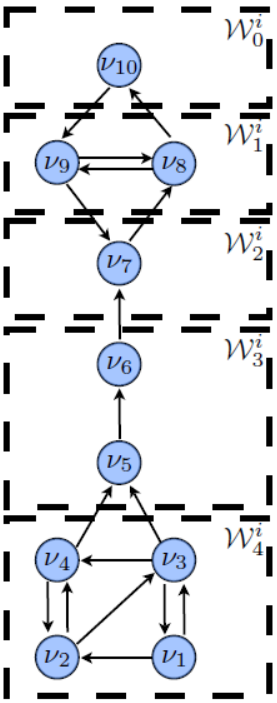

Figure 2-1: From [7]. Illustration of the receding horizon framework showing the relationships between the states of $\mathcal{V}$ and between the subsets $\mathcal{W}_0^i, \ldots, \mathcal{W}_p^i$.

The partial order may be defined as $\mathcal{W}_0^i \prec \mathcal{W}_1^i \prec \ldots \prec \mathcal{W}_4^i$ and the mapping $\mathcal{F}^i$ is defined as $\mathcal{F}^i(\mathcal{W}_j^i) = \mathcal{W}_{j-2}^i$, $\forall j \geq 2$, $\mathcal{F}^i(\mathcal{W}_1^i) = \mathcal{W}_0^i$ and $\mathcal{F}^i(\mathcal{W}_0^i) = \mathcal{W}_0^i$. Suppose ν_1 is the initial state of the system. The key idea of the receding horizon framework is to plan from the initial state $\nu_1 \in \mathcal{W}_4^i$ to any state in $\mathcal{F}^i(\mathcal{W}_4^i) = \mathcal{W}_2^i$, instead of planning from ν_1 to the final goal

state ν_{10} completely in one shot, but while taken into account all the possible behavior of the environment. Once a state in $\mathcal{W}_3^i$ is reached, replanning is done from that state to any state in $\mathcal{F}^i(\mathcal{W}_3^i) = \mathcal{W}_1^i$. This process is repeated until ν_{10} is reached.

With the above definitions of $\mathcal{W}_0^i, \ldots, \mathcal{W}_p^i$ and $\mathcal{F}^i$, we can formally define a short-horizon specification Ψ_j^i associated with $\mathcal{W}_j^i$ for each $i \in I_g$ and $j \in \{0, \ldots, p\}$ as

$$\Psi_j^i \equiv \left((\nu \in \mathcal{W}_j^i) \wedge \Phi \wedge \square \psi_e^e \wedge \bigwedge_{k \in I_f} \square \lozenge \psi_{f,k}^e \right) \implies \left(\bigwedge_{k \in I_s} \square \psi_{s,k} \wedge \square \lozenge \left(\nu \in \mathcal{F}^i(\mathcal{W}_j^i) \right) \wedge \square \Phi \right) \tag{2-5}$$

where ν is the state of the system and ψ_e^e, $\psi_{f,k}^e$ and $\psi_{s,k}^e$ are defined in eq. (2-4). Φ is a propositional formula of variables of $\mathcal{V}$ such that $\psi_{init} \implies \Phi$ is a tautology, i.e. any state $\nu \in \mathcal{V}$ that satisfies ψ_{init} also satisfies Φ. The role of Φ is to add assumptions on the initial states that need to be considered when synthesizing an automaton satisfying Ψ_j^i. These assumptions may need to be added to make Ψ_j^i realizable. More details about the role of Ψ_j^i can be found in [7].

An automaton $\mathcal{A}_j^i$ satisfying Ψ_j^i defines a strategy for going from a state $\nu_1 \in \mathcal{W}_j^i$ to a state $\nu_2 \in \mathcal{F}^i(\mathcal{W}_j^i)$ while satisfying the safety requirements $\bigwedge_{k \in I_s} \square \psi_{s,i}$ and maintaining the invariant Φ. The partial order $\mathcal{P}^i$ provides a measure of "closeness" to the states satisfying $\psi_{g,i}$. Since each specification Ψ_j^i asserts that the system eventually reaches s state that is smaller in partial order, it ensures that each automaton $\mathcal{A}_j^i$ brings the system "closer" to the states satisfying $\psi_{g,i}$. Thus, the function $\mathcal{F}^i$ defines the horizon length for these short-horizon problems. In general, the size of $\mathcal{A}_j^i$ increases with a larger horizon length, thereby again increasing computational demand, but with too short horizon length the specification Ψ_j^i is typically not realizable. All of the above leads to the formal description of the Receding Horizon Strategy:

Receding Horizon Strategy: For each $i \in I_g$ and $j \in \{0, \ldots, p\}$, construct an automaton $\mathcal{A}_j^i$ satisfying Ψ_j^i. Let $I_g = \{i_1, \ldots, i_n\}$ and define a corresponding ordered set $(i_1, \ldots, i_n)$. Note that this order only affects the sequence of progress properties $\psi_{g,i_1}, \ldots, \psi_{g,i_n}$ that the system tries to satisfy, hence it can be picked arbitrarily without affecting the correctness of the receding horizon strategy.

1. Determine the index j_1 such that the current state $\nu_0 \in \mathcal{W}_{j_1}^{i_1}$. If $j \neq 0$, then execute the automaton $\mathcal{A}_{j_1}^{i_1}$ until the system reaches a state $\nu_1 \in \mathcal{W}_k^{i_1}$ where $\mathcal{W}_k^{i_1} \prec_{\psi_{g,i_1}} \mathcal{W}_{j_1}^{i_1}$. Note that since the union of $\mathcal{W}_1^{i_1}, \ldots, \mathcal{W}_p^{i_1}$ is a set $\mathcal{V}$ of all the states, given any $\nu_0, \nu_1 \in \mathcal{V}$, there exist $j_1, k \in \{0, \ldots, p\}$ such that $\nu_0 \in \mathcal{W}_{j_1}^{i_1}$ and $\nu_1 \in \mathcal{W}_k^{i_1}$. This step corresponds to going from $\mathcal{W}_{j_1}^{i_1}$ to $\mathcal{W}_{j_1-1}^{i_1}$ in Figure 2-2.

2. If the current state $\nu_1 \notin \mathcal{W}_0^{i_1}$, switch to the automaton $\mathcal{A}_k^{i_1}$ where the index k is chosen such that the current state $\nu_1 \in \mathcal{W}_k^{i_1}$. Execute $\mathcal{A}_k^{i_1}$ until the system reaches a state that is smaller in the partial order $\mathcal{P}^{i_1}$. Repeat this step until a state $\nu_2 \in \mathcal{W}_0^{i_1}$ is reached. Note that the latter is guaranteed to happen eventually because of the finiteness of the

set $\left\{\mathcal{W}_0^{i_1}, \ldots, \mathcal{W}_p^{i_1}\right\}$ and its partial order. This step corresponds to going all the way down the leftmost column in Figure 2-2.

3. Switch to the automaton $\mathcal{A}_{j_2}^{i_2}$ where the index j_2 is chosen such that the current state $\nu_2 \in \mathcal{W}_{j_2}^{i_2}$. Repeat step 2 with i_1 replaced by i_2 for the partial order $\mathcal{P}^{i_2}$ until a state $\nu_3 \in \mathcal{W}_0^{i_2}$ is reached. Repeat this step with i_2 replaced by $i_3, i_4, \ldots, i_n$ respectively, until a state $\nu_n \in \mathcal{W}_0^{i_n}$ is reached. In Figure 2-2, this step corresponds to moving to the next column, going all the way down this column and repeating this process until the bottom of the rightmost column is reached.

4. Repeat step 1-3.

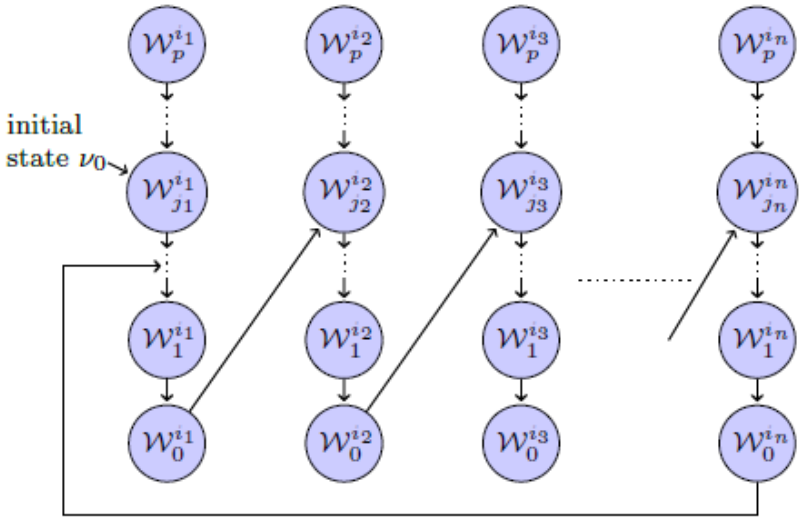

Figure 2-2: From [7]. A graphical description of the receding horizon strategy for a special case where for each $i \in I_g$, $\mathcal{W}_j^i \prec_{\psi_{g,i}} \mathcal{W}_k^i$, $\forall j < k$, $\mathcal{F}^i(\mathcal{W}_j^i) = \mathcal{W}_{j-1}^i$, $\forall j > 0$ and $\mathcal{F}^i(\mathcal{W}_0^i) = \mathcal{W}_0^i$. Step 1-4 as described above as taken to ensure that for each $i \in I_g$, a state satisfying $\psi_{g,i}$ is visited infinitely often in the execution.

Chapter 3

System Design

This chapter will evaluate the related traffic rules, safety measures, comfort criteria and social behavior in order to propose the system specifications for the automated controller synthesis. Also, attention will be given to the necessary information about the vehicle itself (internal states) and the environment, for which assumptions will have to be made to narrow down the otherwise very complex (and thus too computationally demanding) modeling situation.

3-1 Traffic Rules

3-1-1 Line crossings

International rules apply to the prohibition of lane changing with respect to the markings on the road. These rules need to be implemented in the controller to prevent a overtaking maneuver where this is not allowed. Several different lane markings are used in the Netherlands, most of which correspond to the international common standards. For example, a solid line between two lanes means switching between those lanes is prohibited (see Figure 3-1a). Solid lines also mark the most left or right-end side of the road, or the emergency lane. If there is a dashed line parallel to the solid line, switching lanes is only allowed in the direction from the dashed side to the solid side and not in the other direction (see Figure 3-1b). This situation may allow overtaking, but prevent the host vehicle to go back to the original lane. And last but not least, on- and off-ramps, as well as the situation where a highway splits into two, are marked with a wide blocked line (see Figure 3-1c). The host vehicle is only allowed to cross these special markings if it supposed to leave or enter the highway (e.g. according to route information), but not for overtaking.

3-1-2 Speed regulations

Speed limits are another important traffic law that has to be respected by the controller. Since this project focuses on highway situations, the speed limit (in normal situations, i.e.

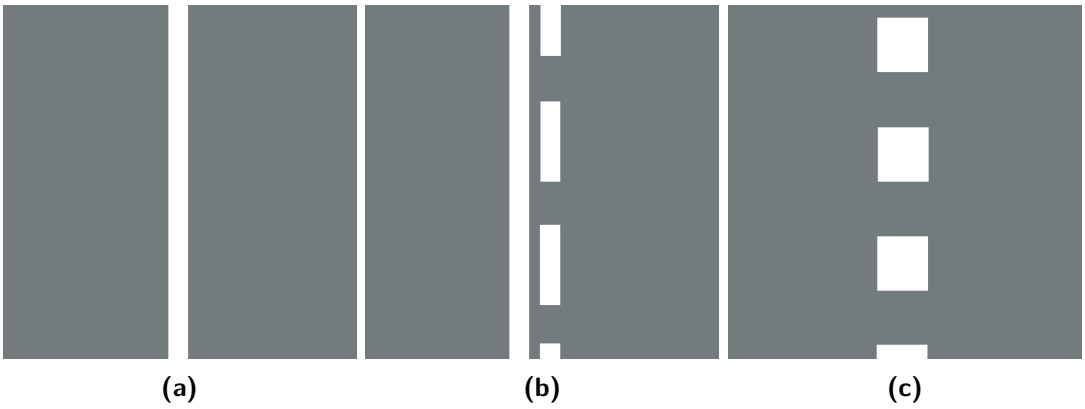

Figure 3-1: Different lane markings and their restrictions. (a) Solid line: no crossing allowed. (b) Solid/dashed line: only crossing from dashed to solid side. (c) Blocks: only crossing if route information requires this.

no traffic jam) will vary between 80km/h and 130km/h in the Netherlands. In case of implementation in a highly automated vehicle, the onboard camera can recognize the speed limit traffic signs, or use GPS-/map data to look up the speed limit for the current (section of the) road, and act accordingly. In case of a partly automated vehicle it will be another possibility to let the user set a desired speed, comparable to the principle of the cruise control system. In both cases, the maximum speed of the vehicle will be allowed to break the speed limit by a 3% margin to enable faster overtaking of other vehicles that are moving just a few kilometers per hour slower than the desired speed of the host vehicle.

The traffic laws in the Netherlands dictate a minimum speed of 60km/h on a highway under "normal driving conditions" [16]. The latter term forces several measurements or assessments about the current environment the host vehicle has to be able to execute. Fog, heavy rain, snow, but also traffic congestion, are all reasons the lower speed limit may have to be broken. Assuming that the weather conditions are normal (clear weather), another cause of having to break the lower speed limit is a traffic jam. Therefore, the lower bound of the speed limitations seems more complicated to implement. However, one could also introduce a *progress formula* that would keep the speed of the vehicle as close as possible to the speed limit (or user defined desired speed), while ensuring that this will not cause a collision with other vehicles by another formula, and then combining these two. Thereby it will be enforced that the host will always travel at maximum speed, unless there is an obstacle in the way. The progress formula that will satisfy this requirement will be discussed in chapter 4.

3-1-3 Safe distance between cars

The distance between the host vehicle and its predecessor is often called the *front gap*. To ensure the ability of the host vehicle to react safely to any kind of event in front of it, the front gap has a minimum value. The Dutch government uses a so-called "two second rule" as a rule-of-thumb the determine the minimum value of the front gap. The reason why time is used instead of distance is because of the independency of different traveling speeds. The average reaction time of an alert driver before he/she hits the brakes when something happens

is one second. At a speed of 120km/h (or 33.3m/s) the average car has a technical braking distance of 69 meters. Add to this the one second (= 33 meters) reaction time distance and the average braking distance is 104 meters. In theory, the front gap should therefore even be three seconds, but in practice the predecessor will not come to a complete standstill in zero meters as well, thus the two second gap will suffices in most situations.

Although this safety gap is a necessary precaution for a human driver, the distance to the predecessor can be significantly decreased in a high or fully automated vehicle, because the system can react much faster than a human being ever could. In fact, TNO (partner of'DAVI) has already shown that for their CACC algorithm, the gap between the leading and following vehicle can be decreased to 1.2 seconds, and even significantly further (approx. 0.2 seconds) if Vehicle-to-Vehicle (V2V) communication, enabling the preceding car to communicate when it is braking, is used. Moreover, if Vehicle-to-Infrastructure (V2I) communication is implemented, warning signals can be passed-through to cars that are still miles away from the incident, allowing them to anticipate and adapt to the situation much in advance. Keeping all of the above in mind, the minimal front gap that will be taken for the design of the controller is set to 1.2 seconds, since we consider our host vehicle to act as a stand-alone system and especially since it will need to be able to share the road with manually driven cars, as described in chapter 1.

Similar to the front gap, the gap between the host vehicle and the following vehicle is also important with respect to safety. Keeping a safe distance at the back of the host vehicle however, is considered to be a task of the following vehicle. Of course attention has to be paid to avoid blocking faster traffic, as well as creating unsafe situations. Since this behavior is part of the "social behavior", it will be further discussed in section 3-2.

3-1-4 Place on the road and Overtaking

In almost all western countries road participants are traveling at the right hand side of the road. Article 3 of the Dutch traffic law states that every vehicle also has to drive to the most right possible lane, also on the highways. Vehicles are only allowed to take over slower traffic on the left hand side and afterwards should return to the most right possible lane again. Exceptions to this rule are in the case of a traffic jam and when the host vehicle is on an on-/off-ramp, indicated with the blocked line as mentioned in section 3-1-1. In all other situations, overtaking on the right hand side is forbidden.

3-2 Social behavior

In this section, several statements about the so-called "rules of the road" will be discussed, with the goal of implementing this social behavior into the controller. This way the automated vehicle will act more human-like, improving the rate of adoption of the automated lane changing system by human drivers. At the same time, during the hybrid transition period where both automated and manually driven vehicles share the same roads, it smoothens the interaction between both, as the behavior of the automated vehicle fulfills the "normal" expectations human drivers have about other road participants.

For example, as mentioned before in section 3-1-3, it is the task of the following vehicle to maintain a safe distance to the host vehicle. But if the host vehicle gets stuck behind a slower vehicle (e.g. truck or bus) due to the inability of the system to overtake in time, the host vehicle should not commence an overtaking maneuver if this causes a vehicle in the "overtaking lane" having to brake or make an evasive maneuver. Although, this kind of behavior is rated as socially correct, a lot of human drivers might take the more egocentric approach where they just expect the faster car to brake, enabling them to overtake. As a matter of fact, this kind of aggressive driving is sitting at the top of the list of traffic annoyances in the Netherlands at the time of writing [17]. To make sure the automated vehicle will take the socially correct approach, a constraint will be introduced that switching lanes may only be executed if the speed of the host vehicle and the faster vehicle match, or can be matched before the faster car enters the safety zone at the back-side of the host.

After the overtaking of a slower vehicle is finished, the traffic rules dictate that the host vehicle has to go back to the most right lane again. However, if right after or even before this lane switching maneuver is finished, another overtaking maneuver has to be started already, this leads to unnecessary complex and unwanted situations. Moreover, it will cause discomfort for the user, as this leads to unnecessarily more lane switching maneuvers, as well as distrust of the user into the system, as a human driver would anticipate on this kind of situation by staying in the faster lane. This behavior will be implemented in the automated lane switching algorithm by using a statement that prevents the host vehicle from going back to the right lane, if another overtaking maneuver has to be initiated within 10 seconds after the lane switch is completed. The value of this time interval can be changed according to the findings during testing. To facilitate this, a sensor range of 10 seconds + the time it takes to switch back to the right lane is needed. According to the work of Bussemaker [18], the sensors of the DAVI vehicle should meet this requirement.

When the traveling speeds of both the current lane and the right lane are equal and below the desired(/set) speed of the host vehicle, the host is also allowed to stay in the current lane. The idea behind this statement is that this scenario will occur when traffic is (temporarily) denser and going back to the right lane does not have a positive effect for both the host as well as other vehicles. If traffic will clear, the current lane is expected to increase traveling speed more quickly than the right lane, thereby allowing the host vehicle to reach its desired speed earlier.

The last scenario that will be considered is the case of very dense traffic, especially in the "overtaking lane", making it difficult for the host vehicle to find a safe gap between the faster vehicles to merge into. To avoid that the host vehicle gets stuck between the slower vehicle in the right lane, the constraints for this safe gap can be softened by allowing the front gap to be smaller than the safety margin of 1.2 seconds as given in section 3-1-3. The back gap is allowed to be smaller as well, but with a minimum of 10 meters. Required for allowing this situation is that the speed of both the host vehicle and the other two vehicles match almost completely (within a 2km/h margin) and the indicator signal has been on for at least two seconds. This to avoid causing a startle to the other driver(s) and to enable the car behind the merging gap to make a little more room. The actual validity and necessity of this (rather tricky) rule has to be examined by testing.

3-3 Assumptions and constraints

This project assumes the presence of a sensing and perception layer which provides a good representation of the environment over the full planning horizon. In [18], the authors show the feasibility of this assumption is realistic and most of the sensors and computational power needed are already available today. GPS data can be used to facilitate information about the current road, e.g. speed limit, number of lanes, etcetera. Object detection and tracking are not in the scope of this project, but it is assumed that for all surrounding vehicles, position, speed and heading are known. Also, full (internal) state information of the host vehicle is assumed to be available, including the information in which lane the vehicle is driving. For the sake of simplicity, the initial aim is to model the dynamics of the host vehicle as a point mass for starters. If the computational power turns out to be sufficient, a non-linear bicycle model can be implemented. Together with the knowledge acquired during the corresponding literature survey, the aim is to propose a feasible solution to the decision making problem on when the host vehicle is allowed to (or should) change lanes and propose a trajectory that should be followed for this action. The corresponding low-level path following controller that is required to perform the requested lane change, following the proposed trajectory, is also not in the scope of this work.

Additional assumptions will be that there should be no overshoot when switching lanes. This means that the vehicle will only be in the current and/or "overtaking lane" during the lane switching procedure and will never enter the other lanes of the road. Moreover, simillar studies to this project [19] [7], found that the assumption that vehicles will not appear or disappear out of thin air, is crucial for the algorithm to synthesize a controller. This will be discussed in more detail in chapter 4.

Constraints that will ensure safety and (equally important) user comfort should be implemented in the controller as well. Standards values of the maximum longitudinal and lateral acceleration are not very clear, as different studies show different opinions. Some link the maximum acceleration values to the current speed [20], others use hard constraints [21]. There is however an ISO norm (ISO 15622) that has guidelines on standard values for maximum acceleration and deceleration and therefore, until tests to determine suitable values for these parameters have been carried out, the ISO values will be used for the controller. Hence, the maximum lateral acceleration and deceleration will be $1.5m/s^2$ and $-3m/s^2$ respectively. Higher values not only cause higher loads on the system components, but also user discomfort. One might argue that in some situations maximum acceleration or deceleration is required, but this project considers only normal, non-emergency maneuvers, thus this will not be necessary in those kind of maneuvers. The other benchmark for user comfort, as found in literature [22], is the third derivative of the (lateral) position called jerk. Jerk can be minimized by introducing a cost function that penalizes the jerk over the computed trajectories, but also in the form of a constraint that forces the (lateral) jerk to remain within certain boundary conditions. Values for lateral acceleration constraints are even harder to find, since these values are most of the time linked to the vehicle dynamics instead of considering user comfort.

3-3-1 Traffic jams

During a traffic jam, most of the traffic laws and social behavior changes. However, the focus of this project is on "normal driving conditions" only, so traffic jam scenarios will be not considered. If one would however want to implement the ability to handle traffic jams, the designed controller has to be able to make a distinction between normal traffic (which can be quiet, normal or busy) and a traffic jam (very slow movement or even complete standstill). The simplest solution would be to disallow the host vehicle to switch lanes if its own speed is below a certain threshold, e.g. 30 km/h. Although probably very effective, this measure could lead to the host vehicle being stuck at the most left lane, when it actually has to leave the highway to reach its set destination. Therefore, an exception of some sort has to be implemented that will take care the route navigation won't be interrupted. However, this project will not take this kind of scenarios into account and assumes that the route following won't be an issue. Besides the former, attention must be given to the situation where a lane simply ends, forcing the host vehicle to merge onto the next lane, e.g. when a three-lane highway converges into a two-lane highway.

3-4 Summary

This chapter reviewed all the basic and more advanced characteristics that should be implemented in the control algorithm for an autonomous vehicle for highway overtaking scenarios. Traffic rules that should be dealt with are discussed, as well as measures to increase/ensure the comfort of the passengers of the car. Special attention is given to the so-called rules of the road, or in other words human like behavior. If these latter rules would be implemented in the algorithm as well, this will significantly increase the chance and rate of a wide adoption of the overtaking algorithm, by letting the vehicle behave in a way that is expected from and by human drivers. It needs to be considered however, that the notion of what is socially accepted behavior or not may vary strongly in different countries. The nature of the automated framework that will be proposed in the next chapter will allow for quick adjustments of the control strategy, just by implementing a different list of specifications. Thereby adapting the behavior of the vehicle for different parts of the world should be quick and relatively easy.

As a recap, and to make it easier to implement the above characteristics in a systematic way, the "checklist" below summarizes all of the mentioned characteristics:

1. **Traffic Rules:**
   (a) Line crossings
      i. Do not cross a solid line.
      ii. Do not cross a solid/dashed line in the direction from the solid to the dashed side.
      iii. Do not cross a blocked line (unless route information requires it).
   (b) Speed regulations

i. Do not drive slower than $60km/h$ (unless this causes a collision).
ii. Do not drive faster than the current speed limit +3% (info from camera or GPS/map data).

(c) Safe distance between cars
i. Always keep a safe distance (dependable on the current speed) of 1.2 seconds behind the predecessor.
ii. Never occupy the same space as another vehicle/obstacle.

(d) Place on the road
i. Always stay in the middle of the lane (except when changing lanes).
ii. Always stay in the most right lane (except to overtake a slower vehicle).
iii. After overtaking a slower vehicle, go back to the most right lane.
iv. During an overtake the host vehicle can only be in the current or goal lane, i.e. no overshoot to other lanes).

2. **Comfort:**

(a) Longitudinal acceleration should be limited between $-3.0m/s^2$ and $1.5m/s^2$.
(b) Lateral acceleration should be limited between $-a_{y,min}m/s^2$ and $a_{y,max}m/s^2$.
(c) **OR:** Longitudinal/lateral jerk should be limited, or even better minimalized. (Values yet unknown)

3. **Rules of the road:**

(a) Do not start an overtaking maneuver if this causes a upcoming (faster) vehicle having to brake or even make an evasive maneuver to avoid a collision, i.e. the speed of the host must be equal to the faster car before the safety margin at the back of the host vehicle is violated.
(b) Do not go back to the most right lane if another overtaking maneuver will be necessary within 10 seconds after finishing the lane switch to the right or if the predecessor in the current lane is traveling at a speed equal or lower than the traveling speed of the right lane.
(c) Safety margins (both in front and behind host) may be violated - in case of dense traffic - if and only if the speed of the host (almost) equals the speed of the vehicles in the faster lane and the indicator light was switched on for at least two seconds.

Note that the assumptions on the environment are not included in the checklist above; the list only includes statements that restrict or require some sort of behavior of the host vehicle. While the environment is dynamical and a priori unknown, we still need to implement the assumptions on the environment for the computation of the possible scenarios the system can encounter and to be able to translate this into corresponding automata, during the synthesis of our controller. We will come back to this in the next chapter.

Chapter 4

Model Construction

This chapter will convert the required functionalities from the previous chapter into a full LTL system specification, including the model used for the vehicle dynamics, obstacle and environment behavior. A strong link to the original model specifications as proposed by Wingpiromsarn is present, with the addition of handling moving obstacles and statements that are very specific to highway situations. The beginning of this chapter introduces a simplified model, only containing very basic functionality to conduct an overtaking maneuver on a highway. This is done to first test the basic functionality of the system, before more additional rules are added to the algorithm.

4-1 Vehicle and Environment

For the modeling of the host vehicle dynamics, a point mass omnidirectional model is used. It was shown in [7] that the non-dimensional equations of motion of the vehicle are given by

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} \dot{x} \\ \dot{y} \\ \frac{2mL^2}{J}\dot{\theta} \end{bmatrix} = \begin{bmatrix} q_x \\ q_y \\ q_\theta \end{bmatrix} \tag{4-1}$$

with the following constraints on the control efforts:

$$\forall t,\ q_x^2(t) + q_y^2(t) \leq \left(\frac{3 - |q_\theta(t)|}{2}\right)^2 \text{ and } |q_\theta(t)| \leq 3. \tag{4-2}$$

Conservatively, set $|q_x(t)| \leq \sqrt{0.5}$, $|q_y(t)| \leq \sqrt{0.5}$ and $|q_\theta(t)| \leq 1$ so that the constraints are decoupled. Since only the translational (x and y) components of the vehicle state are of interest, discretizing the dynamics (4-1) with time step 0.1, leads to the following discrete-time linear time-invariant state space model

$$\begin{bmatrix} z[t+1] \\ v_z[t+1] \end{bmatrix} = \begin{bmatrix} 1 & 0.0952 \\ 0 & 0.9048 \end{bmatrix} \begin{bmatrix} z[t] \\ v_z[t] \end{bmatrix} + \begin{bmatrix} 0.0048 \\ 0.0952 \end{bmatrix} q_z \tag{4-3}$$

where z represents either x or y and v_z represents the rate of change in z. Now, let C_z be the domain of the vehicle state projected onto the (z, v_z) coordinates, restrict the domain C_z to $[zmin, zmax] \times [-1, 1]$ and partition C_z as $C_z = \bigcup_{i \in zmin+1,\ldots,zmax} C_{z,i}$ where $C_{z,i} = [i-1, i] \times [-1, 1]$ as shown in Figure 4-1.

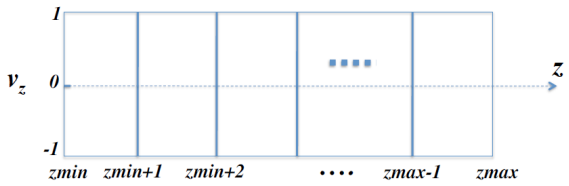

Figure 4-1: From [7], The original partition of the domain C_z.

For the basic model scenario we consider a road with two lanes, each of width 1, so we set $y_{min} = 0$ and $y_{max} = 2$. Since the vehicle dynamics are translationally invariant, without loss of generality we set $x_{min} = 0$ and $x_{max} = L$ where L is the length of the road. For each $i \in \{1, \ldots, L\}$ and $j \in \{1, 2\}$, we define a Boolean variable $O_{i,j}$ that is assigned the value *True* iff an obstacle is detected at some position $(x_o, y_o) \in [i-1, i] \times [j-1, j]$. The state of the system is therefore a tuple $(x, v_x, y, v_y, O_{1,1}, O_{1,2}, \ldots, O_{L,1}, O_{L,2})$ where $(x, v_x, y, v_y) \in [0, L] \times [0, 1] \times [0, 2] \times [-1, 1]$ is the vehicle state or the controlled state (s) and $(O_{1,1}, O_{1,2}, \ldots, O_{L,1}, O_{L,2}) \in \{0, 1\}^{2L}$ is the environment state.

4-2 Basic Functionality

In this section, the full LTL specifications needed for a basic model that can deal with the most basic overtaking scenario; one slower vehicle in the current lane in front of the host vehicle and one faster vehicle in the faster lane. As mentioned before, the goal is to let the host vehicle decide when overtaking of the slower vehicle is safe and then perform the overtaking maneuver.

4-2-1 LTL specifications

For the initial configuration, the host vehicle is assumed to be in the right lane and at least d_{obs} sectors away from any obstacle. Therefore ψ_{init} in 2-4 is defined as: for any $i \in \{1, \ldots, L\}$,

$$\left(x \in \bigcup_{k=i-d_{obs}}^{i+d_{obs}} C_{x,k} \Longrightarrow (\neg O_{i,1} \wedge \neg O_{i,2}) \right) \wedge y \in C_{y,1} \tag{4-4}$$

The following LTL formulas represent the corresponding properties that are assumed for the environment:

1. An obstacle will be detected before the host vehicle gets too close to it, i.e. when normal evasive reactions (e.g. lane switch or slowing down) are still sufficient to avoid a collision. In other words, there is a lower bound d_{popup} on the distance from the vehicle for which an obstacle is allowed to suddenly appear. The corresponding LTL formula is a conjunction of the following formula: for all $i \in \{1, ...L\}$ and $j \in \{1, 2\}$,

$$\square \left(\left(x \in \bigcup_{k=i-d_{popup}}^{i+d_{popup}} C_{x,k} \wedge \neg O_{i,j} \right) \Longrightarrow \square(\neg O_{i,j}) \right) \tag{4-5}$$

2. Because sensing range is limited by the equipped sensor and its specifications, it is assumed that the vehicle cannot detect an obstacle that is farther away than $d_{sr} > d_{popup} \geq 0$. An LTL-formula that represents this assumption is a conjunction of the following formula: for all $i \in \{1, ...L\}$,

$$\square \left(x \in C_{x,i} \Longrightarrow \bigwedge_{k>i+d_{sr}} (\neg O_{k,1} \wedge \neg O_{k,2}) \right) \tag{4-6}$$

3. The road is never permanently completely blocked. Here, an adjustment to the original system from [7] is needed, due to the ability of the obstacles to move (at different speeds). Therefore the new formula to represent this assumption is: for any $i \in \{1, ...L\}$

$$\square \left((O_{i,1} \wedge O_{i,2}) \Longrightarrow \Diamond \neg (O_{i,1} \wedge O_{i,2}) \right) \tag{4-7}$$

4. To model the speed of the obstacles, a change in relative position is considered, depending on the lane the obstacle is in. Since we consider only the scenario with a slower vehicle in the right lane and a faster vehicle in the left lane, the corresponding formulas are: for any $i \in \{1, ...L\}$

$$\square \left((O_{i,1}) \Longrightarrow \bigcirc (O_{i-1,1}) \right) \tag{4-8}$$
$$\square \left((O_{i,2}) \Longrightarrow \bigcirc (O_{i+1,2}) \right) \tag{4-9}$$

   It needs to be examined if this is the correct way to model the behavior of the moving obstacles. Another possibility could be to add more, non-controllable, environment states to the system which will contain the positions and speeds of the obstacles. This will however have a larger impact on the framework, since this is based on only boolean information about a obstacle being in a certain sector or not and not the rate in which this position changes.

Next, the desired safety properties are defined as a conjunction of the following statements:

1. Avoid collisions with other vehicles at all times. The corresponding formula, for any $i \in \{1, \ldots, L\}$ and $j \in \{1, 2\}$,

$$\square \left(O_{i,j} \Longrightarrow \neg (x \in C_{x,i} \wedge y \in C_{y,j}) \right) \tag{4-10}$$

2. The host should also stay in the most right lane unless there is an obstacle blocking it. That is, for any $i \in \{1, \ldots, L\}$,

$$\square \left((\neg O_{i,1} \wedge x \in C_{x,i}) \Longrightarrow (y \in C_{y,1}) \right) \tag{4-11}$$

Finally, the overall progress properties need to be defined. For now, the only two main goals are:

1. To reach the end of the road:

$$\square \Diamond \left(x \in C_{x,L} \right) \tag{4-12}$$

2. And to keep the desired speed $v_{x,desired}$, set either manually by the user or automatically via map or sensor data:

$$\square \Diamond \left(v_x \in V_{desired} \right) \tag{4-13}$$

   where v_x is the speed in longitudinal direction and $V_{desired}$ is the domain defined by the desired speed with a margin of $+/-1\%$, i.e. $[0.99 v_{x,desired}, 1,01 v_{x,desired}]$. The margin of $+/-1\%$ is introduced to act as a virtual "damper" on the acceleration, preventing large input signals. Caution has to be taken here in not making this margin too big as this will probably result in oscillatory behavior of the longitudinal speed, which may lead to uncomfortable situations for the passengers of the car.

4-2-2 LTL checking and synthesis

Now that we have a full system specification in LTL, this section focuses on several tools for the synthesis of controllers with the use of formal specifications, especially in the form of LTL propositions. These tools, among others, were found during the literature survey that was conducted prior to this project. Pessoa, LTLcon and TuLiP will be compared by their capabilities and the choice for using TuLiP for the synthesis of our controller will be explained. Afterwards, the software issues encountered while working with TuLiP will be discussed.

Pessoa

Pessoa is a MATLAB toolbox, created at UCLA's CyPhyLab, based on the theory of [3], for the synthesis of correct-by-design embedded control software. It uses the notion of *approximate bisimulation*, which allows for the replacement of differential equations, representing a physical system, into an equivalent finite-state machine, i.e. a symbolic model of the system.

Next, the user can specify the required system properties, e.g. (physical) limitations/boundaries on certain states or a final goal, which will then be used to synthesize a controller for the system. The resulting controllers are also finite-state and are guaranteed to enforce the control specifications on the original physical system. Although Pessoa sounds like a very promising tool for the synthesis of our controller, unfortunately the current version of Pessoa (1.4) is lacking full LTL support at this point [23].

LTLcon

At CISE, the group of M. Kloetzer and C. Belta developed a MATLAB toolbox named *LTLcon*, which considers the following problem [24]: Given a linear system $\dot{x} = Ax + b + Bu$, with polyhedral control constraints U, and given a specification in terms of an arbitrary LTL-formula φ over an arbitrary set of linear predicates in x, find initial states and a feedback control strategy so that the corresponding trajectories of the closed loop system satisfy formula φ, while staying inside a given full-dimensional polytope P. The underlying methodology is subtracted from [3], as well as the used semantics. LTLcon handles affine control systems and arbitrary LTL specifications.

A downside of LTLcon is that the found solution for the problem above is not complete because of the conservativeness of the approach taken, i.e. if LTLcon can not find a solution, it does not mean there is none. This conservativeness expresses itself in several ways; LTLcon looks for whole sets (full-dimensional polytopes) of initial states instead of investigating isolated ones and also restricts its attention to affine feedback controllers, instead of allowing any type of controllers. The positive side of this approach is that working with sets of states instead of isolated states provides robustness with respect to uncertainty in initial conditions and measurement of the current state. Another plus side of the approach taken is the use of an iterative procedure to construct the set of feasible subpolytopes, while at the same time taking into consideration new constraints. This way it ends up with testing a much smaller number of proposition combinations [24].

TuLiP

The Temporal Logic Planning (TuLiP) toolbox, developed at Caltech, is a collection of Python-based code for automatic synthesis of correct-by-construction embedded control software. The plant, i.e. the physical component regulated by the controller, can consist of continuous and discrete components. In contrast to Pessoa and LTLCon, TuLiP models the a priori unknown environment as an adversary. To deal with the thereby caused state explosion problem (as all admissible environment profiles need to be taken into account in the synthesis process), the receding horizon framework as described in section 2-3 is integrated to reduce the computational complexity of the synthesis. The working principle of TuLiP, as described in [25], is as follows:

TuLiP models the embedded control software synthesis problem as a game between the plant and the environment. Given the model of the plant and specification φ in LTL, it automatically synthesizes a controller that ensures system correctness with respect to φ for any admissible environment, if such a controller exists. If φ is unrealizable, TuLiP provides counter

examples, i.e. initial states starting from which the environment can falsify φ regardless of the controller's actions.

Software difficulties

Eventually TuLiP was selected to be used for the synthesis of our controller for several reasons; it is open-source so no licenses are needed, has full LTL support, has the receding horizon framework discussed earlier already implemented and even comes with an example script file of an autonomous vehicle traveling on a straight road. Unfortunately, when trying to install and work with TuLiP, after the basic model specifications where composed, several problems surfaced. First of all, advanced knowledge of the Linux operating system (in our case Ubuntu) is needed to make sure the required packages, e.g. Python, NumPy, SciPy, CVXopt, etc., are or will be installed properly. As it turns out, version compatibility is a large problem here. Older version of packages don't work with newer versions of other packages and vice versa. Moreover, TuLiP itself is available in several versions and distributions as well, making it even harder to find the correct version for all of its dependencies. And because some of these dependencies again need several other packages to run properly, one can easily get lost in the very large pool of available distributions.

Besides the version mismatching issues discussed above, another major difficulty comes with the fact that the receding horizon framework was developed and implemented using version 0.4a of TuLiP. In the latest release, version 1.0b, the complete program structure is changed, functions have been given different names and even parts of the functionality are removed where others were added. Therefore the example for the autonomous car that came with the 0.4a distribution needed to be rewritten completely. Together with the original developers we have tried to compute a new version of the script file for the autonomous car for the latest version of TuLiP, but since the functionality is significantly changed, this turned out to be impossible to achieve within the time limits of this project. Therefore, we returned to and tried to install the older distribution of TuLiP (version 0.4a). Again with the help of the developers, this was done successfully, but when trying to run some of the examples that came with the distribution of TuLiP a lot of different errors popped up. Most of them were caused by missing dependency packages and disappeared after the correct version of those packages were installed. However, several packages needed were no longer available or had been upgraded to newer versions that introduced new errors by syntax mismatches, which could not be fixed.

Unfortunately, these issues made it impossible to test our LTL specifications using TuLiP. The fact that there are very few, maybe even none, other tools available at this point that can automatically synthesize a controller from the corresponding LTL specification, made it impossible to synthesize our controller within the time scope of this project. Although this is a very disappointing conclusion, we wanted to at least check the satisfiability of the composed LTL formulas of the basic model presented in section 4-2-1, i.e. make sure that their are no conflicting statements making it impossible to satisfy all of the specifications in the first place. Since their are in fact a lot of tools available that let you check your LTL specifications, this goal indeed turned out to be achievable and the process is described below.

Spot is an object-oriented model checking library written in C++ that is co-developed by

people at LRDE and LIP6. It offers a set of building blocks to experiment with and develop custom made model checkers. One of these blocks is the function **ltl2tgba**, which can be used to translate LTL formulas into Büchi automata, as described in [26], [27] and [28]. The translation algorithms were implemented in an online toolbox, that can be found via the Spot-website [29]. To check the satisfiability of our set of LTL formulas of the basic overtaking model, the formulas were entered in the online toolbox (using the default settings) and the "Büchi Run" option for the desired output was chosen. Figure 4-2 shows that indeed an accepting run was found, thereby proving that our combination of LTL formulas entered could be translated to a Büchi automaton and thus is satisfiable.

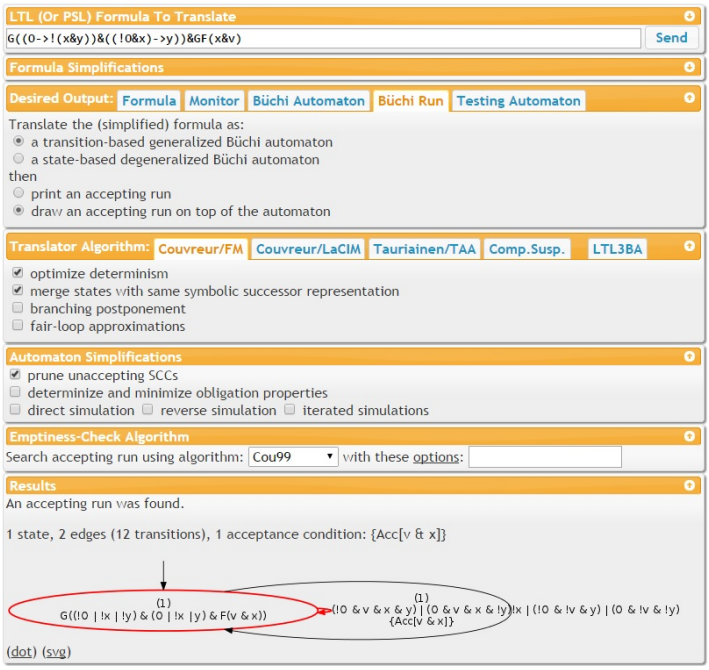

Figure 4-2: The resulting accepted Büchi run after executing the online **ltl2tgba** tool with the basic model formulas as the input.

4-3 Enhanced functionality

In the previous section, we concluded that it is not possible to test the LTL specifications of our basic model, due to the mentioned software issues. Nevertheless, to enable future research on the topic to implement the functionality as proposed in chapter 3, the following section will introduce several LTL statements which could be used to implement these functionalities. Although some of the formulas might need some more attention or testing, the aim is to give the reader an idea of the form and capabilities of these formulas and to show that the proposed properties could indeed be implemented using LTL. The checklist from section 3-4 will be repeated below and for each property a corresponding possible LTL formula will be proposed. Newly introduced variables, which can be internal states or inputs of/to the system that are assumed to be known at all times, will be described if necessary. One important new parameter that is introduced is $L_{left/right}$ which will act as a *"green flag"* for starting a lane

changing maneuver, for example the event of turning on the indicator signal (left or right). This green flag can also be used as an interaction parameter that activates the communication between the high level decision making controller and the low level continuous controller that makes sure the vehicle will actually follow the trajectory as demanded by the high level controller.

Adding these statements to the basic model specification from section 4-2-1 could be done during future research:

1. **Traffic Rules:**
   (a) Line crossings
      i. *Do not cross a solid line.*
      ii. *Do not cross a solid/dashed line in the direction from the solid to the dashed side.*
      iii. *Do not cross a blocked line (unless route information requires it).*

      These three properties can be combined into the following LTL formula:

$$\square\left((\neg L_{left}\mathcal{U}M_{left})\wedge(\neg L_{left}\mathcal{U}M_{right})\right) \tag{4-14}$$

      where $L_{left/right}$ are as described above and $M_{left/right}$ will be Boolean parameters which (dis-)allow a lane change to left/right depending on the lane markings present. The latter can be set by either the camera sensor or GPS map data.

   (b) Speed regulations
      i. *Do not drive slower than* $60km/h$ *(unless this causes a collision).*
      As mentioned earlier in section 3-1-2 there is no need to implement a formula for this property, as it will already be satisfied (whenever possible) because of the overall goal to keep the speed as close to the maximum speed as possible, as defined by eq. (4-13).

      ii. *Do not drive faster than the current speed limit (info from camera or GPS/map data) +3%.*
      The corresponding formula for this is:

$$\square\left(v_x < v_{max}\right) \tag{4-15}$$

      where v_x is the longitudinal speed of the host vehicle and v_{max} is the current speed limit + 3%.

   (c) Safe distance between cars
      i. *Always keep a safe distance (dependable on the current speed) of 1.2 seconds behind the predecessor.*
      Although avoiding a collision will already be ensured by the next bullet point,

this requirement introduces an addition safety layer. This will make sure the distance between the host and its predecessor will not only be safer in case of unexpected behavior, but will also add to the comfort of the passengers of the car. The corresponding formula for this requirement is:

$$\Box\left(H_{t,x} > s_{t,x}\right) \tag{4-16}$$

where $H_{t,x}$ is the time headway in seconds and $s_{t,x}$ is the safety margin in front of the host (in this case 1.2 seconds). The time headway can simply be calculated as the difference in position between the host and its predecessor divided by the current longitudinal speed.

ii. *Never occupy the same space as another vehicle/obstacle.*
In other words, avoid a collision with another vehicle. This is of course already incorporated in the basic model by eq. (4-10).

(d) Place on the road

i. *Always stay in the middle of the lane (except when changing lanes).* This will in fact require a more detailed grid than the one used for our basic model. A corresponding formula for this property could be:

$$\Box\left(L_{left/right} \vee \left(\neg L_{left/right} \wedge \Diamond\Box(y \in G)\right)\right) \tag{4-17}$$

where y is the current lateral position of the host vehicle and G is the goal set of positions in the target lane that are considered the center. Note that this can also be implemented as an overall goal:

$$\Diamond\Box\left(y \in G\right) \tag{4-18}$$

Here y is the lateral position of the vehicle; and G is a set of lateral coordinates, corresponding to the complete center of the current goal lane and a number of lateral position grid points the vehicle is allowed to deviate from the center of the lane. Increasing the latter will allow for more in-lane lateral movement, which could be necessary, e.g. to avoid a collision. The best way to implement this property is not clear at this point, hence great caution is advised when implementing it.

ii. *Always stay in the most right lane (except to overtake a slower vehicle).*
This property is also already captured in the basic functionality model by eq. (4-11).

iii. *After overtaking a slower vehicle, go back to the most right lane.*
This one is actually the same as the previous one and will be covered by eq. (4-11).

iv. During an overtake the host vehicle can only be in the current or goal lane, i.e. no overshoot to other lanes. Again a difficult property to implement, but together with the reasoning on the "stay in the middle of the lane" property a corresponding formula could be:

$$\square \left(L_{left/right} \implies (\neg (y \in G) \mathcal{U} \square (y \in G)) \right) \tag{4-19}$$

stating that once a lane switch is started ($L_{left/right}$) and the target lane center set G is entered, it is never left again.

2. **Comfort:**

(a) *Longitudinal acceleration should be limited between* $-3.0 m/s^2$ *and* $1.5 m/s^2$.
This property can be implemented using the following formula:

$$\square \left(a_x \in S_{a,x} \right) \tag{4-20}$$

where a_x is the longitudinal acceleration of the host vehicle and $S_{a,x}$ is the set of allowed longitudinal accelerations. In this case, $S_{a,x}$ will be $[-3.0, 1.5]$.

(b) *Lateral acceleration should be limited between* $-a_{y,min} m/s^2$ *and* $a_{y,max} m/s^2$. *(Values yet unknown)*
Very similar to the previous property, this can be represented with the following formula:

$$\square \left(a_y \in S_{a,y} \right) \tag{4-21}$$

where a_y is the lateral acceleration of the host vehicle and $S_{a,y}$ is the set of allowed lateral accelerations. In this case, $S_{a,y}$ will be $[-a_{y,min}, a_{y,max}]$.

(c) ***OR:*** *Longitudinal/lateral jerk should be limited, or even better minimalized. (Values yet unknown)*
Again, this can be implemented by the same kind of formula:

$$\square \left(j_y \in S_{j,y} \right) \tag{4-22}$$

where j_y is the lateral jerk of the host vehicle and $S_{j,y}$ is the set of allowed lateral jerk values.

3. **Rules of the road:**

(a) *Do not start an overtaking maneuver if this causes a upcoming (faster) vehicle having to brake or even make an evasive maneuver to avoid a collision, i.e. the speed of the host must be equal to the faster car before the safety margin at the back of the host vehicle is violated.*
This behavior can be modeled using the following formula:

$$\square \left(L_{left/right} \implies (\square (F_{t,x} \geq f_{t,x}) \wedge \Diamond (v_x \geq v_{behind})) \right) \tag{4-23}$$

where $F_{t,x}$ is follow up time (i.e. the free space in seconds behind the host vehicle); $f_{t,x}$ is the set safety margin behind the vehicle; and v_{behind} and v_x are the longitudinal speeds of the follow up car in the faster lane and the host vehicle respectively.

(b) *Do not go back to the most right lane if another overtaking maneuver will be necessary within 10 seconds after finishing the lane switch to the right or if the predecessor in the current lane is traveling at a speed equal or lower than the traveling speed of the right lane.*
This behavior can be modeled using the following formula:

$$\Box \left(\neg L_{left/right} \vee \left(L_{left/right} \wedge (T_{next} \geq T_{min}) \wedge (v_{pre} > v_x) \right) \right) \tag{4-24}$$

where T_{next} is the time until the next overtaking maneuver would take place, T_{min} is the minimum time between going back to the right lane and having to start another overtaking maneuver and v_{pre} and v_x are the longitudinal speeds of the preceding car in the current lane and the host vehicle respectively. Note that this property will demand a longer sensor range than the original model as it needs to be able to look further ahead to predict when another overtaking maneuver would be needed. This prediction is also highly dependable on the predicted (accelerative) behavior of the obstacle vehicle in the right lane. It could for now be assumed that during the prediction horizon the predecessor in the right lane would maintain a constant speed, since object tracking and prediction are not in the scope of this project.

(c) *Safety margins (both in front and behind host) may be violated - in case of dense traffic - if and only if the speed of the host (almost) equals the speed of the vehicles in the faster lane and the indicator light was switched on for at least two seconds.*
This behavior can be modeled be adjusting the formula from 1.c.i.) to a conditional formula, so the margin could be violated if the criteria of equal speeds is met. The new formula for this could be:

$$\Box \left((H_{t,x} > s_{t,x}) \vee (\neg (H_{t,x} > s_{t,x}) \wedge (v_x = v_{fast})) \right) \tag{4-25}$$

where $H_{t,x}$, $s_{t,x}$ are as described above and v_{fast} is the traveling speed of the "faster/overtaking" lane. Note here that for better safety, it may be wise to adjust the $s_{t,x}$ values instead of ignoring them completely.

Chapter 5

Comparison vs conventional control

Finding robust, safe and real time feasible solutions for the problem of controlling an autonomous vehicle is a very popular area of research in both the academic world as well as the automotive industry itself. Proofs of this development are not only the very well-known and highly autonomous Google cars [30], but also the fact that more and more Advanced driver assistance systems (ADAS) features, such a parking assist, adaptive cruise control and lane departure warning, are implemented in their production models by a growing number of car manufacturers every day. When we specifically look at systems that are able to navigate the car autonomously through highway traffic, there are still a lot of promising results already shown by for example BMW [31], Mercedes [32] and Volvo [33] and many others. This proofs that they are all investigating these future technologies. Although publications are very few, due to the corporate classified information, many of them probably take very different approaches as there are many possible ways and control techniques that might be able to solve the problem. As a matter of fact, within the DAVI project, another group of researchers very recently proposed an algorithm based on Model Predictive Control (MPC) techniques, that is capable of controlling a car on the highway, being able to overtake other vehicles, as well [34]. For the sake of comparison to the proposed LTL model from chapter 4, this chapter will review their proposed algorithm by introducing the conventional MPC methodology and describing the basic working principles of the algorithm (section 5-1). Experiments that have been done to test the algorithm in different scenarios and the corresponding results will be presented in section 5-2 to give the reader an idea of what the algorithm is capable of. Finally, section 5-3 will discuss their design choices and possible areas of concern, with respect to safety and computability issues.

5-1 Introduction MPC algorithm

For a better understanding of the proposed algorithm, a brief introduction to model predictive control is given. For more details on the capabilities of MPC, the reader is referred to [35]. The behavior of many dynamic systems can be represented by the following update law:

$$x[k+1] = Ax[k] + Bu[k] \tag{5-1}$$

where k is an instant in time, x is the state of the system, u is the input signal, and A and B are system matrices defining the system dynamics.

The main principle of MPC is to compute a control signal to minimize an objective function J, which is usually a function of the (predicted) system state $x(k)(= 1)$ or the control inputs. The system model above is then used as a constraint in the optimization problem, along with optional additional constraints on again system states or inputs. The MPC algorithm is usually set up to compute the optimal control signal over some period of time (N steps in the future), but only the control signal at the first time step is applied before the optimization is re-solved, using the updated system state, and a new control signal is generated. This *receding horizon* approach was already briefly discussed in section 2-3-1. A more detailed introduction and applications of receding horizon MPC can be found in [36], [35] and [37] respectively.

A simple example for a cost function J is the GPC performance index that is described in [36]:

$$J(u,t) = \sum_{j=N_m}^{N} |\hat{y}_p(k+j|k) - r(k+j|k)|^2 + \lambda^2 \sum_{j=1}^{N_c} |\Delta u(k+j-1)|^2 \tag{5-2}$$

where $r(k)$ is the reference trajectory, $\Delta u(k)$ is the process control increment signal, $\hat{y}_p$ is the predicted process output signal, N_m is the minimum cost horizon, N is the prediction horizon, N_c is the control horizon and λ is the weighting on the control signal. Since this is just an example on how the structure of J may look like, we will not explain all the different parameters and their meaning here. For that, the reader is referred to the above citations.

The key advantages of MPC, which make it a very popular control methodology for e.g. industrial purposes, are: the ability to handle multi-variable processes with large time-delays, non-minimum-phase behavior and/or unstable poles; the easiness of the concept as well as the tuning, simply by chancing the weights of the different elements in the cost function; the fact that MPC can handle (hard) constraints; and that it can handle large structural system changes, e.g. actuator failures, making it very robust and reliable in terms of stability, despite of possible modeling errors. This last property is caused by the fact that for every time step, the optimization is repeated with the currently observed state as an input. Hence, undesired results or behavior caused by the last control inputs can be brushed away by the newly calculated control inputs to the system.

Making use of the model predictive control framework, the data structure of the algorithm proposed in the work of Manazza and Gottardis [34] is developed using MATLAB/Simulink and is composed by four main elements described below. The colors of the different blocks, as shown in figure 5-1, are mentioned in their corresponding paragraph titles.

Off-line trajectory table (blue)
In order to reduce the computational costs, the trajectory generation is done off-line. Corresponding to the known lane width (3,5 meters), the trajectory finding problem can be

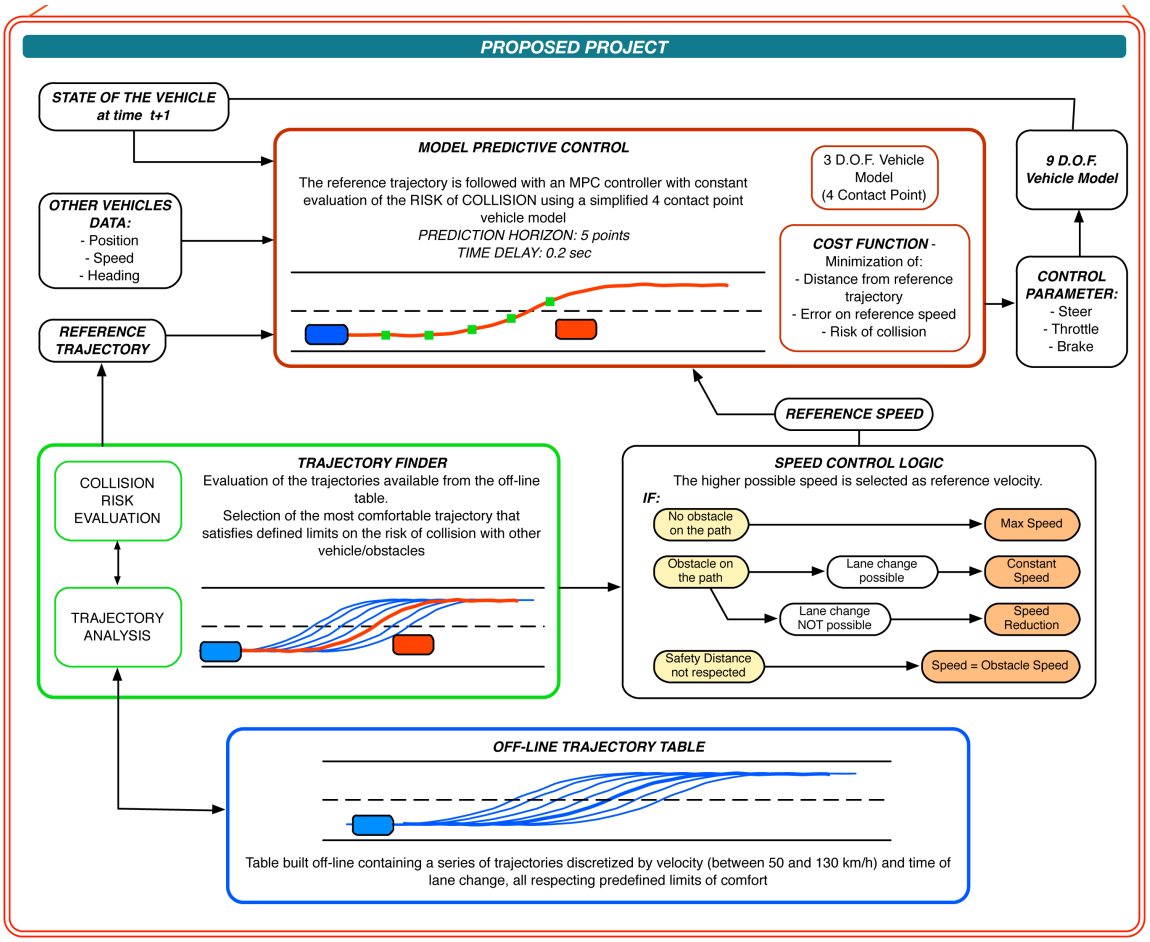

Figure 5-1: From [34]. The modeling structure as proposed by Manazza and Gottardis.

simplified by considering a fixed lateral displacement during a lane switch. Trajectories can then be computed by changing the velocity during the overtaking maneuver (50-130km/h with intervals of 5km/h) and the time needed to complete a lane switch (1-10 seconds with intervals of 0.5 seconds). A simple driver model is applied to the trajectories to estimate the lateral acceleration and the trajectories that do not respect the predefined limit are discarded. Using the current speed and the time needed to complete the lane change as inputs, the corresponding trajectory for the evaluation of the collision risk and comfort level can be found.

Trajectory finder (green)
The algorithm to search and select the lane change trajectory operates online and basically has two main functions. If there are no vehicles or obstacles on the lane where the vehicle is traveling, the algorithm builds a straight trajectory that follows the actual lane (remaining on the right lane if possible). If instead there is an obstacle detected and the estimated time-to-collision (TTC) is shorter than 15 seconds, the research of a lane change maneuver is performed. To search for the lane change trajectory the current speed of the host is used to access the off-line table, obtaining a series of trajectories that respect the limits of comfort. Starting from the slowest and most comfortable trajectory, an estimation of the risk

of collision on that maneuver is computed. If the value of risk computed is acceptable and thus the lane change is reasonably safe, the lane change trajectory is imposed as reference trajectory for the vehicle. Otherwise, the trajectory is discarded and a slightly faster and less comfortable lane change is evaluated. If none of the trajectories for the current velocity satisfy the safety condition, the speed of the vehicle is reduced and the process is repeated with the new velocity. If no safe lane change maneuver is possible, because of the traffic condition or presence of another obstacle on the road, the speed is reduced until it becomes equal to the speed of the vehicle in front and the research of the lane change trajectory continues, until a trajectory is available (e.g. due to a changed traffic situation). For the reversed situation, where the host has already overtaken a slower vehicle and needs to go back to the right lane, an inverse of the trajectories from the look up table is used. To decide whether or not the vehicle is allowed to move back to the right lane, again the TTC is calculated but this time for all of the vehicles in front of the host (i.e. all lanes are considered). When the TTC is smaller than the threshold of 15 seconds, the most comfortable trajectory (i.e. the one with the largest time to complete the lane change) with respect to the current speed is selected. If, in all cases, a suitable lane changing trajectory is found and set as the new reference trajectory, the trajectory finder is deactivated until the lane change is completed.

Model predictive controller (orange)
To follow the selected reference trajectory a non-linear model predictive controller has been designed, with the aim of making the vehicle being able to follow the planned path, but also to deviate from that if necessary in case of unexpected dangerous situations. The controller applied uses a prediction horizon of five points with a time discretization between the points of 0.2 seconds. A simplified 3-DOF vehicle model is used to estimate the position of the vehicle when certain control parameters (steer, throttle and brake) are applied. The system finds the series of control parameters that minimize a cost function for the five prediction points, and applies the first control parameters to move to the next iteration. The cost function to be minimized includes terms for:

- the distance between the center of gravity of the vehicle and the reference trajectory, evaluated in lateral direction only;
- the error on the speed, calculated as the difference between the reference speed selected and the actual speed of the host vehicle;
- the risk of collision with other obstacles or vehicles, which has to be estimated in every moment in time to deal with possible dangerous situations due to the behavior of other vehicles.

In addition, the cost function also contains the control parameters u (steer, throttle and brake) and the variation of the control parameters Δu (first derivative of u). For the simulation and testing of the algorithm, the output control parameters computed by the MPC are passed through as inputs to a 9-DOF vehicle model that simulates the real vehicle. The resulting

"real" state of the vehicle, obtained with this complex vehicle model, is given as an input to the MPC block at the next iteration.

Speed control logic (black)
The speed control logic function block is used to select the reference speed for the cost function of the model predictive controller. Depending on the scenario, the position and behavior of the other vehicles and on the safety distance, the reference speed is selected and/or changed, according to the following logical statements;

- If there is no obstacle on the path → Maximum speed (current speed limit or in standard case 130km/h).
- If there is an obstacle either,
  - find a possible lane change for the current speed → Keep constant (current) speed during the maneuver;
  - **OR:** if no possible lane change for the current speed can be found → Reduce the reference speed by 5km/h.

Although the collision risk term in the MPC cost function would be sufficient to prevent a collision under normal conditions, it does not demand to keep a safe distance to other vehicles, in case of a critical situation, for example excessive braking by the other car. That is why, to deal with these potential critical/dangerous scenarios, an addition layer of safety is introduced in the speed logic block by adding a function, which constantly checks if the fifth point of the prediction horizon of the MPC will enter a so-called *safety area*. This safety area is a rectangular shaped space behind the predicted future position (at the next time step) of vehicles detected in the scenario. When the fifth point of the prediction horizon enters the safety area, the reference speed is set to match the speed of the corresponding obstacle. The size of the safety area corresponds to a traveling distance of 1 second and, as mentioned before, the MPC uses a prediction horizon of five points, distantiated 0.2 seconds from each other. Hence, the fifth point of the prediction horizon enters the safety area if the distance between the host and the other vehicle becomes smaller than approximately 2 seconds, depending of the velocity of the host. The ability to keep a larger distance to the preceding vehicle would require a larger prediction horizon, but this would also significantly increase the computational demands of the algorithm.

The information as presented above is believed to be sufficient for the basic understanding of the working principles of the proposed algorithm, needed for evaluation and comparison with our LTL specifications. More details, as well as the complete cost function implemented in the MPC, can be found in the original work of Manazza and Gottardis [34].

5-2 Testing and results

In this section, some of the experiments conducted to test the MPC controller algorithm will be shown and quickly discussed, just to give the reader an impression on the capabilities and

the behavior of the algorithm in different scenarios. From the original nine scenarios that were tested in the original work (see table 5-1), just two are selected for repetition below; scenario 2, showing a double overtaking maneuver, and scenario 5, which describes a critical/dangerous situation. All of the original scenarios were tested and visualized by the use of the PreScan package for MATLAB/Simulink [38].

Scenario Number	What happens in the scenario
1	fixed obstacle appears suddenly in the sensor range
2	double overtake of two moving obstacles
3	little oscillations of the car in the side lane
4	big oscillation of the car in the side lane
5	sudden hard braking of the leading car
6	sudden lane change of the vehicle in the side lane
7	unexpected traffic jam
8	general traffic situation
9	obstacle vehicle entering the highway

Table 5-1: Scenarios tested in the original work of Manazza and Gottardis

5-2-1 Scenario 2: Double overtake of moving obstacles

Scenario 2 is chosen for evaluation for it is the closest to the most basic functionality of the algorithm: a simple overtake of a moving vehicle. The only difference from a scenario with just the host vehicle and a slower vehicle driving in the same lane, is the addition of another slower moving vehicle in the middle lane of a three lane highway, which will be overtaken as well before the host will return to the most right lane. This scenario is initialized with the host vehicle traveling at $v_{host} = 36m/s$ (or $130km/h$) and both of the obstacle vehicles traveling at $v_{obs} = 28m/s$ (or $100km/h$). The obstacle vehicle in the right lane is initially positioned 125 meters away and the one in the middle lane slightly further at 150 meters.

As soon as the TTC value drops below 15 seconds, the trajectory finder searches the look-up table for a new reference trajectory that minimizes the collision risk, starting from the most comfortable one. To do so, the behavior of the obstacles is predicted by the algorithm, checking if their position will respect the safety distance requirements imposed during the complete lane change. After 0.6 seconds a suitable trajectory is found and passed through as new reference. The host vehicle starts the lane switch to the middle of the three lanes. At $t = 11,6$ seconds the lane change is completed and the trajectory finder is enabled again, but since the TTC value for the vehicle in the middle lane at this moment is 7.84 seconds, a new trajectory is searched for the lane switch to the left lane. But in this case, no trajectory is found for the current velocity (one that can respect the safety distance during the overall lane change). The reference speed of the host is decreased by $5km/h$ and the process is repeated until a trajectory was found for a speed of $120km/h$ that respects the safety limits. This trajectory is again inserted as the new reference trajectory.

After both of the vehicles are overtaken by the host, the road ahead of the host vehicle is empty. Therefore (as the speed is increased again to the maximum value of $130km/h$) two trajectories are found for both the lane switches to the middle and the most right lane at $t = 27$ seconds and $t = 39.4$ seconds respectively and same as before, inserted as reference trajectories one after the other until the lane changes are completed. The completed trajectory for this double overtaking maneuver is shown in figure 5-2a and 5-2b. The results show that the algorithm is perfectly capable of performing such a maneuver.

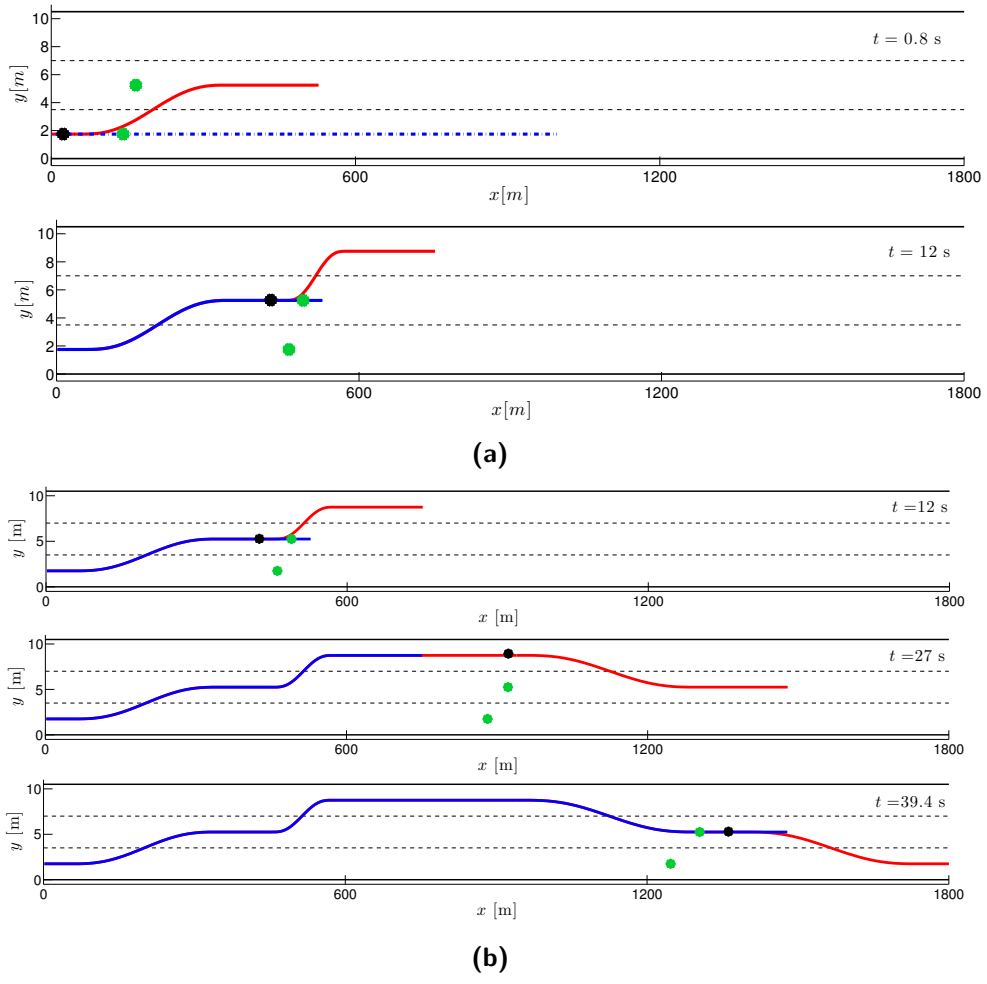

Figure 5-2: Reference trajectories as used during the double overtaking maneuver. (a) Trajectories used to overtake the two vehicles (b) Trajectories to move back to the right lane.

5-2-2 Scenario 5: Sudden braking of the vehicle in front

This is one of the scenarios designed to simulate a critical condition that can occur on a highway. At the beginning of the simulation, two vehicle are positioned inside the scenario:

the host vehicle and an obstacle vehicle. Both the vehicles are on the right lane and the host vehicle is at a distance of 100 meters behind the obstacle. The host vehicle is traveling at $v_{host} = 36m/s$, while the other vehicle is moving at a slightly lower velocity, $v_{obstacle} = 35m/s$. After 10 seconds from the beginning of the simulation, the leading vehicle suddenly brakes. The entity of the braking maneuver can be considered as critical; decelerating the vehicle from $35m/s$ to a velocity of $15m/s$ in just 3 seconds. The aim of this simulation is to test the reaction of the host vehicle to a really critical braking maneuver, to verify its capability to reduce the velocity while searching for a lane change maneuver. Moreover, the application of the safety area will be tested, being the scenario properly configured to cause the host vehicle (prediction horizon) to enter the safety area.

In the beginning the distance between the host and the obstacle is large enough, so no overtake is needed, but due to the braking of the obstacle vehicle, at $t = 10.8$ seconds the TTC limit is reached, so the trajectory finder starts to look for a suitable overtaking trajectory (with the speed of $36km/h$ finds it and applies it as a new reference trajectory. However, due to the excessive braking of the obstacle vehicle, at $t = 12.6$ seconds, the fifth point of the MPC prediction horizon enters the safety area behind the obstacle vehicle. The speed logic controller therefore changes the reference speed to the current $v_{obstacle}$ and the MPC controller needs to hit the brakes to reduce the error between the current and reference speed. Figure 5-3 shows this behavior, where the reference speed drops multiple times as a result of the obstacle vehicle lowering speed.

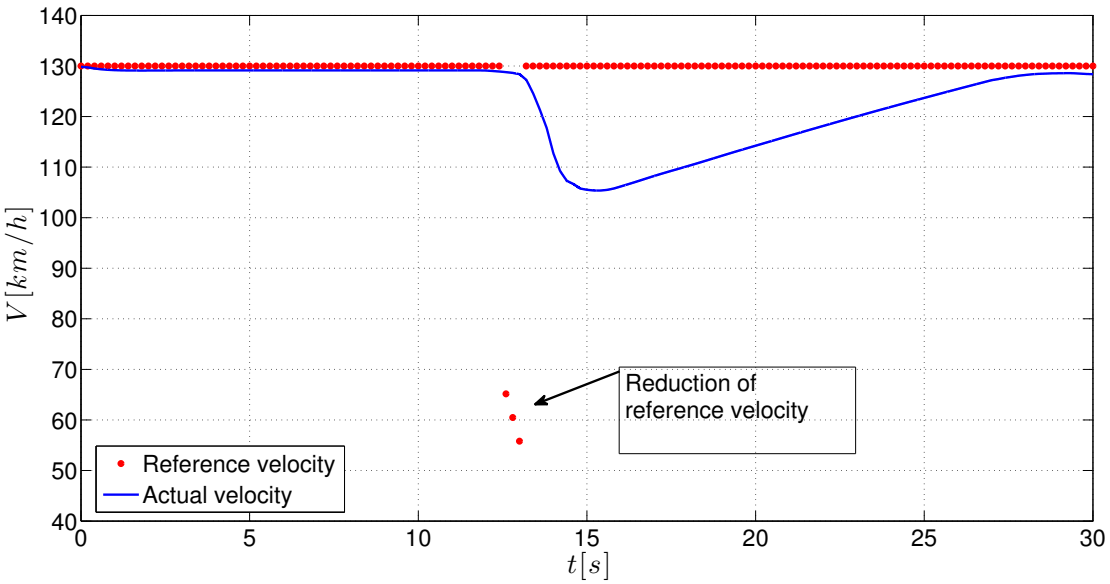

Figure 5-3: Reference vs actual velocity. Reduction of the reference when the fifth point of MPC goes in the safety area

While the host continues to follow the reference trajectory, at $t = 13.6$ seconds the fifth point of the prediction horizon leaves the safety area behind the obstacle vehicle, enabling the speed

logic block to restore the initial overtaking speed as the reference speed for the cost function. The car smoothly accelerates and the overtaking maneuver is completed successfully. Figure 5-4 shows the position of the simplified model used by the MPC (blue) and the output of the complex model used in the simulation for a sampling rate of 5Hz and 20Hz. Overshoot of the vehicle and following oscillations in lateral direction when the vehicle tries to go back to the center of the lane are shown to be big problems here. These common problems for applications of MPC in autonomous vehicles find their cause in weight tuning of the cost function and the prediction error of the MPC model according to the "actual car" (read: complex 9-DOF model used for simulation).

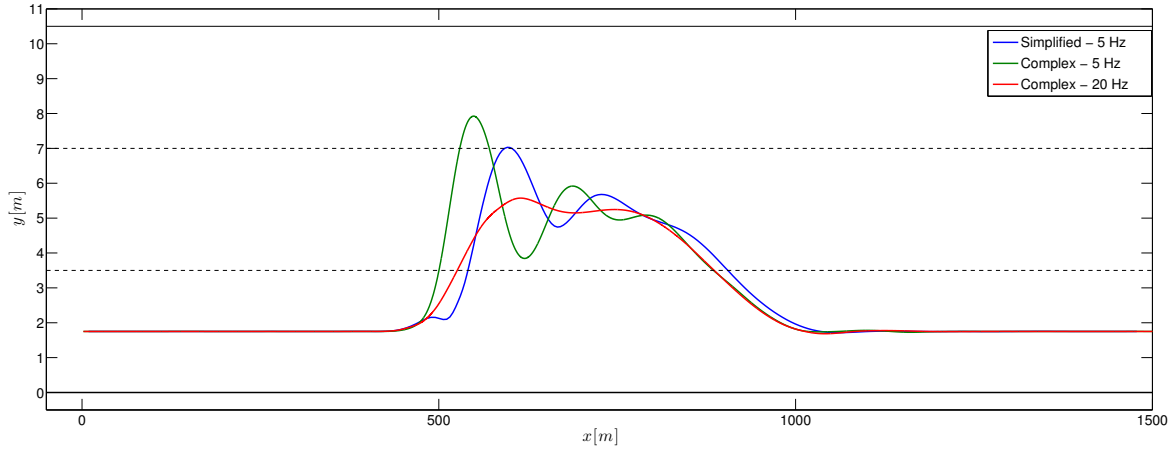

Figure 5-4: Lateral position of the three models during the overtaking maneuver

5-3 Discussion

Considering the cases of the autonomous vehicle moving in normal situation, with other vehicles on the road traveling at higher or lower velocities and performing ordinary maneuvers, such as lane changes, smooth braking and acceleration, the proposed MPC algorithm performs very well and the general behavior is smooth and comfortable. The system shows the capability to maintain the desired higher possible velocity whenever this is possible. If another vehicle is present, traveling at a slightly lower velocity, the autonomous system is able to detect it sufficiently in advance to plan a completely safe and most comfortable lane change trajectory possible, in order to avoid the obstacle. Even in scenarios simulating common highway traffic situations marked as critical in autonomous vehicle testing, the behavior obtained fulfills the desired requirements. Remarks can be made about the conservativeness of the chosen parameters, e.g. the TTC of 15 seconds feels a bit large, especially for situations where the difference in velocities are larger, as human driver might not switch to the overtaking lane that far in advance. This is of course strongly related to ones driving style. Also the safety area of one second, resulting in a traveling distance of two seconds, although suitable for human drivers considering reaction times, for autonomous vehicles this might be

unnecessary as the algorithm is capable of reacting much faster than that.

More troubling is the overshoot as shown in the results of Scenario 5 (see section 5-2-2). Increasing the weight of the term on the error between reference and current position could improve the reference tracking ability, but also causes the other terms to be less important. Therefore careful tuning could improve the behavior, but a trade-off will always have to be made. An increase in sampling rate for the MPC controller could also improve the behavior a lot, allowing the MPC controller to better predict the behavior of the system, but this would cause a significant increase in computational demand, making it impossible for the algorithm to run in real time. Nevertheless, the overall conclusion can be made that the MPC controller designed looks very promising as a candidate to be implemented in the DAVI vehicle for road testing.

However, even if the algorithm works very well for all of the nine scenarios simulated and tested, it can not be guaranteed that the decision making logics from both the trajectory planning and the speed logic block, will work in every possible situation, since no formal verification is done. Could we find possible safety issues, caused by the failure of the controllers logic, due to unexpected scenarios? Because of the fact that after a decision has been made to switch lanes, this can not be made undone, a closer look was taken at the behavior of the system in case of significant environment changes, after the decision was made to switch lanes. For example, after overtaking a slower vehicle, the host is in the left lane, trying to get back to the most right lane. The condition for this trajectory to be accepted is that the TTC with respect to all of the vehicles in front is larger than 15 seconds. Assume that the road in front is completely empty, hence the trajectory finder sets the most comfortable trajectory with respect to the current host speed, as the new reference trajectory. At that point the trajectory finder is disabled and rules of the speed logic block dictate that the speed is kept constant during the lane switch. But what will happen if the slower vehicle (just overtaken by our host) suddenly decides to accelerate? The safety area, introduced to keep a safe distance to other vehicles, is only considered at the back of other vehicles, therefore nothing prevents the host from starting to move to the right lane, even while the other vehicle is accelerating. The question is if the risk collision term of the MPC cost function is large enough to keep avoiding a collision, i.e. keep the host vehicle in the left lane, even if the error between the current position and the reference trajectory increases over time.

Another question mark can be placed at the ability of the algorithm to handle other vehicles switching lanes while it is deciding whether or not it is safe to overtake. When this would happen in front of the host vehicle, the safety area function will induce the necessary speed adaption as was shown in "scenario 6" of the original work. Possible safety risk could occur however, if for example on a three lane highway, faster traffic might be switching into our overtaking (or middle) lane after overtaking a vehicle in the middle lane.

Besides these possible safety risks introduced by unexpected scenarios, another concern could be placed at the fact that too large accelerations or braking during steering actions may result in the car becoming unstable. In normal situations, the cost function term that penalizes the change in acceleration/braking might be sufficient to prevent unstable situations, but when a collision needs to be avoided, for whatever reason, the collision risk term will gain the upper hand. Moreover, when the road is curved, changes in the steering angle may also have very

strong effects on the behavior of the car. Although the MPC framework is very robust and stabilizing in nature, the 3-DOF model may not be enough to prevent the calculation of system inputs that cause unstable situations. The fact that acceleration and braking both lack the presence of a hard constraint does not help either, since there is nothing to prevent full acceleration or braking. As an addition to the previous mentioned scenario, an accelerating other vehicle might cause a situation where a collision with the host from behind will occur if the host does not accelerate.

In summary, as mentioned before, while most of the highway scenarios should not be a problem for the algorithm to deal with, it can never be guaranteed that this is the case for every possible scenario. Besides that, overshoot followed by lateral oscillations (between the lane boundaries) does occur in multiple scenarios and increasing the sample rate is too computationally demanding to be used as a solution. The authors even had to conclude their work with a statement that the algorithm would never be able to run in real time, before it is converted to a more suitable programming language, for example C++. Finally, sensor information and correct object tracking are vital assets for the algorithm to work in every scenario. But since correct object detection and tracking and pre-filtered sensor information are assumptions made during the project description of both this work and the work of Manazza and Gottardis, this remark will be neglected.

Chapter 6

Conclusion

This chapter will summarize the former chapters by evaluating the problem statement from chapter 1 and defining some conclusions about the outcomes of this project. Progress and problems during the research period corresponding to this work, will also be briefly discussed. Section 6-2 will finalize this report by stating some recommendations and suggestions for possible future research.

6-1 Conclusions & Experiences

Formally, if we review the problem statement presented in section 1-3 we can conclude that the outcome of this work is a success. The required capabilities of a controller that enables an autonomous vehicle to navigate through highway traffic are described and the corresponding formal specifications are proposed in chapter 4. Full LTL specifications are presented for a simplified model that only has basic overtaking functionality and additional specifications to add more functionality are proposed as well. These specifications are implemented in the existing receding horizon framework as described in section 2-3 to keep the computational demands of the synthesis of our controller as low as possible. This framework was already successfully proven to be able to navigate the autonomous vehicle on a road with static obstacles and was extended by using our new specifications to handle moving obstacles (i.e. other vehicles) as well.

This being said, it is clear that the fact that we were not able to actually synthesize a controller, and test its corresponding performance, is very disappointing. Difficulties from the installation and usage of our selected tool, TuLiP, could not be solved in time, even with the help of the original developers of the software suite. However, the fact that only adjustments were made to an already existing and proven successfully framework, we are confident that the LTL specifications as presented in chapter 4 should deliver a very well capable control algorithm. The checks that were done on the satisfiability of the combination of formulas, using the online model checking tool, further strengthen this expectation.

As for the evaluation of the conventional MPC algorithm; it is clear that the results shown by the experiments are very promising. The vehicle is capable of handling a lot of different situations, some even considered critical and none expected by the assumptions. Nevertheless, it can never be guaranteed that this is the case for every possible scenario, due to the lack of formal proof of the safety specification. Moreover, despite the efforts the designers took to keep the computational costs as low as possible, the algorithm would not be able to run in real time, even for the low sampling rate, due to the various MATLAB functions which need to be compiled in every time step. Also, overshoot is still a big issue, which could only be slightly improved by very carefully tuning the weights of the used cost function. An increase of the sampling rate would reduce this problem, but would require significantly more processing power.

In the end, both approaches look very promising, with the conventional MPC having the advantage that it is already running in simulation, but both also still need a lot of work, before they could be implemented in a real autonomous vehicle, such as the DAVI car.

6-2 Recommendations & Future Work

While the correct-by-design methodology and receding horizon framework create large expectations, it is clear that the software issues were slowing down the progress of designing a LTL based controller significantly. Since most of the problems occurred by version mismatching from the large list of dependencies TuLiP has, it could be really helpful if a list of all the dependencies, including their compatible version numbers, would be kept up-to-date. The most ideal solution of having one command (or master script) that could be executed to install every single package needed, seems impossible to achieve due to the open-source nature of the Linux platform and the ever changing software environments. Other improvements could be found in adding a graphical user interface or trying to port the complete TuLiP functionality to a more user friendly operating system like Microsoft Windows. Downside of the latter would be the loss of the open platform, however.

Due to limited time and maybe keeping our hopes up for too long that TuLiP would eventually run properly on our system, we were not able to validate and test the proposed LTL statements, so this would have to be done in future research. Although model checking tools are available in large numbers, software packages that can synthesize a control strategy are very rare. The fact that, as far as we know, there are no other tools which might be able to synthesize a controller from the found specifications, this does not mean none exist. Research could be done if such tools exist and if the proposed receding horizon framework could also be implemented within those tools as well.

Besides the actual implementation and testing of the proposed specifications, more research should be done on the values of the various comfort and safety parameters. Many research papers on the comfort criteria of human beings for trajectories exist, but the number of different outcomes and assumptions is almost just as large.

With respect to the discussed MPC algorithm, future work could involve the transforma-

tion of the MATLAB/Simulink code into a "proper" real time program language (e.g. C++) and trying to further reduce the computational demands. Next step would be the implementation and testing of the algorithm in the DAVI vehicle. Another topic of research could be to formally proof the correctness of the controller with respect to safety specification (formulated in LTL or another language). This latter would improve the trust value of the algorithm with respect to it being able to handle the possible safety issues as discussed.

Bibliography

[1] VA, "Davi official website," *http://davi.connekt.nl/*.

[2] R. Hoogendoorn, B. van Arem, R. Happee, M. M. Espinoza, and D. Kotiadis, "Towards safe and efficient driving through vehicle automation: The dutch automated vehicle initiative," *http://davi.connekt.nl/*, 2013.

[3] P. Tabuada, *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, 2009.

[4] C. Baier and J. P. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[5] A. Valmari, "The state explosion problem," in *Lectures on Petri Nets I: Basic Models* (W. Reisig and G. Rozenberg, eds.), pp. 429–528, Springer Berlin Heidelberg, 1998.

[6] I. M. Mitchell, "Comparing forward and backward reachability as tools for safety analysis," in *Hybrid Systems: Computation and Control*, pp. 428–443, Springer Berlin Heidelberg, 2007.

[7] T. Wongpiromsarn, "Formal methods for design and verification of embedded control systems: Application to an autonomous vehicle," Master's thesis, California Institute of Technology, Pasadena, California, 2010.

[8] Y. Gao, J. Lygeros, and M. Quincapoix, "The reachability problem for uncertain hybrid systems revisited: A viability theory perspective," in *Hybrid Systems: Computation and Control*, pp. 242–256, Springer Berlin Heidelberg, 2006.

[9] R. M. Bayen, I. M. Mitchell, M. M. K. Oishi, and C. J. Tomlin, "Aircraft autolander safety analysis through optimal control-based reach set computation," in *Journal of Guidance, Control and Dynamics, Vol. 30, No. 1*, 2007.

[10] N. Lynch, R. Segala, and F. Vaandrager, "Hybrid i/o automata," *Inf. Comput.*, pp. 105–157, 2003.

[11] E. Dolginova and N. Lynch, "Safety verification for automated platoon maneuvers: A case study," in *Proceedings International Workshop on Hybrid and Real-Time Systems*, pp. 154–170, Springer-Verlag, 1997.

[12] C. Livadas, J. Lygeros, and N. A. Lynch, "High-level modeling and analysis of the traffic alert and collision avoidance system (tcas)," *Proceedings of the IEEE*, pp. 926–948, 2000.

[13] N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive(1) designs," in *Verification, Model Checking and Abstract Interpretation, volume 3855 of Lecture Notes in Computer Science*, pp. 364–380, 2006.

[14] H. Kress-Gazit, G. Fainekos, and G. Pappas, "Where's waldo? sensor-based temporal logic motion planning," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 3116–3121, 2007.

[15] T. Wongpiromsarn, U. Topcu, and R. Murray, "Receding horizon temporal logic planning for dynamical systems," in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pp. 5997–6004, 2009.

[16] M. van Infrastructuur en Milieu, ed., *Verkeersborden en Verkeersregels in Nederland*. abstract from 'Reglement Verkeersregels en Verkeerstekens 1990', 2014.

[17] YPCA, AllSecure, and TrafficRadio, "Het nationale automobilisten onderzoek 2014," *http://www.ypca.nl/artikel/295/resultaten-het-nationale-automobilisten-onderzoek-2014-bekend*.

[18] K. Bussemaker, R. Happee, and D. Kotiadis, "Automated driving using existing adas systems," 2014.

[19] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *Automatic Control, IEEE Transactions on*, pp. 287–297, 2008.

[20] D. A. Reece and S. Shafer, "A computational model of driving for autonomous vehicles," tech. rep., Transportation Research, 1991.

[21] T. Eigel, ed., *Integrierte Längs- und Querführung von Personenkraftwagen mittels Sliding-Mode-Regelung*. AutoUni - Schriftenreihe, 2010.

[22] T. Flash and N. Hogan, "The coordination of arm movements: An experimentally confirmed mathematical model," *Journal of Neuroscience 5*, 1985.

[23] A. D. M. Mazo Jr. and P. Tabuada, "Pessoa: towards the automatic synthesis of correct-by-design control software," tech. rep., Work-in-progress HSCC, 2010.

[24] M. Kloetzer and C. Belta, "A fully automated framework for controller synthesis from linear temporal logic specifications," in *Technical Report CISE 2005-IR-0050, Boston University*, 2005.

[25] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. Murray, "Tulip: A software toolbox for receding horizon temporal logic planning," in *Hybrid System: Computation and Control*, pp. 313–314, 2011.

[26] A. Duret-Lutz and D. Poitrenaud, "Spot: an extensible model checking library using transition-based generalized büchi automata.," in *Proceedings of the 12th IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'04)*, pp. 76–83, 2004.

[27] J.-M. Couvreur, "On-the-fly verification of linear temporal logic.," in *Proceedings of the World Congress on Formal Methods in the Development of Computing Systems (FM'99)*, pp. 253–271, 1999.

[28] P. Gastin and D. Oddoux, "Fast ltl to büchi automata translation," in *Proceedings of the 13th International Conference on Computer Aided Verification (CAV'01)*, pp. 53–65, 2001.

[29] LRDE and LIP6, "Spot website - wiki," *http://spot.lip6.fr/wiki/SpotWiki*.

[30] T. Simonite, "Data shows googleŠs robot cars are smoother, safer drivers than you or i," *MIT Technical Review*, 2013.

[31] J. Carfrae, "An automated adventure at the wheel of a driverless bmw," *The National, United Arab Emirates*, 2011.

[32] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. Keller, E. Kaus, R. Herrtwich, C. Rabe, D. Pfeiffer, F. Lindner, F. Stein, F. Erbs, M. Enzweiler, C. Knoĺppel, J. Hipp, M. Haueis, M. Trepte, C. Brenk, A. Tamke, M. Ghanaat, M. Braun, A. Joos, H. Fritz, H. Mock, M. Hein, and E. Zeeb, "Making bertha drive; an autonomous journey on a historic route," *Intelligent Transportation Systems Magazine, IEEE*, pp. 8–20, 2014.

[33] Volvo, "Volvo car groups: First self-driving autopilot cars test on public roads around göthenburg.," *https://www.media.volvocars.com/global/en-gb/media/pressreleases/145619/volvo-car-groupsfirst-self-driving-autopilot-cars-test-on-public-roadsaround-gothenburg.*, 2014.

[34] S. S. Manazza and P. Gottardis, "Automated controlled vehicle based on nonlinear model predictive control connected to a safety path planner with online collision risk estimation," Master's thesis, Politecnico di Milano, 2014.

[35] T. Howard, C. Green, and A. Kelly, "Receding horizon model-predictive control for mobile robot navigation of intricate paths," in *Proceedings of the 7th International Conferences on Field and Service Robotics*, 2009.

[36] A. v. d. Boom, "Lecture notes for the course sc4060 - model predictive control," in *Model Predictive Control*, Delft Center for Systems and Control, TU Delft, 2011.

[37] Y. Lee, B. Kouvaritakis, and M. Cannon, "Constrained receding horizon predictive control for nonlinear systems," in *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, pp. 3370–3375, 1999.

[38] TASS, "Prescan product website - tass international," *https://www.tassinternational.com/prescan-overview*.

Glossary

List of Acronyms

ADAS	Advanced driver assistance systems
CACC	Cooperative Adaptive Cruise Control
Caltech	California Institute of Technology, Pasadena
CISE	Center for Information and Systems Engineering, Boston University
CyPhyLab	Cyber-Physical Systems Laboratory
DAVI	Dutch Automated Vehicle Initiative
DOF	Degrees of Freedom
GR(1)	Generalized Reactivity(1)
HIOA	hybrid I/O automata
I/O	input/output
ITS	Intelligent Transport Systems
LIP6	Laboratoire d'Informatique de Paris 6
LRDE	EPITA Research and Development Laboratory
LTL	Linear Temporal Logic
MPC	Model Predictive Control
PHAVer	Polyhedral Hybrid Automaton Verifier
PVS	Prototype Verification System
TCAS	Traffic Alert and Collision Avoidance System
TTC	time-to-collision

TU Delft Delft University of Technology

UCLA University of California, Los Angeles

V2I Vehicle-to-Infrastructure

V2V Vehicle-to-Vehicle