To Err is Human: Designing Correct-by-Construction Driver Assistance Systems using Cognitive Modelling

Francisco Girbal Eiras

Linacre College, University of Oxford

Supervised by:

Dr. Morteza Lahijanian and Prof. Marta Kwiatkowska

A thesis submitted for the degree of

Master of Science in Computer Science

Trinity 2018

Acknowledgements

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

Contents

List of Figures

Chapter 1

Introduction

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

Chapter 2

Literature Review and Definitions

This chapter presents the concepts and theory required to understand the methodology, rationale and relevance of this dissertation. It starts by defining a Cognitive Architecture, and goes into more detail about Adaptive Control of Thought - Rational, which is the underlying framework used in the Integrated Driver Modelling described in Section 2.2 and the basis for the system developed in Chapter 3: *Human Driver Modelling*. The chapter concludes with the introduction of several important concepts in the area of Probabilistic Model Checking, which are essential for modelling, verification and synthesis in this context.

2.1 Cognitive Architectures

Different definitions of a cognitive architecture can be found in the literature [11]. According to Sun, a cognitive architecture is a *"broadly-scoped, domain-generic computational cognitive model, capturing the essential structure and process of the mind, to be used for a broad, multiple-level, multiple-domain analysis of behavior"* [23]. In that sense, a cognitive architecture is no more than a formalised framework of the perception, memorisation, decision making, reasoning and execution of the human mind. It is important to note that this architecture should capture both the abilities (e.g. memory storage and recall, perception or motor action) and constraints (e.g. memory decay and limited motor performance) of the human system [21].

The idea of a cognitive architecture originated in the early 1970s. A survey by Korseruba and Tsotsos [11] revealed that, since then, an estimated 195 different cognitive

architectures with different assumptions, structural organisations and implementations have been developed. Applications of these range from the fields of Robotics to Natural Language Processing [11]. One of the most well known of those architectures is Adaptive Control of Thought - Rational.

2.1.1 Adaptive Control of Thought - Rational (ACT-R)

Adaptive Control of Thought - Rational (or ACT-R as it is commonly known) is a particular cognitive architecture first presented by John R. Anderson in 1983 [2] and further developed by Anderson *et al.* in [3].
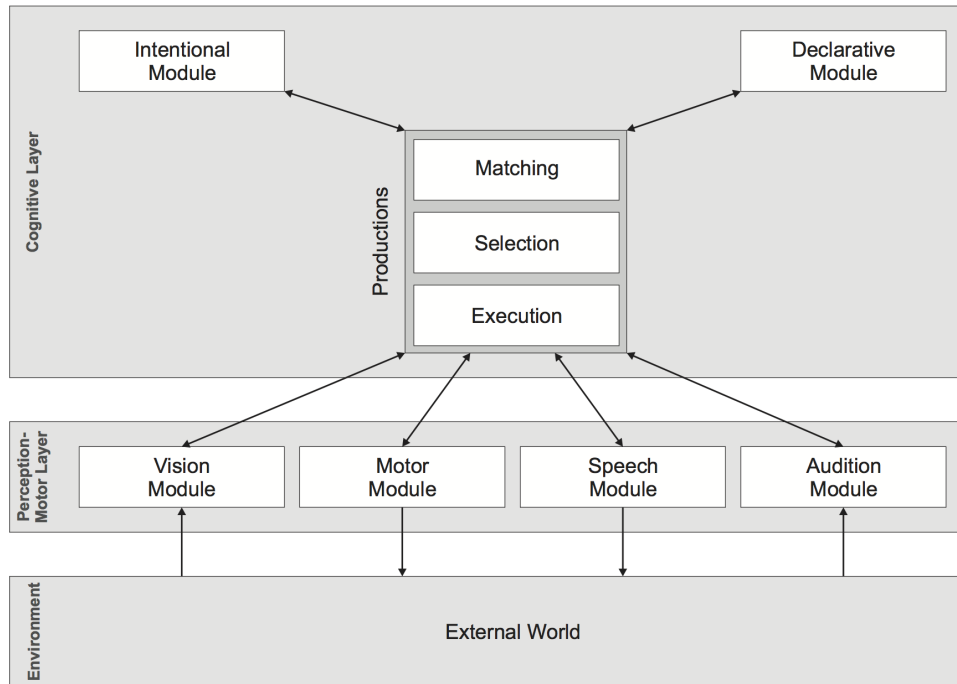
Figure 2.1: Overview of the ACT-R high level architecture (from [5]).

The architecture can be generally described as two distinct modules: a *perceptual-motor* layer and a *cognitive* layer, as presented in Figure 2.1. The perceptual-motor module corresponds to the interface of the cognition with the environment (which plays a key role in ACT-R), being comprised of sub-modules such as vision and motor actions. The cognitive layer is focused on memory, which can be divided into two different categories: *declarative* (consisting of factual knowledge and goals - e.g. *"The maximum driving speed in a typical UK motorway is 70 mph"* or *"Try get to point B"*) and *procedural* (consisting of rules/procedures - e.g. *"If the lead vehicle is going*

slowly, attempt an overtake") [5]. Furthermore, declarative memory is composed of *chunks*, which are smaller pieces of information divided into categories or attributes that form more complex memories. In the example *"The maximum driving speed in a typical UK motorway is 70 mph"*, `maximum driving speed in a typical UK motorway` is a category, whereas `70 mph` is an attribute. Another possible attribute for this category could be `112 km/h` (since $70mph = 112km/h$).

2.2 Integrated Driver Modelling in ACT-R

The first integrated approaches to this problem, proposed in [17, 6, 22], rely on modelling human behaviour through continuous controllers and specifically engineered frameworks. However, these methods lack the human behaviour variability attributed to the discrete nature of the control actions performed by the drivers. With this in mind, in [20] Salvucci *et al.* proposed a proof-of-concept of the introduction of human constrains in the driver model through the use of the cognitive architecture ACT-R. In [21], Salvucci proposed an updated version of the human driver model initially introduced in [20], which was improved using the advances in ACT-R (the model uses version 5.0 of the architecture) and re-designed elements (e.g. lane changing behaviour or distraction) resulting from validation performed by empirical studies.

The model Salvucci introduced in [21] was developed with a specific scenario in mind: *"(...) driving a standard midsize vehicle on a multilane highway with moderate traffic"* [21]. It consists of three distinct modules interacting in a sequential way: **control**, **monitoring**, and **decision making**. A high level schematic of the model can be found in Figure 2.2.
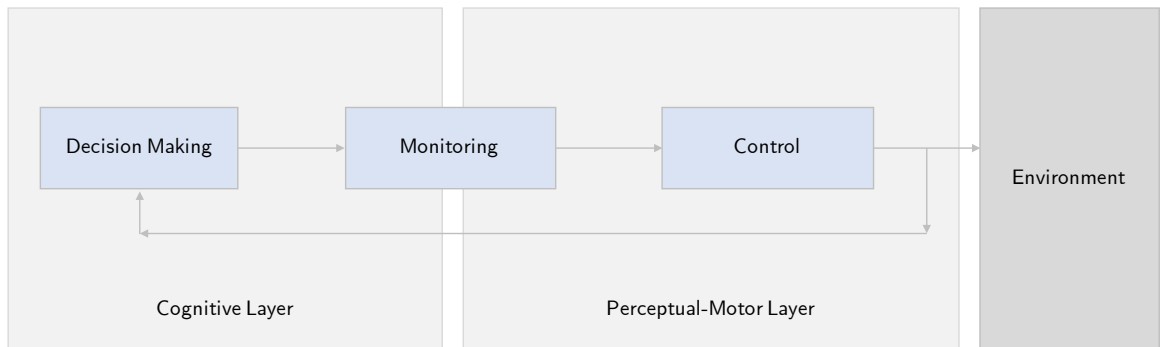
Figure 2.2: High level representation of the architecture presented in [21].

The **control** component manages both the lower level perception cues and the physical manipulation of the vehicle (e.g. steering, accelerating or breaking). The module can be divided into lateral (i.e. steering) and longitudinal (i.e. speed) control, each modelled by a separate control law.

The lateral control is determined by the existence of two artefacts that the driver obtains using lower level perception cues: the *near point* and the *far point*. In this model, *"the near point represents the vehicle's current lane position, used to judge how close the vehicle is to the center of the roadway (...) [and] is characterized as a point in the center of the near lane visible in front of the vehicle, set at a distance of 10 m from the vehicle's center"* while *"the far point indicates the curvature of the upcoming roadway, used to judge what the driver should execute to anticipate the upcoming curvature and remain in the current lane* [21]. At any cycle $i > 0$, the model works by using perception to determine the visual angles $\theta_{near}^{(i)}$ and $\theta_{far}^{(i)}$ and calculates:

$$
\begin{aligned}
\Delta\theta_{near} &= \theta_{near}^{(i)} - \theta_{near}^{(i-1)} \\
\Delta\theta_{far} &= \theta_{far}^{(i)} - \theta_{far}^{(i-1)}
\end{aligned}
\tag{2.1}
$$

The control law for the steering angle φ can be defined as:

$$
\Delta\varphi = k_{far}\Delta\theta_{far} + k_{near}\Delta\theta_{near} + k_I \min\left(\theta_{near}^{(i)}, \theta_{\max}\right)\Delta t
\tag{2.2}
$$

with k_{far}, k_{near} and k_I defined as in [21].

The process for the longitudinal control is very similar. At any cycle $i > 0$, the model starts by encoding the position of the lead vehicle and calculating the time headway $thw_{car}^{(i)}$ to it. It then computes:

$$
\Delta thw_{car} = thw_{car}^{(i)} - thw_{car}^{(i-1)}
\tag{2.3}
$$

The control law for the linear acceleration ψ can be written as:

$$
\Delta\psi = k_{car}\Delta thw_{car} + k_{follow}(thw_{car} - thw_{follow})\Delta t
\tag{2.4}
$$

with k_{car}, k_{follow} and thw_{car} defined as in [21].

These control laws can be applied in order to model the behaviour of the driver in terms of lane keeping and curve negotiation, and generalise to lane changing actions in a straightforward manner. In order to initiate a lane change, the driver just starts following the near and far points of the destination lane instead of the current lane, as Salvucci and Liu presented in [22].

The **monitoring** component maintains situational awareness through the awareness of the position of other vehicles around the driver's vehicle. It accomplishes this through a random-sampling, with probability $p_{monitor}$, of one of four areas: either the left or right lane, and either forward or backward (with equal probability). After deciding which area to sample (if any), the model uses visual perception to determine whether a vehicle is present or not. If it is, the distance, lane and direction are saved in ACT-R's declarative memory. The value of $p_{monitor}$ is defined in [21].

Finally, the **decision making** module uses the information gathered in the monitoring and control stage (if any is available) to determine which tactical decision should be taken. In this model, the decision corresponds to determining when and where a lane change should occur. Salvucci describes this process in [21] as: *"If the driver's vehicle is in the right lane, the model checks the current time headway to the lead vehicle (if any); if the lead car time headway drops below a desired time headway thw_{pass}, the model decides to change lanes to pass the vehicle. If the driver vehicle is in the left lane, the model checks instead simply for the presence of a lead vehicle. If there is a lead vehicle, the model remains in the left lane (because this vehicle is also passing other vehicles); otherwise it decides to [try to] change lanes to return to the right lane"*. This process is summarised in the flowchart presented in Figure 2.3, and the value of thw_{pass} is defined in [21].

Even after this decision to change lanes is taken, the model must then determine whether it is actually safe to do so. In order to verify that it is, the model initially attempts to recall, from the declarative memory, the nearest vehicle to it in the destination lane. If one is recalled at a distance closer than d_{safe}, then the change is aborted. If not, then the vehicle performs a safety monitoring of the destination lane. If this monitoring does not observe a vehicle at a distance closer than d_{safe}, then the vehicle initiates the execution of the lane change. The parameter d_{safe} is as defined in [21].
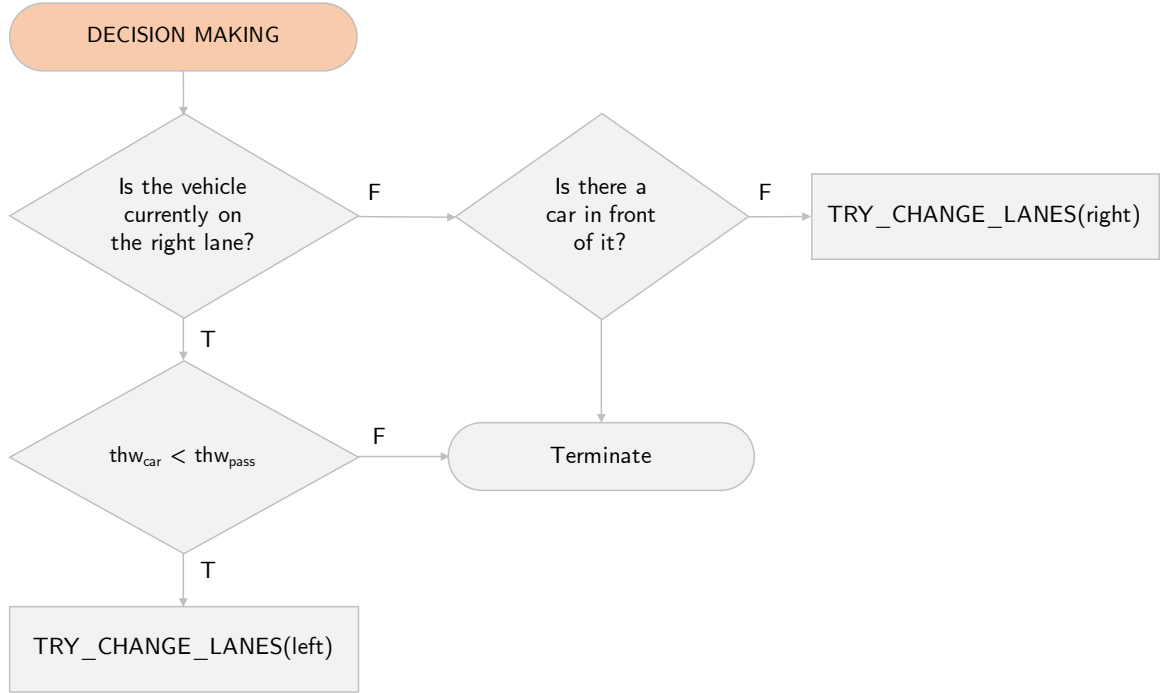
Figure 2.3: Flowchart of part of the decision making process of the model.

2.3   Probabilistic Model Checking

Model checking is a formal verification technique consisting of the process of automatically verifying, through exhaustive search, whether a model meets a given specification [4]. A model in this context is considered to be a labelled finite state-transition system, with states corresponding to configurations of the system and transitions between states representing the evolution between the said configurations.

Probabilistic model checking is a generalisation of model checking for the automated verification of systems that exhibit probabilistic behaviour [10, 4]. It should be noted that this is not to be confused with *probabilistic verification*, which amounts to partial state-space exploration [4]. Within this section, important concepts of probabilistic model checking related to modelling of systems, and verification and synthesis of models will be explained in detail.

2.3.1 Markov Chains and Decision Processes

Probabilistic model checking begins with the appropriate modelling of the system in a stochastic finite state-transition representation. There are several ways to do this according to the type of system to be modelled.

In *discrete-time Markov Chains* (DTMCs), all choices (i.e. transitions) are probabilistic [4]. While this is useful in some cases, others require nondeterminism, which is where *Markov Decision Processes* (MDPs) come in. In MDPs, nondeterministic probabilistic choices coexist. Essentially, a DTMC is an MDP with a uniquely determined probabilistic distribution [4]. Both DTMCs and MDPs can be augmented with reward structures, which correspond to real positive values that can be interpreted as bonuses or costs [4].

The formal definitions of DTMCs, MDPs and reward structures follow (adapted from [4, 10, 13]).

Definition 1. Discrete-time Markov Chain (DTMC)

A *discrete-time Markov chain* is a tuple $\mathcal{M} = (S, \mathbf{P}, p_{init}, AP, L)$ such that:

- S is a countable, non-empty set of states

- $\mathbf{P} : S \times S \to [0, 1]$ is a transition probability function such that for all states $s \in S$:

$$\sum_{s' \in S} \mathbf{P}(s, s') = 1 \tag{2.5}$$

- $p_{init} : S \to [0, 1]$ is the initial state distribution, such that:

$$\sum_{s \in S} p_{init}(s) = 1 \tag{2.6}$$

- AP is a set of atomic propositions and $L : S \to 2^{AP}$ is a labelling function.

$\mathcal{M}$ is denoted *finite* if S and AP are finite sets. ∎

Definition 2. Markov Decision Process (MDP)

A *Markov decision process* is a tuple $\mathcal{M} = (S, Act, \mathbf{P}, p_{init}, AP, L)$ such that:

- S is a countable, non-empty set of states

- Act is a set of actions

- $\mathbf{P} : S \times Act \times S \to [0,1]$ is a transition probability function such that for all states $s \in S$ and actions $\alpha \in Act$:

$$\sum_{s' \in S} \mathbf{P}(s, \alpha, s') \in \{0, 1\} \tag{2.7}$$

- $p_{init} : S \to [0,1]$ is the initial state distribution, such that:

$$\sum_{s \in S} p_{init}(s) = 1 \tag{2.8}$$

- AP is a set of atomic propositions and $L : S \to 2^{AP}$ is a labelling function.

An action $\alpha \in Act$ is enabled in a state s if, and only if, $\sum_{s' \in S} \mathbf{P}(s, \alpha, s') = 1$. Let $Act(s)$ denote the set of actions enabled in a state s. It must be that for all states $s \in S$, $Act(s) \neq \emptyset$. ∎

Definition 3. Reward Structures for DTMCs and MDPs

A *reward structure* for a discrete-time Markov chain $\mathcal{M} = (S, \mathbf{P}, p_{init}, AP, L)$ is a tuple $\mathcal{C} = (\rho, \iota)$ such that:

- $\rho : S \to \mathbb{R}_{\geq 0}$ is a state reward function

- $\iota : S \times S \to \mathbb{R}_{\geq 0}$ is a transition reward function

A *reward structure* for a Markov decision process $\mathcal{M} = (S, Act, \mathbf{P}, p_{init}, AP, L)$ is a tuple $\mathcal{C} = (\rho, \iota)$ such that:

- $\rho : S \to \mathbb{R}_{\geq 0}$ is a state reward function

- $\iota : S \times Act \to \mathbb{R}_{\geq 0}$ is an action reward function

∎

There are some additional concepts related to DTMCs and MDPs which are relevant to the understanding of temporal logics and model checking. These are the concepts of paths in Markov chains or decision processes, and adversary in a Markov decision process (adapted from [4, 10]).

Definition 4. Paths in DTMCs and MDPs

An *(infinite) path* π in a discrete-time Markov chain $\mathcal{M} = (S, \mathbf{P}, p_{init}, AP, L)$ is defined as an infinite sequence of states $\pi = s_0 \to s_1 \to s_2...$ (also written as $\pi = s_0 s_1 s_2...$) such that $\forall i \geq 0 : \mathbf{P}(s_i, s_{i+1}) > 0$.

A *finite path* ρ in the discrete-time Markov chain $\mathcal{M}$ is defined as the prefix of an infinite path π, $\rho = s_0 \to s_1 \to ... \to s_n$ for $n > 0$.

An *(infinite) path* π in a Markov decision process $\mathcal{M} = (S, Act, \mathbf{P}, p_{init}, AP, L)$ is defined as an infinite sequence of states and actions $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2...$ (also written as $\pi = s_0 a_0 s_1 a_1 s_2...$) such that $\forall i \geq 0 : \mathbf{P}(s_i, a_i, s_{i+1}) > 0$.

A *finite path* ρ in the Markov decision process $\mathcal{M}$ is defined as the prefix of an infinite path π, $\rho = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} ... \xrightarrow{a_{n-1}} s_n$ for $n > 0$.

In both Markov chains and decision processes, the set of all infinite paths is denoted by $Paths(\mathcal{M})$, while the set of finite paths is denoted by $Paths_{fin}(\mathcal{M})$ ∎

Definition 5. Adversary (alternatively Strategy or Scheduler)

An *adversary* for an MDP $\mathcal{M} = (S, Act, \mathbf{P}, p_{init}, AP, L)$ is a function $\sigma : S^+ \to Act$ such that $\sigma(s_0 s_1...s_n) \in Act(s_n)$ for all $s_0 s_1...s_n \in S^+$.

An adversary σ is called memoryless if, and only if, for any $\pi_1 = s_0 s_1...s_n$ and $\pi_2 = s_0' s_1'...s_n$ it is true that $\sigma(\pi_1) = \sigma(\pi_2) = \sigma(s_n)$. ∎

2.3.2 Temporal Logics

In order to verify properties in models, there needs to be a way of expressing them formally and unequivocally. While some reachability and reward-based properties come directly from the definition of DTMCs, MDPs and reward structures [10], more interesting and complex properties require the use of a temporal logic. A propositional temporal logic is essentially an extension of propositional logic by temporal operators [4]. In this context, one can consider time as being *linear* or *branching*. In linear time, at every moment in time there must only exist a single successor moment, whereas from the branching perspective, there may exist multiple ones, forming a tree-like structure of alternatives. Linear Temporal Logic (LTL) is a temporal logic which assumes the first, while Computation Tree Logic (CTL) is based on the branching perspective. Probabilistic Computational Tree Logic (PCTL) is an extension of CTL that takes into account a probabilistic operator.

Given the definition of a path in an MC and MDP, it is possible to define the syntax for LTL, CTL and PCTL (adapted from [4, 10]).

Definition 6. Syntax and Semantics of LTL

An *LTL formulae* φ over a set of atomic propositions AP can formed according to the following grammar:

$$\varphi ::= \texttt{true} \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \mathrm{X}\varphi \mid \varphi_1 \ \mathrm{U} \ \varphi_2 \tag{2.9}$$

where $a \in AP$, X is the next operator and U is the until operator as defined below for Markov chains and decision processes.

For a given path $\pi = s_0 s_1 s_2...$ in a discrete-time Markov chain $\mathcal{M} = (S, \mathbf{P}, p_{init}, AP, L)$ (generalising trivially to a Markov decision process):

- $\pi \models \texttt{true}$ always.

- $\pi \models a$ if, and only if, $a \in L(s_0)$.

- $\pi \models \varphi_1 \wedge \varphi_2$ if, and only if, $\pi \models \varphi_1 \wedge \pi \models \varphi_2$.

- $\pi \models \neg\varphi$ if, and only if, $\pi \not\models \varphi$.

- $\pi \models \mathrm{X}\varphi$ if, and only if, $s_1 s_2... \models \varphi$.

- $\pi \models \varphi_1 \mathrm{U} \varphi_2$ if, and only if:

$$\exists i \geq 0 \text{ s.t. } (s_i s_{i+1}... \models \varphi_2) \wedge (\forall j < i : s_j s_{j+1}... \models \varphi_1) \qquad (2.10)$$

The operator F (eventually) can also be defined as:

$$\mathrm{F}\varphi = \texttt{true} \ \mathrm{U} \ \varphi \qquad (2.11)$$

■

It can be noticed that, as a result of the linear time consideration in LTL, all the formulas are taken over paths. This is not the case in CTL which can have both path and state formulae, as shown below.

Definition 7. Syntax and Semantics of CTL

A *CTL state formulae* Φ over a set of atomic propositions AP can formed according to the following grammar:

$$\Phi ::= \texttt{true} \ | \ a \ | \ \Phi_1 \wedge \Phi_2 \ | \ \neg\Phi \ | \ \mathrm{E}\varphi \ | \ \mathrm{A}\varphi \qquad (2.12)$$

where $a \in AP$ and φ is a path formula.

For a given state $s \in S$ in a discrete-time Markov chain $\mathcal{M} = (S, \mathbf{P}, p_{init}, AP, L)$ (generalising trivially to a Markov decision process):

- $s \models \texttt{true}$ always.

- $s \models a$ if, and only if, $a \in L(s)$.

- $s \models \Phi_1 \wedge \Phi_2$ if, and only if, $s \models \Phi_1 \wedge s \models \Phi_2$.

- $s \models \neg\Phi$ if, and only if, $s \not\models \Phi$.

- $s \models \mathrm{E}\varphi$ if, and only if, $\exists \pi \in Paths(s) : \pi \models \varphi$.

- $s \models \mathrm{A}\varphi$ if, and only if, $\forall \pi \in Paths(s) : \pi \models \varphi$

A *CTL path formulae* φ over a set of atomic propositions AP can formed according to the following grammar:

$$\varphi ::= \mathrm{X}\Phi \mid \Phi_1 \ \mathrm{U} \ \Phi_2 \tag{2.13}$$

where $a \in AP$ and Φ is a state formula.

For a given path $\pi = s_0 s_1 s_2...$ in a Markov chain $\mathcal{M} = (S, \mathbf{P}, p_{init}, AP, L)$ (generalising trivially to a Markov decision process):

- $\pi \models \mathrm{X}\Phi$ if, and only if, $s_1 \models \Phi$.

- $\pi \models \Phi_1 \ \mathrm{U} \ \Phi_2$ if, and only if:

$$\exists i \geq 0 \ \text{s.t.} \ (s_i \models \Phi_2) \wedge (\forall j < i : s_j \models \Phi_1) \tag{2.14}$$

It is also possible to define the operators F (eventually) and G (globally) as:

$$\begin{aligned} \mathrm{F}\Phi &= \texttt{true} \ \mathrm{U} \ \Phi \\ \mathrm{G}\Phi &= \neg \mathrm{E}(\neg\Phi) \end{aligned} \tag{2.15}$$

∎

Definition 8. Syntax and Semantics of PCTL

A *PCTL state formulae* Φ over a set of atomic propositions AP can formed according to the following grammar:

$$\Phi ::= \texttt{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \mathbb{P}_{\rhd p}(\varphi) \tag{2.16}$$

where $a \in AP$, $p \in [0,1]$, $\rhd \in \{>, <, \geq, \leq\}$ is a probability bound and φ is a path formula.

For a given state $s \in S$ in a discrete-time Markov chain $\mathcal{M} = (S, \mathbf{P}, p_{init}, AP, L)$ (generalising trivially to a Markov decision process):

- $s \models \texttt{true}$ always.

- $s \models a$ if, and only if, $a \in L(s)$.

- $s \models \Phi_1 \wedge \Phi_2$ if, and only if, $s \models \Phi_1 \wedge s \models \Phi_2$.

- $s \models \neg\Phi$ if, and only if, $s \not\models \Phi$.

- $s \models \mathbb{P}_{\triangleright p}(\varphi)$ if, and only if, $P_s(\pi \in Paths(s) : \pi \models \varphi) \triangleright p$ (that is, the probability, from state s, that φ is true for an outgoing path satisfies $\triangleright p$).

A *PCTL path formulae* φ over a set of atomic propositions AP can formed according to the following grammar:

$$\varphi ::= \mathrm{X}\Phi \mid \Phi_1 \ \mathrm{U}^{\leq k} \Phi_2 \mid \Phi_1 \ \mathrm{U} \ \Phi_2 \tag{2.17}$$

where $a \in AP$, $k \in \mathbb{N}$ and Φ is a state formula.

For a given path $\pi = s_0 s_1 s_2...$ in a discrete-time Markov chain $\mathcal{M} = (S, \mathbf{P}, p_{init}, AP, L)$ (generalising trivially to a Markov decision process):

- $\pi \models \mathrm{X}\Phi$ if, and only if, $s_1 \models \Phi$.

- $\pi \models \Phi_1 \ \mathrm{U}^{\leq k} \Phi_2$ if, and only if:

$$\exists i \leq k \text{ s.t. } (s_i \models \Phi_2) \wedge (\forall j < i : s_j \models \Phi_1) \tag{2.18}$$

- $\pi \models \Phi_1 \ \mathrm{U} \ \Phi_2$ if, and only if:

$$\exists i \geq 0 \text{ s.t. } (s_i \models \Phi_2) \wedge (\forall j < i : s_j \models \Phi_1) \tag{2.19}$$

Similarly to CTL, it is possible to define the operators F (eventually) and G (globally) as:

$$\begin{aligned} \mathrm{F}\Phi &= \texttt{true} \ \mathrm{U} \ \Phi \\ \mathrm{G}\Phi &= \neg\mathrm{E}(\neg\Phi) \end{aligned} \tag{2.20}$$

$\blacksquare$

While PCTL path formulas are defined above, it should be mentioned that all PCTL formulas are state formulas, and that path formulas should only appear inside the probabilistic operator.

In relation to the probabilistic operator in PCTL, the generalisation from the Markov chain to the Markov decision process can be done using the concept of adversary (also denoted strategy or scheduler), initially introduced in Section 2.3.1. In the case of an MDP, the operator $\mathbb{P}_{\rhd p}$ should be interpreted under the following semantics:

$$s \models \mathbb{P}_{\rhd p}(\varphi) \text{ if, and only if, } \forall \sigma \in Adv : Pr_s^{\sigma}(\pi \in Paths^{\sigma}(s) : \pi \models \varphi) \rhd p \qquad (2.21)$$

that is, the probability, from state s, that φ is true for an outgoing path satisfies $\rhd p$ for all adversaries $\sigma \in Adv$ (with Adv constituting the set of all adversaries that the MDP admits).

2.3.3 PRISM Language

PRISM is a probabilistic model checking tool developed in cooperation between the Universities of Oxford and Birmingham [14]. It was one of the first tools within stochastic model checking to be released and it is currently maintained by David Parker. In order to use PRISM (and other alternative model checking techniques), it is essential to be able to specify the model. To achieve this, the team behind PRISM developed the PRISM Language, initially described in [12].

A model described in the PRISM Language (typically a `.pm`, `.nm` or `.prism` text file) should contain a keyword that indicates the kind of model to expect. For example, the keyword `dtmc` indicates the model is a DTMC (discrete-time Markov chain) and `mdp` indicates a MDP (Markov decision process).

The PRISM language uses two different basic components to define a model *modules* and *variables*. A module is composed of a set of local variables whose behaviour is described by *commands*. A general command is of the form:

```
[] guard -> prob_1: update_1 + ... + prob_n: update_n;
```

where *guard* is a boolean expression over the variables, *prob_i* is the probability that *update_i* will be taken (with $\sum_i \text{prob\_i} = 1$) and *update_i* is the the new value of local and/or global variables if the guard is true and a transition is taken to this configuration.

A state of the module is defined as a configuration of its local variables, and a state of the model is a configuration of the states of all modules which compose it and the global variables.

In Listing 2.1, an example of a simple MDP model described in PRISM Language is shown. This example is comprised of two modules, `M1` and `M2`, each with their own local variables, and no global variables are declared. It should be noted that the variables in PRISM Language should either be integers (`int`) or booleans (`bool`) and can be initialised by specifying `init` after the initial declaration. Due to the fact that we are dealing with finite models, the integers must be in a finite range as well.

```
// Example 1 (modified)
// Two process mutual exclusion

mdp

module M1

    x : [0..2] init 0;

    [] x=0 -> 0.8:(x'=0) + 0.2:(x'=1);
    [] x=1 & y!=2 -> (x'=2);
    [] x=2 -> 1:(x'=2);
    [] x=2 -> 1:(x'=0);

endmodule

module M2

    y : [0..2] init 0;

    [] y=0 -> 0.8:(y'=0) + 0.2:(y'=1);
    [] y=1 & x!=2 -> (y'=2);
    [] y=2 -> 1:(y'=2);
    [] y=2 -> 1:(y'=0);

endmodule
```

Listing 2.1: Example of a simple MDP model from [1]

Models in the PRISM Language can also include *formulas*, which are expressions that avoid repetition of code, and *labels*, which are boolean expressions that identify sets of states of interest.

It is also possible to define rewards structures using the following convention:

```
rewards "name"
    guard: state_reward;
    ...
    [action] guard: transition_reward;
    ...
endrewards
```

where the *guards* are similarly defined as before, but the distinction between state and transition rewards is defined by the *action* (which may be empty).

2.3.4   Model Checking in DTMCs and MDPs

After the description of the model using a language such as the PRISM Language, it is possible to verify properties on the said model. These properties can be of different types according to the model and the model checking tool chosen.

In this dissertation, the models built will be represented through discrete-time Markov chains and Markov decision processes, so the properties of interest tend to be best expressed in the temporal logics LTL or PCTL. The PRISM model checker is one of the tools which allows a user to import a text file with the probabilistic model, build it and verify LTL and PCTL properties on it [14]. Another tool used for this purpose is the Storm model checker [9]. Each of these tools has it strengths and weaknesses. PRISM is an extremely robust tool, with lots of options in terms of engines to perform the required calculations, and a graphical interface. Storm is a relatively recent tool which builds on top of some of PRISM's features and introduces new ones, such as an optimised Sparse engine (a great advantage when dealing with models with a small state space) and conditional properties in DTMCs [9]. However, it does not provide a graphical interface and is limited in terms of capabilities and I/O. Due to the fact that Storm builds on top of PRISM, it uses the PRISM Language for model input and the properties are specified using the same convention.

In order to illustrate properties, consider the example of Listing 2.1. A PCTL property in this case could be defined as:

```
P>=0.5 [F x = 2]
```

This property can be interpreted as *"with probability greater or equal to 0.5, eventually the model reaches a state where* `x = 2`*"*. Building the model and verifying the property yields the value `false`, which means that, according to this model, there exists at least one adversary under which this property is not satisfied. Both PRISM and Storm allow for quantitative probabilistic properties as well. These properties take different formats in DTMCs and MDPs, as presented below.

```
// DTMC
P=? [F x = 2]

// MDP
Pmin=? [F x = 2]
Pmax=? [F x = 2]
```

Instead of returning a boolean value of satisfaction, these return the actual values of the probabilities of the properties being satisfied. They are, therefore, extremely useful metrics for model comparison. For the case of the DTMC, due to the fact that there is no nondeterminism, the probability of a path is uniquely defined (as seen in Section 2.3.2), and thus can be expressed simply as `P=?`. For MDPs, this is not the case. In order to verify the probabilistic operator $\mathbb{P}_{\rhd p}$ on the path formula φ, a model checker needs to calculate the following quantities:

$$
\begin{aligned}
p_{\max,s}(\varphi) &= \sup_{\sigma \in Adv} P_s^\sigma(\varphi) \\
p_{\min,s}(\varphi) &= \inf_{\sigma \in Adv} P_s^\sigma(\varphi)
\end{aligned}
\tag{2.22}
$$

where $P_s^\sigma(\varphi) = Pr_s^\sigma(\pi \in Paths^\sigma(s) : \pi \models \varphi)$.

Therefore, PRISM and Storm allow users to access these values by writing properties using `Pmax` and `Pmin` (boolean or quantitative).

Storm allows for conditional probability properties of the type:

18

```
P=? [ F x=2 || F y=2 ]
```

This property can be interpreted as *"what is the probability that the model eventually reaches the state* `x = 2`*, given that it eventually reaches the state* `y = 2`*"*. These properties, in DTMCs, correspond simply to the application of Bayes' theorem of conditional probability, which states that:

$$P_s(\varphi_1 \mid \varphi_2) = \frac{P_s(\varphi_1 \wedge \varphi_2)}{P_s(\varphi_2)} \tag{2.23}$$

For $P_s(\varphi_2) \neq 0$. It must be emphasised that this is **not** the case for MDPs, where the existence of multiple adversaries complicates the calculations of conditional probabilities.

This does not mean that it is not possible to reason over conditional probabilities in MDPs. For some specific scenarios, it is possible to obtain meaningful bounds which allow comparisons between the ones obtained in DTMCs and MDPs. One of those scenarios is defined in the proposition below, with the proof following it.

Proposition 2.3.1. Consider a Markov decision process $\mathcal{M} = (S, Act, \mathbf{P}, p_{init}, AP, L)$ which admits adversaries $\sigma \in Adv$. Let φ_1, φ_2 be two PCTL path formulas, such that, for any path $\pi \in Paths(s)$: $\pi \models \varphi_1 \Rightarrow \pi \models \varphi_2$. Assume as well that:

$$\sigma' \in \arg \sup_{\sigma \in Adv} P_s^\sigma(\varphi_1 \mid \varphi_2) \tag{2.24}$$

And that:

$$P_s^{\sigma'}(\varphi_1) = p_{\max,s}(\varphi_1) \tag{2.25}$$

That is, an adversary which maximises the probability of $\varphi_1 \mid \varphi_2$ also maximises the probability of φ_1. In this case it holds that:

$$p_{\max,s}(\varphi_1 \mid \varphi_2) \geq \frac{p_{\max,s}(\varphi_1)}{p_{\max,s}(\varphi_2)} \tag{2.26}$$

with:

$$p_{\max,s}(\varphi_1 \mid \varphi_2) = \sup_{\sigma \in Adv} P_s^\sigma(\varphi_1 \mid \varphi_2) \tag{2.27}$$

Proof. Consider $\sigma_2 \in Adv$ such that:

$$\sigma_2 \in \arg \sup_{\sigma \in Adv} P_s^\sigma(\varphi_2) \tag{2.28}$$

Assume, for the sake of contradiction, that:

$$p_{\max,s}(\varphi_1 \mid \varphi_2) < \frac{p_{\max,s}(\varphi_1)}{p_{\max,s}(\varphi_2)} \tag{2.29}$$

Under adversary σ', it can be written that, through Bayes' theorem of conditional probability:

$$p_{\max,s}(\varphi_1 \mid \varphi_2) = P_s^{\sigma'}(\varphi_1 \mid \varphi_2) = \frac{P_s^{\sigma'}(\varphi_1 \wedge \varphi_2)}{P_s^{\sigma'}(\varphi_2)} = \frac{P_s^{\sigma'}(\varphi_1)}{P_s^{\sigma'}(\varphi_2)} \tag{2.30}$$

Therefore, it follows that:

$$\frac{P_s^{\sigma'}(\varphi_1)}{P_s^{\sigma'}(\varphi_2)} < \frac{p_{\max,s}(\varphi_1)}{p_{\max,s}(\varphi_2)} \tag{2.31}$$

For this to be true, it must be that $P_s^{\sigma'}(\varphi_2) > p_{\max,s}(\varphi_2)$, since $P_s^{\sigma'}(\varphi_1) = p_{\max,s}(\varphi_1)$. This constitutes a contradiction, since in that case σ_2 would not maximise the probability of φ_2 ($p_{\max,s}(\varphi_2) = P_s^{\sigma_2}(\varphi_2)$, by definition), a violation of the assumption described in Equation 2.28. Thus, the proposition is proven. ∎

In MDPs, however, it is also possible to perform multi-objective verification, whereas multiple properties will be tested simultaneously (both in PRISM and Storm). One example of this would be:

```
multi(P>=0 [F x=2], P>=0.2 [F y=2])
```

The value of the verification of this property corresponds to the boolean conjunction of the individual properties. It is also possible to optimise over a single property while constraining other properties, restricting the possibilities in terms of outcomes. An example of this would be:

```
multi(Pmax=? [F x=2], P>=0.2 [F y=2])
```

Finally, it is also possible to perform multi-objective optimisation over the properties by defining them as (for example):

```
multi(Pmax=? [F x=2], Pmax=? [F y=2])
```

This leads to a situation where compromises between the optimisation of either one of the properties might have to be reached. Pareto curves are the result of the verification of such properties [7].

Definition 9. Multi-objective Query, Pareto Vector and Pareto Curve (alternatively Set)

A *multi-objective query* (MQ) ϕ of n objectives is a positive boolean combination of n predicates of the form $r_i \geq v_i$, where r_i is a reward function, $v_i \in \mathbb{Q}$ is a bound. The notation $\phi[\mathbf{x}]$, $\mathbf{x} \in \mathbb{R}^n$ is used to denote ϕ in which each $r_i \geq v_i$ is replaced by $r_i \geq x_i$

Consider a multi-objective query MQ ϕ of n objectives. The vector $\mathbf{x} \in \mathbb{R}^n$ is a *Pareto vector* of ϕ if, and only if:

1. $\phi[\mathbf{x} - \varepsilon]$ is achievable for all $\varepsilon > 0$, and

2. $\phi[\mathbf{x} + \varepsilon]$ is not achievable for any $\varepsilon > 0$

A *Pareto curve* of ϕ is the set of all Pareto vectors that ϕ admits.

■

Thus, the result of the verification of a multi-objective optimisation property corresponds to a Pareto curve (this curve might contain a single vector). Both PRISM

and Storm allow for the verification of properties containing up to 2 objectives each. As a result, 2D plots are the best way to represent these solutions. In Figure 2.4, the result of the verification of the property `multi(Pmax=?  [F G x=2], Pmax=?  [F G y=2])` is shown. As it can be observed, either property can be satisfied with certainty, but not both simultaneously (mutual exclusion). The green area under the curve corresponds to the achievable set.
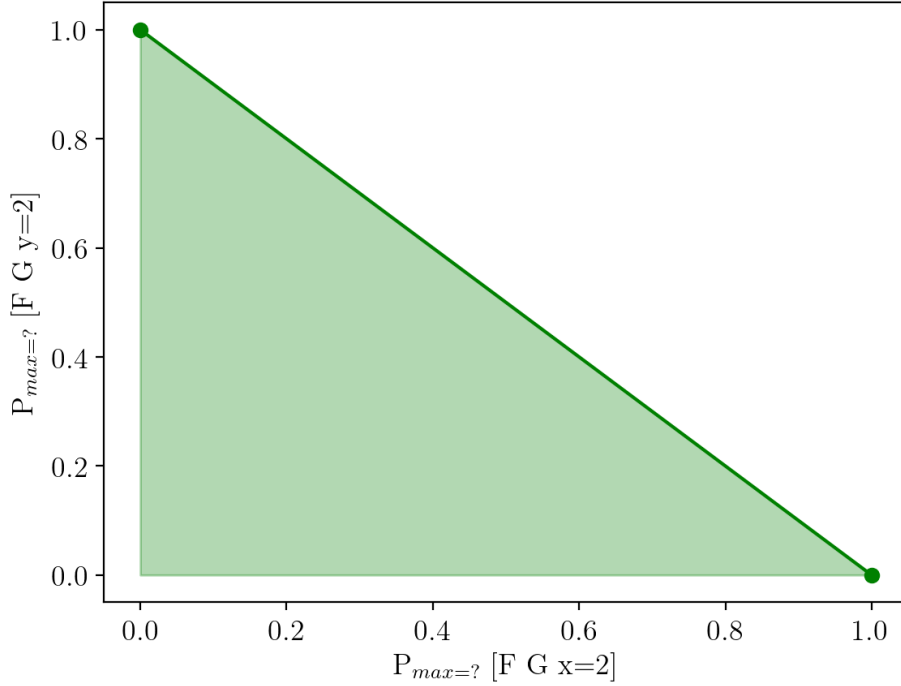
Figure 2.4: Example of a Pareto curve represented as a 2D plot.

Each of the values resulting of the verification of quantitative properties in MDPs (whether they be a single value from a single optimisation, or a Pareto vector) is the result of an adversary which solves the nondeterminism and converts the MDP into a DTMC [4]. The process of obtaining the adversary which satisfies certain properties is denoted by **strategy synthesis** or **adversary generation**, and it is formally defined below.

Definition 10. Strategy Synthesis (alternatively Adversary Generation)

Consider a Markov decision process $\mathcal{M} = (S, Act, \mathbf{P}, p_{init}, AP, L)$. The process of maximal (similarly minimal) *strategy synthesis* of the path formula φ is defined as a process that yields an adversary $\sigma \in Adv$ such that:

$$\sigma \in \arg\sup_{\sigma' \in Adv} P_s^{\sigma'}(\varphi) \tag{2.32}$$

(respectively $\arg\inf$ for minimal), where Adv is the set of all adversaries that $\mathcal{M}$ admits.

$\blacksquare$

Chapter 3

Human Driver Modelling

In this chapter, the human driver model that Salvucci presents in [21] is implemented and the metrics used to verify the model are established. Initially, the model is implemented in *Matlab* using the continuous driver model. To be able to verify it using the probabilistic model checking techniques presented in Chapter 2, the model undergoes an abstraction process, out of which a discrete-time Markov chain is obtained using several different approaches and assumptions. Metrics are then established to be able to evaluate whether the model meets the requirements and how well the human performs in a given situation. Finally, a simulator is presented in Section 3.4 in order to allow visualisation of paths in the model.

3.1 Continuous Driver Model using ACT-R

In order to obtain a continuous driver model, we interpret Salvucci's integrated driver model as presented in [21]. The scenario considered is the one Salvucci envisioned: a multilane highway with moderate (or low) traffic [21]. The model uses the three modules presented in Figure 2.2, that is, monitoring, decision making and control. All these modules rely on the constant update of the environment, as some are based on visual perception or low-level perception cues. While the human model is presented clearly in [21], the dynamics of the car (which is part of the environment) are left to the reader, as it is outside the scope of the paper. In that sense, some assumptions had to be made about the environment.

The main initial assumption is that the driver is perfectly aware of its surroundings,

and thus it is able to obtain the positions of other vehicles (within a certain distance) and the near and far points perfectly (this assumption is challenged later in the abstraction process). It is also assumed that the vehicle the driver is using is directly controlled by the updating of any pertinent variables within the control module of the architecture (i.e. there is no intermediate system or separation). The environment is assumed to change at the same rate as a cycle of the ACT-R model.

The overall view of the system is presented in Figure 3.1. The information flow is marked using the dotted arrows, while the sequential flow of the program is marked in the filled, lighter gray ones. Both the monitoring and the decision making make use of the perception which corresponds to querying the environment for the information (e.g. near and far point or another vehicle's position). The control both queries and sends updated information (e.g. vehicle position) to the environment. Modules are updated sequentially, following the order: control, monitoring, decision making and environment (e.g. position, velocity and acceleration of other vehicles).
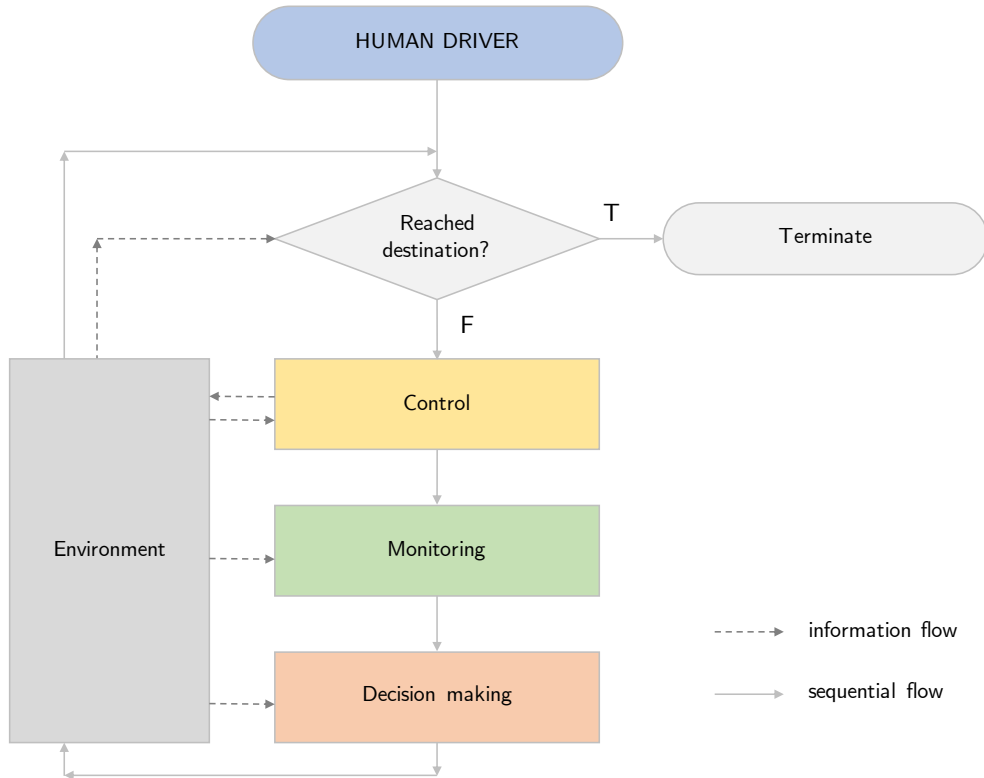

Figure 3.1: Continuous Driver Model overview.

The monitoring module is implemented according to the flowchart presented in Figure 3.2, from the description given in [21]. The threshold for monitoring is set at 0.2,

as this is the value suggested in Salvucci in [21]. The output of this process corresponds to altering the global variable a corresponding to the declarative memory cell composed of k chunks. No specific value for k is given by Salvucci in [21], but it is taken to be 8 from [15]. The function GET_DISTANCE is not described in depth due to the fact that it is trivial under the assumption stated in the previous paragraph.

The overall flowchart of the decision making module is shown in Figure 3.3. The high level part is similar to the flowchart presented in Figure 2.3, with small differences related to the loading of variables and information from the different memories available. It is worth noticing that the control makes use of the function LOOK_VEHICLE initially presented in the monitoring module to verify the presence of a vehicle in a relative position of a lane. The function SET_LANE_FOLLOWING is also omitted due to the trivial nature of this function under the assumptions made.

Finally, a flowchart for the control module is presented in Figure 3.4. This module is responsible for the calculation of $\Delta\varphi(t)$ and $\Delta\psi(t)$ and the update of the position. Given these two values, the position is updated following discrete laws of motion applied to rigid objects:

$$
\begin{aligned}
\mathbf{v}(t) &= \mathbf{v}(t-1) + \mathbf{a}(t)\Delta t \\
\mathbf{x}(t) &= \mathbf{x}(t-1) + \mathbf{v}(t-1)\Delta t + \frac{1}{2}\mathbf{a}(t)\Delta t^2 \\
t &= t + \Delta t
\end{aligned}
\tag{3.1}
$$

where $\mathbf{x}(t) = (x(t), y(t))$, $\mathbf{v}(t) = (v_x(t), v_y(t))$, and $\mathbf{a}(t) = (a_x(t), a_y(t))$, with:

$$
\begin{aligned}
a_x(t) &= \Delta\psi(t) \\
a_y(t) &= \sin(\Delta\varphi(t))
\end{aligned}
\tag{3.2}
$$

In this case, as Salvucci mentions in [21], $\Delta t = 0.5s$. It is also assumed that the vehicle does not speed, and therefore, $v_x \in [15, 34]\ m/s$ (between $50 km/h$ and $120 km/h$ - typical limits in highway speeds).

Using these assumptions, the continuous driver model is implemented in *Matlab* as presented in Appendix A.1.
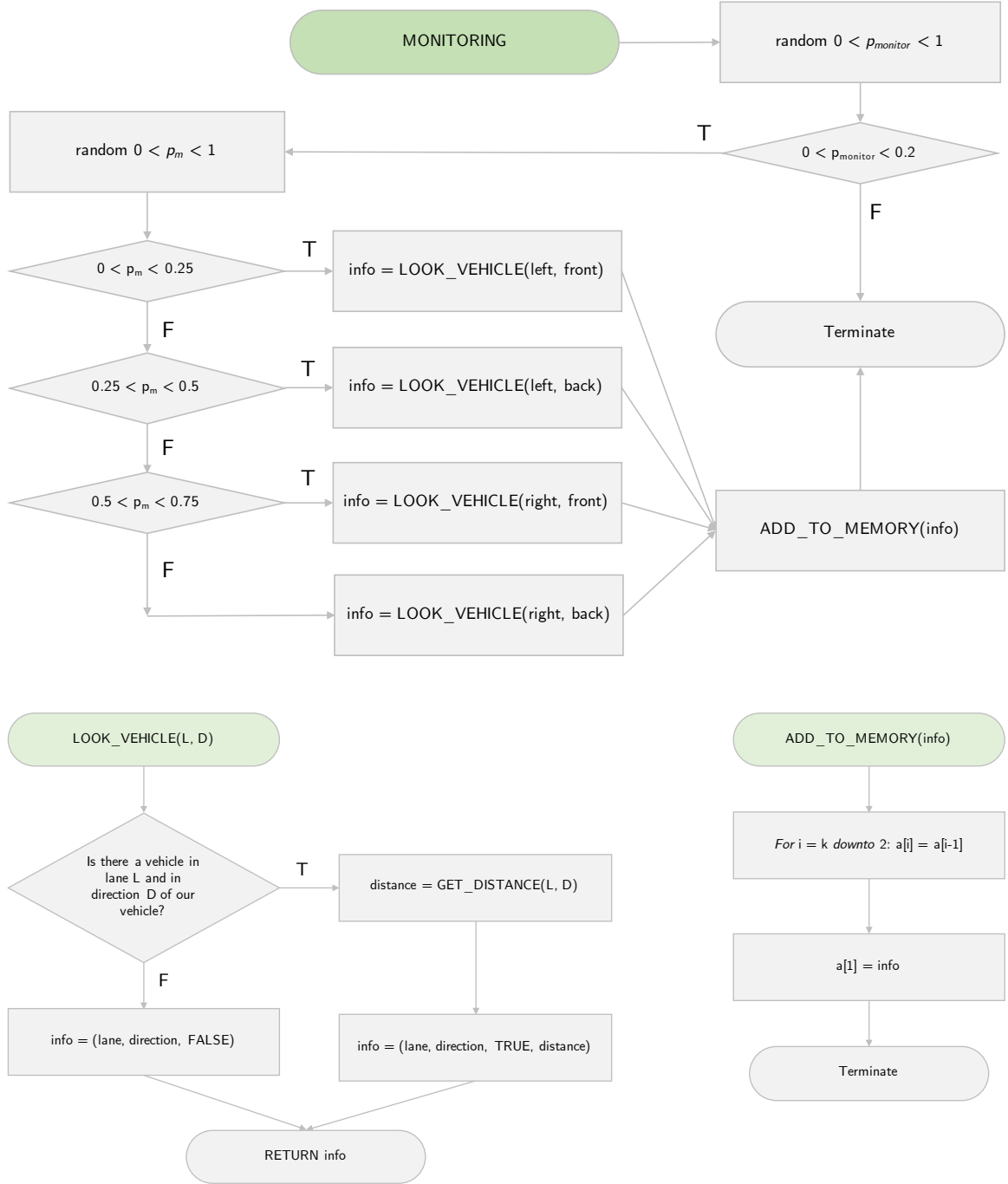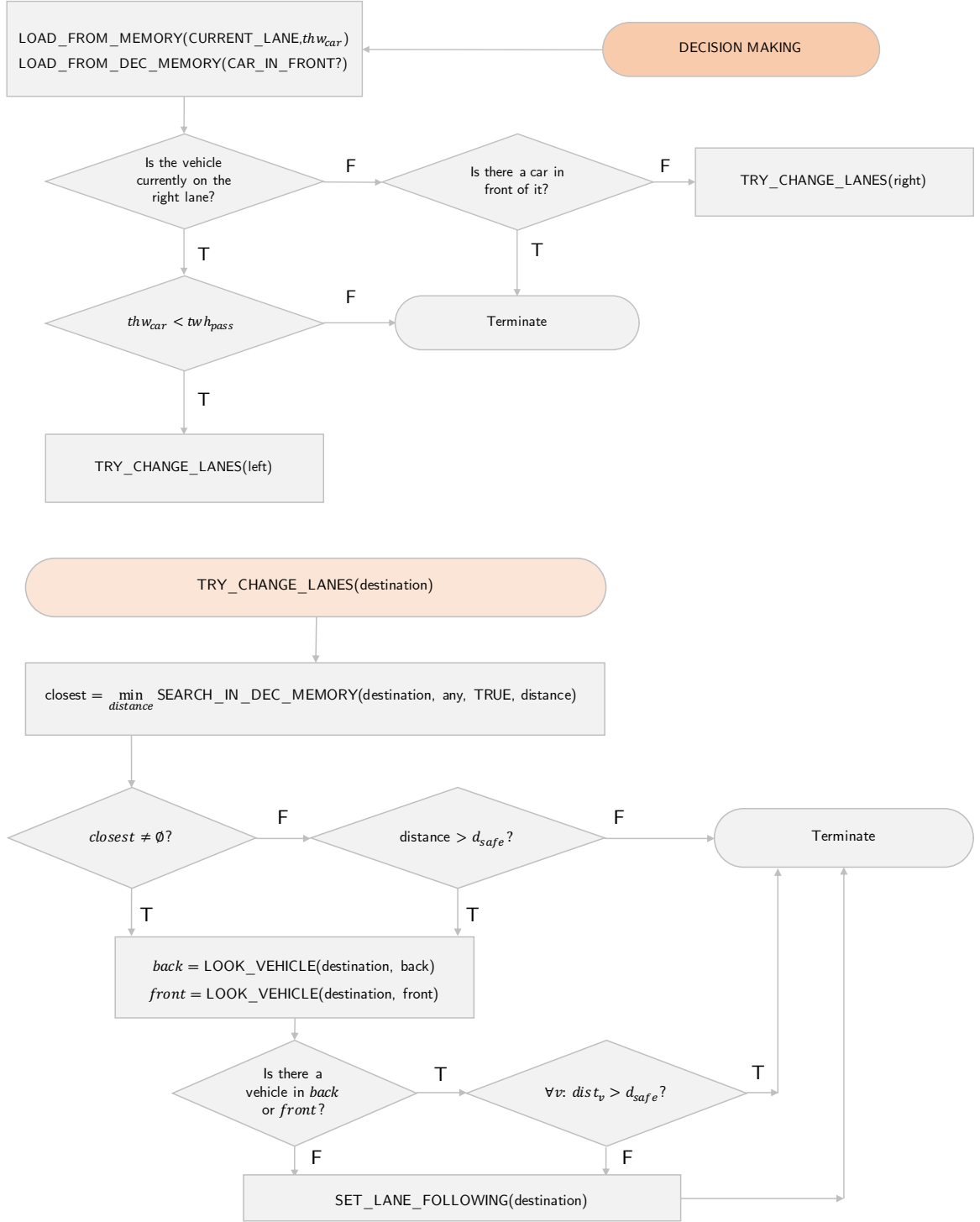
Figure 3.2: Flowchart of the Monitoring module.

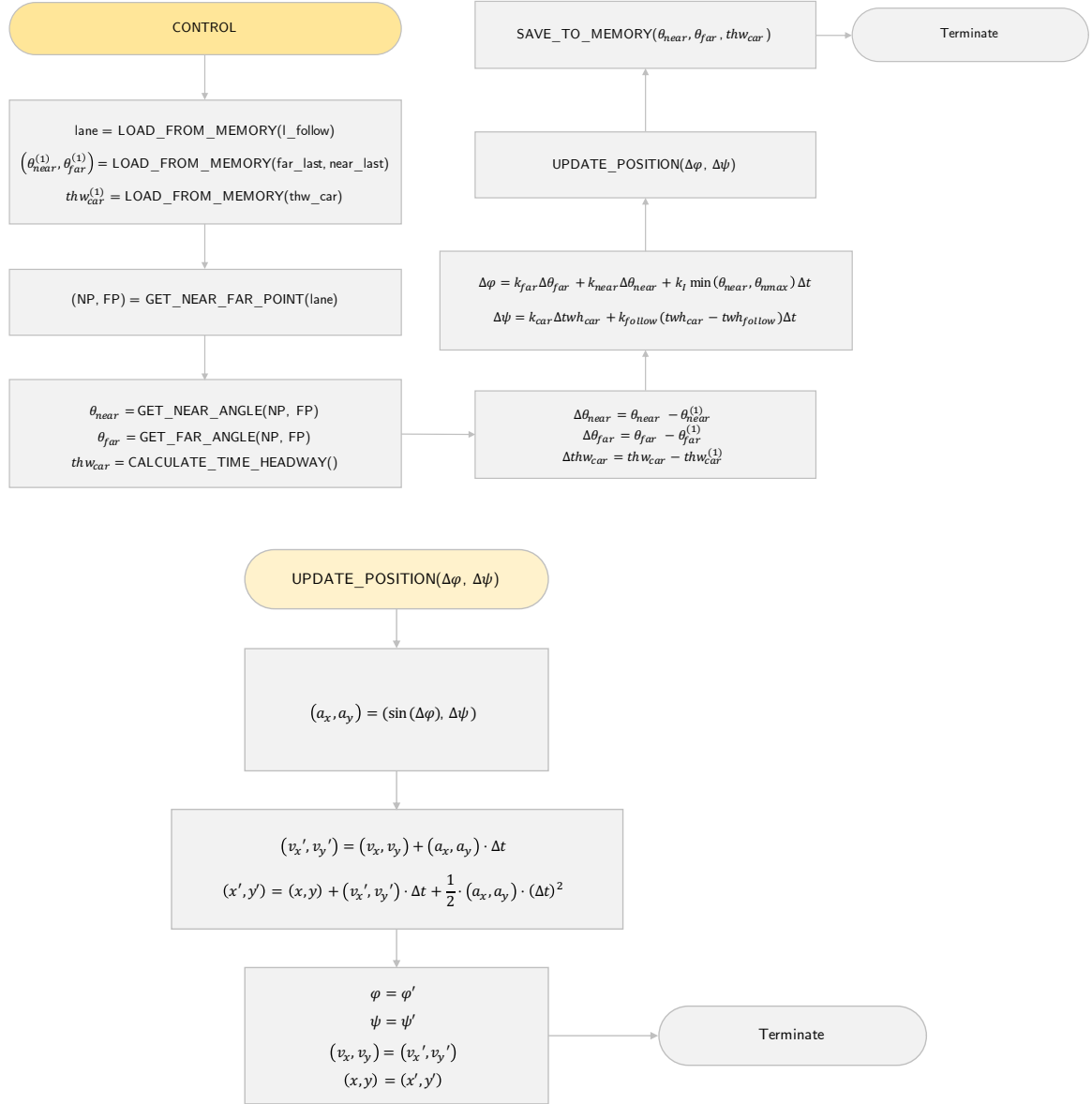Figure 3.3: Flowchart of the Decision Making module.

Figure 3.4: Flowchart of the Control module.

3.2 Model Abstraction

The process of abstraction consists in transforming the evolution of the continuous variables that constitute the model presented and discretising it in order to obtain a finite discrete model. In this case, the goal is to go from the continuous model of driver behaviour presented in the previous section to a discrete-time Markov chain (DTMC) which represents the same process. This transformation is less than trivial due to the fact that, in model checking of finite state abstractions, as the number of state variables in the system increases, the size of the system state space grows exponentially [8]. This problem is known as state explosion, and there has been some research into efficient ways of dealing with it [16, 19].

In the following sections, the abstraction process for the different modules of ACT-R is presented in detail, as well as the thought process behind the choices made.

3.2.1 Non-Probabilistic Control Module

The control module, as presented in the previous section and following Salvucci's approach in [21], is fully deterministic, in the sense that there is no probabilistic reasoning involved. It makes use of perception (for the near and far points) and two simple control laws which influence the position, velocity and acceleration of the vehicle in the environment.

The straightforward approach to this problem would be to represent it as a simple grid $N \times M$, in which a tuple (x, y) with $x \in \{0, ..., N\}$ and $y \in \{0, ..., M\}$ would describe the position of the vehicle and all other variables (i.e. v_x, v_y, a_x, a_y and t) would be integer versions of its continuous model representation. In this approach, the control laws could be applied directly over the grid and the results would be the movement of the vehicle in it. However, there is an intrinsic problem with this approach. The error associated with the use of the grid as a way to discretise space would, logically, be reduced with an increase of the N and M for a representation of the same segment (i.e. increase in resolution). For example, the error associated with a road segment of $500 \times 7m^2$ represented by $N = 500$ and $M = 7$ (each grid square corresponds to $1m^2$) would be significantly greater than if $N = 5000$ and $M = 70$ (each grid square corresponds to about $10^{-2} \ m^2$). The conclusion would be that one should increase N and M in order to obtain an appropriate resolution, but this

incurs in the problem of state explosion presented in the beginning of the section: as the variable ranges increase, the number of states in the system grows exponentially. Thus, while the situation of $N = 5000$ and $M = 70$ is appealing in terms of error, it would be intractable to actually build and perform model checking on it (e.g. for $v_x, v_y \in \{15, ..., 34\}^2$ and $a_x, a_y \in \{-3, ..., 3\}^2$, the state space of the control module alone would have approximately 6.89×10^9 states, making it impossible to add decision making and monitoring).

The straightforward strategy is not, by any standards, efficient in terms of state space and does not use any of the information of the problem to simplify it. In particular, one should notice in this case that the movement in the y direction is used simply for the purpose of lane changing, a process which is purely deterministic (as described in [21]). Therefore, a more interesting approach to the abstraction of the control module could take this into account and **pre-simulate the lane changing operations**.

In this approach, the vehicle's position is represented by two variables $x \in \{0, ..., N\}$ for a given N and $lane \in \{right, left\}$ (can be extended to more than two lanes), and the acceleration and velocity of the vehicle are simply $a = a_x$ and $v = v_x$. Time is still represented in the same way, with $t \in \{0, ..., max\_time\}$ If a lane change is decided by the decision making module, then, within one transition, the $lane$ variable switches its value and the variables x, v, a and t are updated to reflect this. These values are obtained from a look-up table and depend on the distance to the obstacle/vehicle in front of it (d) and the velocities of both the vehicle in question (v) and the obstacle/vehicle (v_1). Additionally, a variable $crashed$ is used to represent whether a vehicle has crashed in the process or not. The linear acceleration is implemented similarly using pre-simulation to determine the value of acceleration and the motion law is updated in the model itself.

The continuous model of driver behaviour obtained in the previous section in *Matlab* is used as basis for the simulation of lane changes and linear acceleration. From this model, the look-up tables are obtained which, given an origin lane (o_{lane}), a distance to the lead vehicle and the velocities of both the vehicle in question and the lead vehicle determines whether a crash happened or not, what is the Δx and ΔT incurred (how did the position in x changed and how long did it take), and the final velocity of the vehicle in question. The code presented in Appendix A.2 corresponds to a similar version of the control (but probabilistic, as presented later in this section).

An example of the simulation of the scenario $o_{lane} = 1$, $d = 20m$, $v = 15m/s$ and $v_1 = 20m/s$ is shown in Figure 3.5.
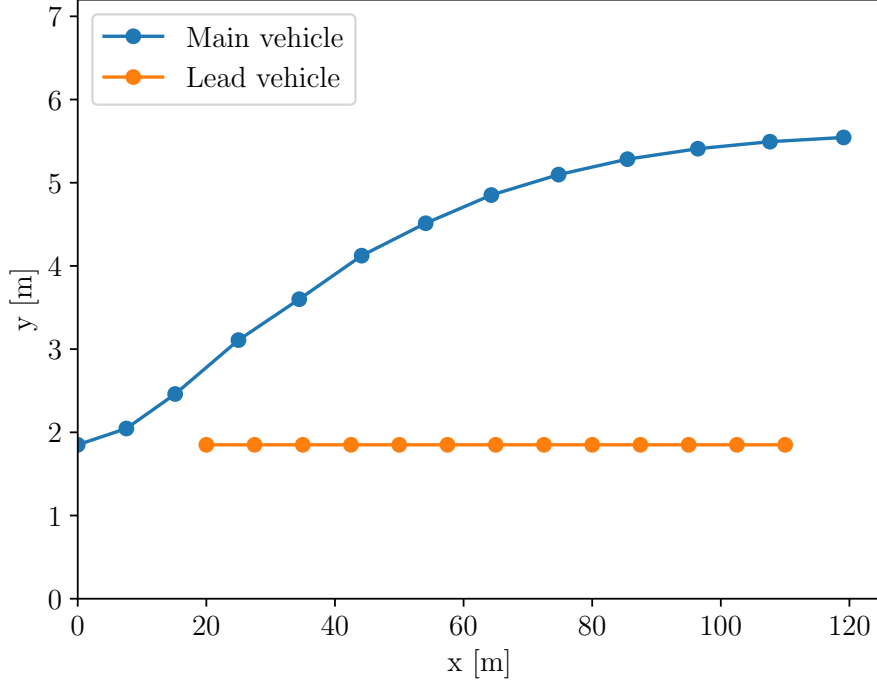
Figure 3.5: Example of the simulation of the lane change for $o_{lane} = 1$ (left lane), $d = 20m$, $v = 15m/s$ and $v_1 = 15m/s$. In this case, after the change is complete the variable assignments are $crashed = 0$ (no collision happened), $\Delta x = 119m$, $\Delta T = 6s$ and $v = 23m/s$.

This focuses the computational effort on the pre-calculations, simplifying the final model significantly in terms of the state space, while removing the error made by discretising in the y direction of the grid (the error in the x direction is still present though). With regards to the size of the lane changing table, for the considered scenario of 2 lanes, $v, v_1 \in \{15, ..., 34\}^2$ and for any discrete $d \in \{0, ...N\}$, the size of the table would be $2 \times 20^2 \times N = 800 \times N$. Given that the only value of the table that is effectively altered with a change in d is the value of $crashed$ (as the Δx, Δt and final v depend solely on the initial velocities of both vehicles), the table can consider:

$$d_{\max} < N, \text{ s.t. } \forall v, v_1 \in \{15, ..., 34\}^2, d' \geq d_{\max} : crashed_{d'} = false \qquad (3.3)$$

that is, a great enough value of d such that, regardless of the speed of the two vehicles,

no crash will occur between them. For the given ranges of v and v_1, $d_{\max}$ is determined to be 43, and thus the obtained look-up table contains $34,400$ rows.

The linear acceleration table depends on the thw_{car} which varies only with the distance to the front vehicle and the current velocity of the vehicle. Considering the distance in this case to be up to $80m$ (otherwise maximum acceleration will be applied), the look-up table generated has $1,600$ rows.

3.2.2   Decision Making and Monitoring Module

Due to the fact that, in the model presented by Salvucci in [21], the monitoring module influences the declarative memory which is then used exclusively for decision making purposes (and not for control), logically these two can be incorporated into one for abstraction purposes. Again, taking the straightforward approach of trying to include the declarative memory and the process associated with monitoring and decision making into the model directly would make it intractable.

In its simplest form, the decision making process localises the car and decides whether or not to change lanes based on the time headway, thw_{car}, to the car in front of it (if there is one). This value essentially corresponds to the time the main vehicle has before it crashes into the vehicle in front of it, assuming that vehicle came to a full stop ($v = a = 0$) [18]. The lower this time headway, the more likely a driver is to perform the manoeuvre. While Salvucci presents a fully deterministic driver in [21] based on the average driver, in this dissertation it was decided to try and analyse drivers according to different profiles in order to simulate how different parts of the population of drivers might make decisions. The decision making in such a case follows from a stochastic reasoning which can be simulated and incorporated in the model using look-up tables. In this case, it was decided to model the probability of a driver performing a lane change based on the time headway could be modelled using an exponentially decreasing function, such as:

$$\mathrm{P}[lC = true]_{thw} = P_{lC}(d, v) = e^{-\alpha \cdot thw} = e^{-\alpha \cdot d/v} \tag{3.4}$$

where α is a parameter unique to each population of drivers. It should be noted that this function, while logical, is an assumption of the abstraction, since the model

33

presented by Salvucci in [21] does not make any reference to this parameterisation. It is also worth referring that, while there was no real-world data involved in this project, this function can be learned from real data without changing the fundamental paradigm of the abstraction.

For the purposes of analysis, three classes of drivers are considered: **Aggressive**, **Average** and **Cautious** drivers (with different values for the parameter α). In Figure 3.6, the functions are presented for the three profiles of drivers assumed in this project.
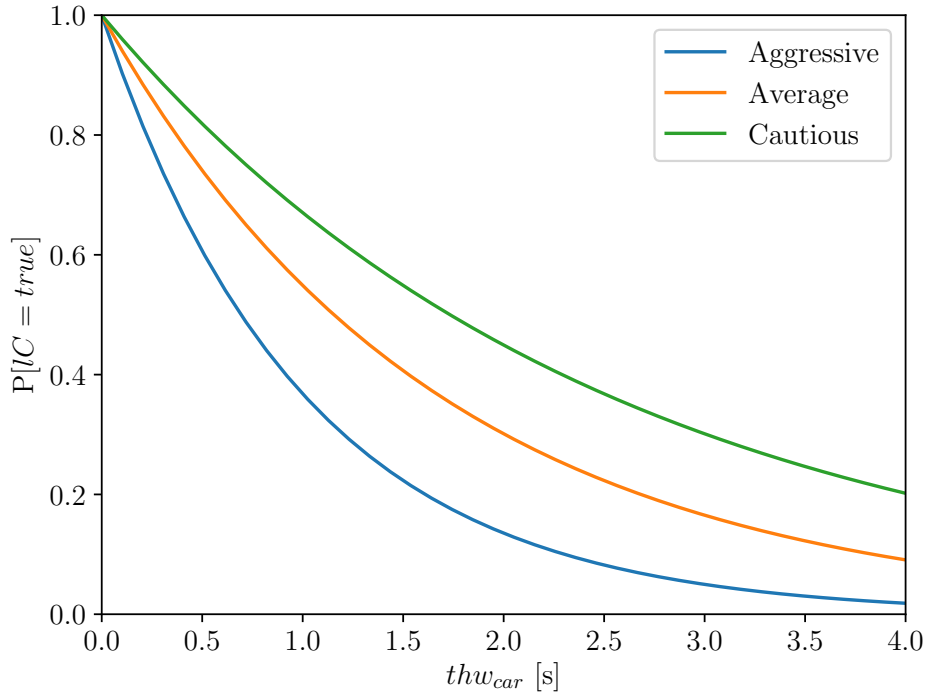
Figure 3.6: $P[lC = true]$ as a function of thw_{car} for Aggressive ($\alpha = 1$), Average ($\alpha = 0.6$) and Cautious ($\alpha = 0.4$) drivers.

However, this corresponds to a version of decision making which is not influenced by the monitoring module at all, and it relies on the correct calculation of the value of thw_{car}. This assumption is unrealistic, and in order to simulate these issues, it was decided that stochastic noise could be added to the measurement of the distance (as this is what human drivers have to instinctively measure through perception). It was chosen that this noise, n, would be normally distributed as:

$$n \sim \mathcal{N}(0, \sigma) \tag{3.5}$$

such that:

$$thw'_{car} = thw_{car} + n(thw_{car}) \tag{3.6}$$

Alternatively, this can be represented directly as (using integration steps of 1):

$$
\begin{aligned}
P'_{lC}(d, v) &= \sum_{i=-\infty}^{+\infty} [N(d+i+0.5) - N(d+i-0.5)] \cdot P_{lC}(d+i, v) \\
&= \sum_{i=-\max_d}^{+\max_d} [N(d+i+0.5) - N(d+i-0.5)] \cdot P_{lC}(d+i, v)
\end{aligned}
\tag{3.7}
$$

where $\max_d$ is the maximum length considered and $N(d)$ corresponds to the cumulative probability function (CDF) defined as:

$$N(d) = \int_{-\infty}^{d} n(t)dt \tag{3.8}$$

From this, a look-up table can be generated which, for different driver types, yields the probability of lane changing for a certain distance to the lead car and velocity. For the 3 driver profiles mentioned, considering $d \in \{1, ..., 80\}$ ($\max_d = 80$) and $v \in \{15, ..., 34\}$, the table generated has $4,800$ rows.

The same logic can be applied for a driver in the left lane overtaking a vehicle behind it in the left lane, except in this case the opposite effect will be seen in the decision making. In such a case, the probability of changing lane can be modelled as a normalised logarithmic function over the distance of the vehicles (it doesn't make sense to define time headway in this context):

$$\mathrm{P}[lC = true]_d = P_{lC}(d) = \frac{1}{\log(\beta * \max_d + 1)} \cdot \log(\beta * d + 1) \tag{3.9}$$

where $\max_d$ is the maximum length considered and β is a parameter unique to each population of drivers. In Figure 3.7, the functions are presented for the three profiles of drivers previously mentioned.
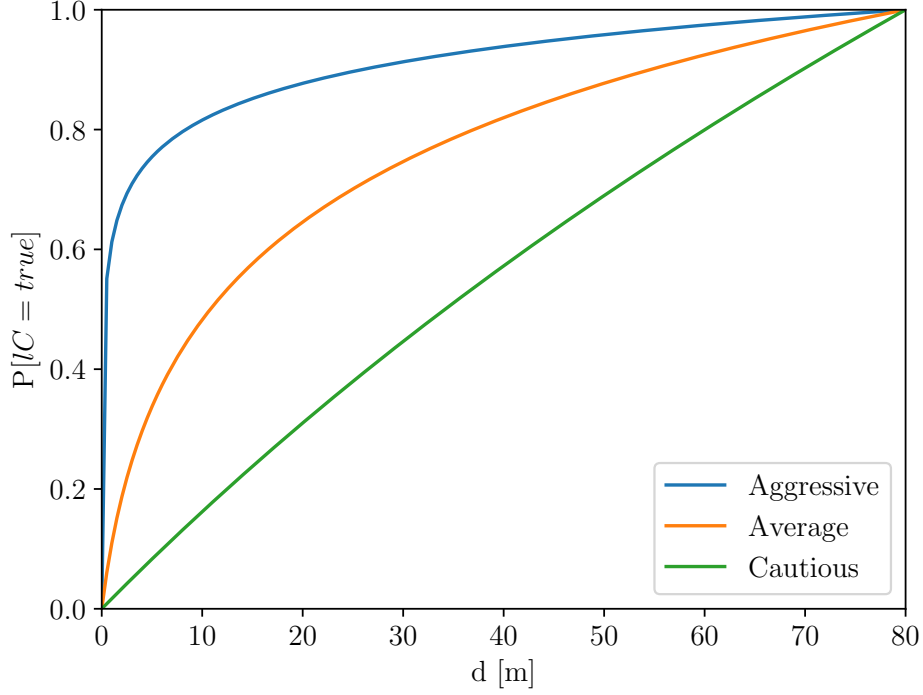
Figure 3.7: $\text{P}[lC = true]$ as a function of d for Aggressive ($\beta = 1000$), Average ($\beta = 0.5$) and Cautious ($\beta = 0.01$) drivers.

For the 3 driver profiles mentioned, considering $d \in \{1, ..., 80\}$ ($\max_d = 80$), this lookup table obtained has 240 rows. The unified decision making table has, therefore, $5,040$ rows.

The code presented in Appendix A.3 corresponds to the methods to obtain the lookup tables mentioned regarding the decision making and monitoring module.

3.2.3 Probabilistic Control Module

In light of the uncertainty introduced in the decision making and monitoring module regarding the visual perception of distance, it becomes only natural that this would be extended to the control module as well. In order to cope with this, a probabilistic version of the control module was designed, where the noise n presented in the

previous subsection is taken into consideration.

In order to obtain the equivalent look-up table for the probabilisitic control module, the noise can be simulated using repeated trials, and an average can be obtained for the crashing probability, the estimated final difference in position, the estimated difference in time and estimated final velocity of the vehicle. In the case of this dissertation, the tables generated (for linear and lane changing control) were the result of 100 trials.

The code presented in Appendix A.2 corresponds to the simulation of this probabilistic version of the control.

3.2.4 Unified Two-Module Model

Using the three tables generated (two for the probabilisitic control - linear and steering control - and one for the decision making), the final DTMC model unifies both using a sequential variable $actr\_state \in \{1, 2\}$, where $actr\_state = 1$ corresponds to the control and $actr\_state = 2$ corresponds to decision making. In the unified model, only two vehicles are present: one is the one controlled by the driver and another one starts at a distance $x = x_{1,0}$ from the main vehicle (which starts at $x = 0$) and moves with a constant speed of v_1. Its movement is, therefore, completely determined by the formula:

$$x_1(t) = x_{1,0} + v_1 \cdot t \tag{3.10}$$

The distance between the two vehicles can be written as:

$$d(t) = |x(t) - x_1(t)| \tag{3.11}$$

And the boolean function $posDist$ can be defined as:

$$posDist(t) = \begin{cases} \text{true,} & \text{if } x(t) \geq x_1(t) \\ \text{false,} & \text{otherwise.} \end{cases} \tag{3.12}$$

Due to the size of the tables generated, the models should be generated automatically for a given tuple of initial conditions $(d_{type}, v, v_1, x_{1,0})$ (where d_{type} is the driver class according to the ones defined in the decision making and monitoring subsection - 1 is aggressive, 2 is average and 3 is cautious), following a set of assumptions:

1. A transition between two states s and s' is possible if, and only if, the variable *actr_state* takes different values in s and s'.

2. If the vehicle has crashed, the model enters a deadlock.

3. If the vehicle is on the left lane and behind the other vehicle, it will not attempt a lane change.

4. If the vehicle is on the right lane and in front of the other vehicle, it will not attempt a lane change.

From these, it is possible to build a model generator, as presented in Appendix A.4.

By running the model generator using the conditions $(d_{type}, v, v_1, x_{1,0}) = (1, 30, 22, 51)$, that is, an aggressive driver, starting at $30m/s$ with another vehicle going at $22m/s$ and starting at $51m$, the resulting model is the one presented in Listing 3.1 (in PRISM Language).

```
//Model automatically built using model_generator.py for v1 = 22 and
    driver_type = 1 (to alter these values, run the script again).
//Generated on 31-07-2018 at 18:47.

dtmc

const int length = 500; // road length
const int driver_type = 1; // 1 = aggressive, 2 = average, 3 = cautious
    drivers - do not alter this manually!
const int max_time = 35; // maximum time of experiment

// Other vehicle
const int v1 = 22; // do not alter this manually!
const int x1_0 = 51;

// Environment variables
global t : [0..max_time] init 0; // time
global crashed : bool init false;
```

```
// Vehicle controlled
global actrState : [1..2] init 1; // active module: 1 = control (both cars
    ), 2 = decision making + monitoring
global lC : bool init false; // lane changing occuring?
global x : [0..length] init 0;
global v : [15..34] init 30;
global a : [-2..3] init 0;
global lane : [1..2] init 1;

formula x1 = x1_0 + v1*t;
formula dist = x1>x?(x1 - x):(x - x1);
formula positiveDist = (x < length)?x > x1:true;

module Decision_Making_Monitoring

  // If a crash occurs, then nothing else can happen
  //[] actrState = 2 & crashed -> 1:(crashed' = true);

  // If we are in lane 2, but behind the other vehicle, don't try to pass
  [] actrState = 2 & !crashed & lane = 2 & positiveDist = false -> 1:(
     actrState' = 1);

  // If we are in lane 1, and no vehicle is in front, don't change lanes
  [] actrState = 2 & !crashed & lane = 1 & positiveDist = true -> 1:(
     actrState' = 1);

  [] actrState = 2 & !crashed & lane = 1 & positiveDist = false & dist = 1
     & v = 15 -> 0.8:(actrState' = 1) & (lC' = true) + 0.2:(actrState' =
     1) & (lC' = false);

  ...

  [] actrState = 2 & !crashed & lane = 2 & positiveDist = true & dist >=
     80 -> 1:(actrState' = 1) & (lC' = true) + 0:(actrState' = 1) & (lC'
     = false);
endmodule

module Control

  // If we are in lane 1, and no lane change was decided, continue forward
       (which might result in crash)
  // The vehicle is behind the other driver (positiveDist = false, x < x1)
  [] actrState = 1 & !crashed & !lC & lane = 1 & x <= length - v & t < max
     _time & positiveDist = false & (x1 + v1 - x - v) >= 6 & v + a < 34 &
     v + a > 15 & dist = 1 & v = 15 -> 1:(x' = x + v) & (t' = t + 1) & (
     v' = v + a) & (a' = -2) & (actrState' = 2);
```

```
    ...

    [] actrState = 1 & !crashed & lC & lane = 2 & dist >= 43 & v = 34 & x >
        length - 136 & t > max_time - 6 -> 1:(crashed' = false) & (x' =
        length) & (v' = 34) & (t' = max_time) & (a' = 0) & (lane' = 1) & (
        actrState' = 2) & (lC' = false);

endmodule
```

Listing 3.1: Example of the model generated for the tuple $(d_{type}, v, v_1, x_{1,0}) = (1, 30, 22, 51)$ (shortned)

By loading and building the model in either PRISM or Storm, it is observable that it has 257 states and 295 transitions, corroborating the assumption that the abstraction methods used returned efficient outcomes. Due to the fact that there are $3 \times 20 \times 20 \times$ length (where length is the size of the road segment considered in meters) different possibilities in terms of initial conditions, it would be intractable, in the timeline of the dissertation, to obtain the number of states for all the combinations. Table 3.1 presents the number of states and transitions for some arbitrarily generated conditions.

d_{type}	v	v_1	$x_{1,0}$	# States	# Transitions
3	21	30	20	305	333
1	27	22	66	396	471
1	28	17	43	167	183
2	33	15	35	6	7
3	28	21	38	199	222
1	19	16	81	391	443
2	25	23	28	390	481

Table 3.1: Results of the number of states and transitions for arbitrary initial conditions.

3.3 Model Evaluation Metrics

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum

at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

3.3.1 Completeness Property

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

3.3.2 Safety Property

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere,

varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie
nulla fermentum.

3.3.3 Liveness Properties

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan
magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur portti-
tor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum
at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non
risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula
purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend
diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod
mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed
eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere,
varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie
nulla fermentum.

3.4 Simulation of Paths in the Model

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan
magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur portti-
tor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum
at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non
risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula
purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend
diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod
mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed
eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere,
varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie
nulla fermentum.

Chapter 4

Advanced Driver Assistance Systems

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

4.1   Driver Assistance System Design

4.1.1   Decision Making based ADAS

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula

purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

4.1.1.1 Decision Making with Fully Compliant Drivers

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

4.1.1.2 Decision Making with Partially Compliant Drivers

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

4.1.2 Control based ADAS

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

4.1.2.1 Active Linear Acceleration Control

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

4.1.2.2 Active Steering Control

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula

purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

4.2 Evaluation

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

4.2.1 Multi-Objective Metrics

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

4.2.2   Test Cases

4.2.2.1 $v = 21$ m/s, $v_1 = 19$ m/s, $x_{1,0} = 70$ m

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

4.2.2.2 $v = 30$ m/s, $v_1 = 22$ m/s, $x_{1,0} = 50$ m

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

4.2.3   Overall Comparison

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum

at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

Chapter 5

Experimental Results

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

5.1 Safety Metrics

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed

eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

5.2 Liveness Metrics

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

5.3 Discussion

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

5.4   Additional Experiments

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

Chapter 6

Conclusions and Future Work

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

Appendix A

Code for Human Driver Modelling

A.1   ACT-R Implementation (Matlab)

A.1.1   Main Loop

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

A.1.2   Control Module

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non

risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

A.1.3 Decision Making Module

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

A.1.4 Monitoring Module

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

A.2   Control Abstraction (Matlab)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

A.3   Decision Making and Monitoring Abstraction (Matlab)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

A.4 Probabilistic Two Module Model Generator (Python)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

A.5 Prism Automatic Model Checker (Python)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

A.6 Storm Automatic Model Checker (Python)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur portti-

tor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

A.7 Simulator (Python)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

Appendix B

Code for the Advanced Driver Assistance Systems

B.1 MDP Model Generators for Different ADAS (Python)

B.1.1 Perfect Decision Making

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

B.1.2   Imperfect Decision Making

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

B.1.3   Additive Linear Control

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

B.1.4   Additive Steering Control

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula

purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

B.2   Pareto Curve Generator (Python)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

B.3   Strategy Synthesis and Simulator (Python)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

B.4 Prism Automatic Model Checker (Python)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

Appendix C

Code for the Experimental Results and Evaluation

C.1   Plot Generator for Human Driver Model (Python)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

C.2   Plot Generator for the Decision Making and Full Control ADAS (Python)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan magna eu metus congue, vitae tristique enim commodo. Aliquam consectetur porttitor nisl, eget pulvinar quam venenatis eget. Aliquam erat volutpat. Nullam at ipsum

at felis tincidunt consequat non sed mi. Morbi placerat iaculis dapibus. Aliquam non risus ac urna interdum pretium. Donec maximus, massa vitae suscipit congue, ligula purus sagittis magna, vitae interdum enim dui sed felis. Curabitur mollis eleifend diam a varius. Vivamus eu porttitor lorem. Mauris sagittis ultrices ligula, a euismod mi efficitur a. Aenean molestie sapien a nibh suscipit, nec venenatis urna varius. Sed eu ex id lacus tincidunt ultricies. Aenean quam massa, dignissim ac suscipit posuere, varius sit amet turpis. Nunc semper turpis ullamcorper elit venenatis, in molestie nulla fermentum.

Bibliography

[1] PRISM Manual | The PRISM Language | Example 1, Dec 2010.

[2] John R Anderson. *The Architecture of Cognition*. Psychology Press, 2013.

[3] John R Anderson, Michael Matessa, and Christian Lebiere. ACT-R: A theory of higher level cognition and its relation to visual attention. *Human-Computer Interaction*, 12(4):439–462, 1997.

[4] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[5] Tina Balke and Nigel Gilbert. How do agents make decisions? A survey. *Journal of Artificial Societies and Social Simulation*, 17(4):13, 2014.

[6] Ewin R Boer and Marika Hoedemaeker. Modeling driver behavior with different degrees of automation: A hierarchical decision framework of interacting mental models. In *Proceedings of the 17th European annual conference on human decision making and manual control*, pages 63–72, 1998.

[7] Taolue Chen, Vojtěch Forejt, Marta Kwiatkowska, Aistis Simaitis, and Clemens Wiltsche. On stochastic games with multiple objectives. In *International Symposium on Mathematical Foundations of Computer Science*, pages 266–277. Springer, 2013.

[8] Edmund M Clarke, William Klieber, Miloš Nováček, and Paolo Zuliani. Model checking and the state explosion problem. In *Tools for Practical Software Verification*, pages 1–30. Springer, 2012.

[9] Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A storm is coming: A modern probabilistic model checker. In *International Conference on Computer Aided Verification*, pages 592–600. Springer, 2017.

[10] Vojtěch Forejt, Marta Kwiatkowska, Gethin Norman, and David Parker. Automated verification techniques for probabilistic systems. In *International School on Formal Methods for the Design of Computer, Communication and Software Systems*, pages 53–113. Springer, 2011.

[11] Iuliia Kotseruba and John K Tsotsos. A review of 40 years of cognitive architecture research: Core cognitive abilities and practical applications. *arXiv preprint arXiv:1610.08602*, 2016.

[12] Marta Kwiatkowska, Gethin Norman, and David Parker. Prism: Probabilistic symbolic model checker. In *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 200–204. Springer, 2002.

[13] Marta Kwiatkowska, Gethin Norman, and David Parker. Stochastic model checking. In *International School on Formal Methods for the Design of Computer, Communication and Software Systems*, pages 220–270. Springer, 2007.

[14] Marta Kwiatkowska, Gethin Norman, and David Parker. Prism 4.0: Verification of probabilistic real-time systems. In *International conference on computer aided verification*, pages 585–591. Springer, 2011.

[15] Chak Lam. Driver assistance using cognitive modelling and strategy synthesis.

[16] Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. In *ISAIM*, 2006.

[17] Andrew Liu and Dario Salvucci. Modeling and prediction of human driver behavior. In *Intl. Conference on HCI*, 2001.

[18] Highway Capacity Manual. Highway capacity manual. *Washington, DC*, 11, 2000.

[19] Austin Mohr. A survey of state abstraction techniques for markov decision processes.

[20] Dario Salvucci, Erwin Boer, and Andrew Liu. Toward an integrated model of driver behavior in cognitive architecture. *Transportation Research Record: Journal of the Transportation Research Board*, (1779):9–16, 2001.

[21] Dario D Salvucci. Modeling driver behavior in a cognitive architecture. *Human factors*, 48(2):362–380, 2006.

[22] Dario D Salvucci and Andrew Liu. The time course of a lane change: Driver control and eye-movement behavior. *Transportation research part F: traffic psychology and behaviour*, 5(2):123–132, 2002.

[23] Ron Sun. The importance of cognitive architectures: An analysis based on CLARION. *Journal of Experimental & Theoretical Artificial Intelligence*, 19(2):159–193, 2007.