Expert System and a Rule Set Development Method for Urban Behaviour Planning

by

Frédéric Bouchard

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2020

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Today, autonomous vehicles have the capacity to achieve fully autonomous driving in predefined environments. This ability can be in part attributed to advancements in motion planning, which plans the vehicle' behaviours and navigation through complex environments. This thesis introduces a novel hierarchical expert system architecture along with a rule set development method for expanding an operational design domain. In the method, the knowledge engineering is tool-assisted and supports semi-automatic rule creation based on test cases. Additionally, the method incorporates a qualitative analyzer that probes the maintainability and the run time efficiency of the rule set. Moreover, the proposed architecture and method are successfully applied to implement a behavioural planner for an actual autonomous vehicle. The thesis also describes additional strategies to address noisy perception, avoid jittery behaviour, and improve the overall run time efficiency, which were necessary to achieve satisfactory performance of the planner on the road. This system was tested and proven effective in an open road test, which involved over 110 kilometres of autonomous driving in populated urban environments. During the open road test, 58 interventions were required due to perception noise or limitations arising by the small range of the lidar sensor. Finally, the strengths and weaknesses of the proposed methodology and architecture, along with an outlook on the role rule-based planning in autonomous driving, are discussed.

Acknowledgements

This work was partially supported by Fonds de Recherche du Québec – Nature et Technologies (FRQNT). I would like to thank colleagues and friends who made this thesis possible, in particular, Marko Illevski who helped me to draft the nomenclature and Ashish Gaurav who dig down the mathematics to accurately express the algorithms presented.

Table of Contents

List of Figures

List of Tables

List of Symbols

(M_t, C_t) The adopted behaviour is the output of the Rule-Engine. It is composed by the highest priority maneuver M_t derived from the maneuver rules $R(\mathcal{W}_t) \vdash \mathcal{M}_t$ in which the constraints have been resolved by the constraint rules $R(\mathcal{C}_t) \vdash C_t$. 12

$\mathcal{B}$ The behavioural test suite is a serie of integration test allowing the knowledge expert to certify the behaviour of the autonomous vehicle encountering a tested situation. 57, 59–61, 63, 64, 72, 75–77

$(W_k, R_m, R_c, M^*, C^*)$ A situation adequacy test is a type of test contained in the behavioural test suite, $\mathcal{B}$, which assess if the valuation of the World Abstraction, $\mathcal{W}_k$, described by the maneuver rules R_m and resolved by the constraint rules R_c, produce the expected maneuver, M^*, along with the expected set of constraints, C^*. 58

(W_k, R_m, M^*, C^*) A situation test is a type of test contained in the behavioural test suite, $\mathcal{B}$, which assess if the valuation of the World Abstraction, $\mathcal{W}_k$, described by the maneuver rule R_m produce the expected maneuver, M^*, along with the expected set of constraints, C^*. 57

$\mathcal{C}_t$ The atomic considered constraints set is derived by the constraints produced by the maneuver rules $R(\mathcal{W}_t) \vdash \mathcal{M}_t$ where the maneuver match the selected maneuver M_t. The set contains a Boolean representation of the distinct constraints produced by the maneuver rules. 12, 25, 26, 54

C^* The expected constraints are the constraints associated with the expected maneuver, M^*, that should be produced by the Rule-Engine when he is facing a given valuation of the World Abstraction, $\mathcal{W}_k$. 57, 58

M The maneuver set agglomerates the maneuvers known by the Local Planner interface which is $M := \{track\text{-}speed, follow\text{-}leader, overtake, decelerate\text{-}to\text{-}halt, yield, stop, emergency\text{-}stop\}$ 11, 12, 25

M_t The selected maneuver is the highest priority a maneuver achievable at time step t taken from the ordered maneuver set M where *track-speed* as the lowest priority. 12, 24, 25

M^* The expected maneuver is the maneuver that should be produced by the Rule-Engine when he is facing a given valuation of the World Abstraction, $\mathcal{W}_k$. 57, 58

$\mathcal{M}_t$ The potential behaviours is the set of achievable constrained maneuvers at time step t. This set constitutes the collection of output produced by the maneuver rules $R(\mathcal{W}_t) \vdash \mathcal{M}_t$. 12, 23, 24

$\mathcal{M}_{i,t}$ A potential behaviour is the output produced by the evaluation of a maneuver rule which, more specifically, is a tuple composed by a maneuver and its constraints set at time step t. A potential behaviour may be adopted, partially adopted, or rejected by the ego vehicle. 23

$R(\mathcal{W}_t) \vdash \mathcal{M}_t$ The maneuver rules evaluates the atomic world abstraction $\mathcal{W}_t$ and output the set of potential maneuvers with their constraints. 12

$R_i(\mathcal{W}_t) \vdash \mathcal{M}_{i,t}$ A maneuver rule evaluates the atomic world abstraction $\mathcal{W}_t$ and output a potential maneuver with its constraints. 23

$(\mathcal{W}_t - \mathcal{W}_{t-1}) \cap \mathcal{W}_i \neq \varnothing$ A maneuver rule is only evaluated when the atomic world abstraction, $\mathcal{W}_t$, temporal difference leads to an non empty set of atomic proposition when comparing with the clauses required for the rule evaluation. 23

$R(\mathcal{C}_t) \vdash C_t$ The constraint rules evaluates the atomic considered constraint set $\mathcal{C}_t$ and output the restrictions that the autonomous vehicle must take into account when performing the selected maneuver M_t. 12

t The time step is a unit of time at which the Behaviour Planner produces a behaviour to undertake. 11, 12, 21, 23, 38, 40, 60, 68, 102

$t-1$ The previous time step is a unit of time at which the Behaviour Planner has produced a behaviour to undertake. 23, 38

W The world abstraction model is a mixed set containing continuous and discrete values describing the environment at a high level. 57

$\mathcal{W}_k$ A valuation of the World Abstraction, W, is valuation of a subset of the model attributes. The valuation should be similar to the disjunctive normal form of one-to-many maneuver rules. 57, 58, 61, 62, 64, 65

W_t The world abstraction model at time step t is a value assignment of the world abstraction model W. 11, 20, 21, 23, 35, 39, 53, 64

$\mathcal{W}$ The world abstraction atomic model is the atomic representation of the World Abstraction Model. The set is constituted by a mixture of atomic temporal properties and all the attributes included in the world abstraction model including their Boolean version and some combination memorised to boost the run time efficiency. 81

$\mathcal{W}_t$ The atomic world abstraction at time step t is a value assignment of the world abstraction atomic model $\mathcal{W}$. 11, 21, 23, 25, 64

$\mathcal{W}_{t-1}$ The previous atomic world abstraction at time step $t-1$ is a value assignment of the world abstraction atomic model $\mathcal{W}$. 23

$\mathcal{W}_i \subseteq \mathcal{W}$ The clauses of the maneuver's rules are subset of the world abstraction atomic model, $\mathcal{W}$ 23

$\mathcal{W}_t \subseteq \mathcal{W}$ The world abstraction atomic model conversion at time step t is a value assignment of the world abstraction atomic model $\mathcal{W}$. 23

Chapter 1

Introduction

Recently, a large community of researchers and practitioners have been directing their efforts towards the development of a fully autonomous vehicle. Today, there exist autonomous vehicles capable of driving in predefined environments with multiple road users and with limited human intervention. The recent advancements in this field can be attributed to the improvements in both the perception and adaptation to a complex urban environment. This adeptness capability is typically handled by the motion planning system of the autonomous vehicle.

The motion planning problem can be defined as providing a feasible reference trajectory to be executed by the autonomous vehicle through the environment. This trajectory is subject to requirements of safety, comfort, traffic regulations and continued progress towards a goal, which can be executed by the vehicle hardware, considering the vehicle's dynamics [1].

The motion planning problem is computationally difficult to solve and is thus, typically decomposed into three sub-problems [2]: (i) mission planning; (ii) behaviour planning; and (iii) local planning. In this thesis, the focus is on behaviour planning, which generates a sequence of high-level driving maneuvers to safely and efficiently navigate through the environment towards the specified goal.

Traditionally, behaviour planning algorithms were implemented with state machines that relied on environmental triggers to transition to the required maneuver [3]. Meanwhile, an alternative implementation based on the Rete family of expert systems has also proven that a knowledge base and inferential mechanism can facilitate the coverage of a small operational design domain [4]. However, in recent years, machine learning techniques for behaviour planning have become an active area of research, especially following the

success of deep learning. For instance, Bojarski et al. demonstrate the transformation of raw sensor data directly to steering control inputs without any explicit rules or logic [5], whereas Abbeel et al. use demonstrations from a human driver to infer good driving paths [6].

Although machine learning approaches show a lot of promise, there are two main issues that hinder their practicality. Firstly, even simple machine learning networks that tackle behaviour planning require a large amount of data to train [7]. To be effective, this data must be relevant, error-free, and often accurately hand-labelled, which is costly. Secondly, the insufficient explainability of the results produced by a machine learning agent complicates requirements engineering. Despite interpretability being an active area of research [8], state-of-the-art machine-learned models are currently largely black boxes. They lack reliable methods for explaining their decisions and assuring proper behaviour when presented with new inputs, and ultimately for guaranteeing the safety and reliability of the learned driving policy.

Considering these drawbacks, this thesis introduces a behaviour planning approach based on a novel hierarchical expert-system design, which tackles knowledge acquisition without requiring large datasets. The driving policy development strongly relies on the interpretability of the inference mechanism used in the system, and supports incremental growth of the operational design domain. To ensure the scaleability of the proposed architecture, a tool-assisted method is presented, which is based on an analysis framework that agglomerates metrics to assess and reduce the number of rules by maximizing their utility. In addition to providing insights about the scaleability of a knowledge base, the effectiveness of the proposed methodology is demonstrated by autonomously driving 110 kilometres in populated urban environments. Although 58 interventions were required during this demonstration, none of them were due to erroneous behavior planning, but rather due to perception noise or range limitations of the used lidar sensor.

The design objectives for the proposed behavior planning approach are described in Chapter 2. Chapter 3 presents the high-level architecture of the proposed design, along with the notation used throughout this thesis. Chapter 4 describes the proposed inferential process, illustrated by deriving the track-speed maneuver in a simple scenario without interaction with dynamic objects. Interactive behaviours are then introduced in Chapter 5, where we detail the preprocessing mechanism that deal with partial observability. Then, Chapter 6 discusses a method to maximize the coverage of a behavioural test suite while minimizing the number of situations to model. The chapter ends with a collection of metrics for measuring the quality of the domain knowledge. Chapter 7 describes the discrepancy detection task, which is used to identify missing requirements and calibrate or extend the domain knowledge. Furthermore, we demonstrate the practicality of the

proposed architecture and report on the performance outside a virtual environment, by discussing in Chapter 8 the results of a successful implementation on an autonomous vehicle. Chapter 9 discusses the fulfillment of the design objectives that guided the creation of the proposed system design and development method. Chapter 10 relates our design and method with important works in the literature covering behaviour planning, expert systems, and world abstraction models. Finally, the thesis ends with Chapter 11, which summarizes the presented work and suggests future research.

The contributions of this thesis include: (i) The specialized rule-based design using two stratified rule sets: one for maneuver decision and a subsequent one for maneuver constraint determination, with parallel evaluation of rules in each set. Further, the design uses a precedence table to resolve conflicting maneuver decisions. The stratified design avoids the understandability and maintainability challenges of the existing general-purpose rule-based systems, which use ordered rules and forward chaining. The design also optimizes performance by eliminating unnecessary rule executions. (ii) A tool-assisted quality assurance method to assess the efficiency and maintainability of the rule sets. (iii) A semi-automated rule creation algorithm which updates the knowledge base to satisfy a behavioural test suite. (iv) A proof of concept implementation of the proposed design and method involving an open-road test, with no disengagements due to behavioral planning errors.

Chapter 2

Design Objectives

Driving is a complex task for a robot, necessitating an incremental development approach for the behavior planner. During development, the behavior planner functionality needs to be adjusted and extended on a case-by-case basis [9]. Such long-term maintenance poses significant methodological challenges, including correctness, efficiency, interpretability, and maintainability.

Correctness: The development cycle of a behaviour planner requires to incrementally design the proper behavior within its operational design domain. This behavior is subject to requirements of safety, comfort, traffic regulations and continued progress towards a goal. The behavior planner design and development method should provide some way of specifying desired behavior and ensuring that behavior is implemented correctly.

Efficiency: Since driving requires decisions to be taken in a timely manner and on-board computing resources are limited, the behaviour planner should minimize its execution time, in addition to providing a recovery mechanism to ensure that stacking requests can be merged to avoid producing outdated decisions.

Interpretability: At any time, a user of the system might ask, "Why the behaviour planner selected such a behavioural sequence?". To increase the user trust in the system, the behaviour planner should offer behavioural inspection. During those inspections, the behaviour planner should highlight the causal links from the important properties of the input world model to the output behaviour. Indeed, this causality should be understandable by the developers.

Maintainability: The large list of situations to be supported by the planner are rarely known in advance. Thus, the planner design and development method should support

adjusting the behavior in covered situations and the incremental addition of new situations, while ensuring that existing desired functionality is preserved.

Given these methodological challenges, we opted to develop an expert system. Expert systems represent knowledge using a set of declarative rules. They use the rules and a set of atomic propositions, along with an inference mechanism, to deduce new propositions. Expert systems aid interpretability since the decision making can be defined by a set of easy-to-understand rules. In particular, the conjunctive clauses of each rule represent the decision boundaries of the system, which aid both understanding and testing.

Further, we also decided to use test-driven development to address the correctness and maintainability challenges. Consequently, the proposed development method uses a behavioral test suite to specify and assess the behavioral decisions. Black box tests are inexpensive and ensure the consistency of the outputs since the developers can count the number of failing tests to approximate the impact of each update to the rules.

We also decided to create a specialized design for the rule-based system to fit the needs of behavior planning. Existing rule-based approaches, such as the Rete family of algorithms [10], represent knowledge by a set of ordered rules. This ordering is used in forward chaining to provide execution semantics for the rules, which also allows controlling rule precedence by ordering. Such design negatively impacts interpretability and extensibility, since any permutation of the rules might affect the output of the system, that is, the complexity of the rule base grows with the factorial of the number of rules. Rete systems commonly use alpha and beta caches to allow the system to discover which of the previous rules should be revaluated. Unfortunately, the resulting system is becomes more difficult to trace and comprehend. The re-evaluations also add computational cost.

To address the challenges of the existing rule-based systems, the proposed design implements inference using rule stratification and a precedence table. In particular, we use one set of rules for maneuver decisions and a subsequent one to decide constraints on maneuvers. Stratification prevents cycles in rule chaining, which would be difficult to comprehend. Further, the rules are unordered, which eliminates the need to reason about order. Instead, a precedence table is used to resolve conflicting decisions of a set of rules, which are evaluated in parallel. This design aids to address correctness, interpretability, and extensibility challenges. Finally, the design also optimizes the rule execution, elminating the need to execute rules whose output would not change given the input of the current execution cyle. This optimization helps addresses the efficiency challenge.

Further, the proposed development method, in addition to exploiting test-driven development, provides techniques to improve efficiency, interpretability, and extensibility. In particular, it offers set of metrics to assess the efficiency and complexity of the rule base,

and it also provides tool support to explain decisions and to aid extensions by synthesizing rule conditions and rule improvements.

In the next three chapters, we present the architecture of our expert system that uses rule stratification. The chapters are organized to describe the system from high to low levels. Then the subsequent two chapters describe the development method.

Chapter 3

High-Level Architecture

The behaviour planner has the responsibility to determine the most beneficial maneuver which can be executed by the local planner considering the environment representation, subject to driving constraints. Temporally, the sequence of constrained maneuvers produced by the behaviour planner should be safe and ensures progress towards the desired location. This chapter describes the purpose of each major component used in this decision-making process, provides insight on the data flow, and exemplifies the maneuver and constraint selection with three specifications. For ease of understanding, perception noise will only be introduced in Chapter 5.

As depicted in Fig. 3.1, the behaviour planner takes the environment representation, usually provided by perception and prior map modules, and produces a high-level driving maneuver along with a set of constraints restricting its execution in the environment. The maneuver and the constraints are then passed to a local planner, which generates a derivable trajectory to be executed by the vehicle's controller. The environment representation consists of a set of discrete and continuous features, which include: a predefined environment map, a set of all road users in the environment, and the egocentric localization. Throughout this thesis, we will name the autonomous vehicle "Ego" and thus, an egocentric localization shall be interpreted as a coordinate system using a reference point attached to the autonomous vehicle as the origin (specifically, we use the so-called *base-link frame* with the origin being the centre of the rear axle). This mostly continuous world representation is then abstracted into a set of attributes that can be processed by an expert system, the Rule-Engine, which decides the behaviour to undertake. Afterwards, this behaviour, constituted by a maneuver and a set of constraints, is refined and converted into a specification of a continuous constrained maneuver, which is then submitted to the local planner for implementation.

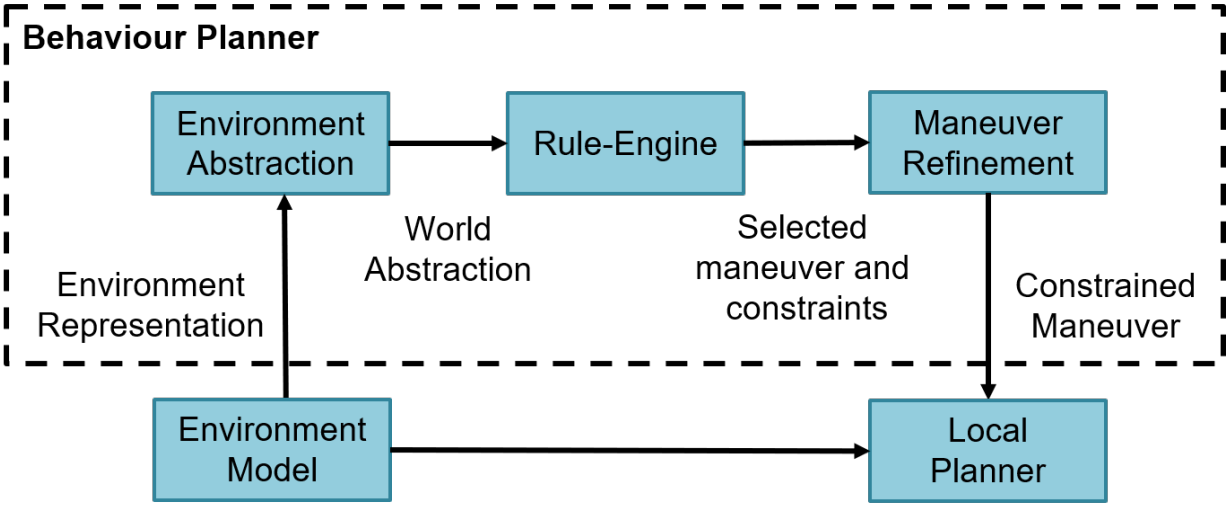


Figure 3.1: High-level architecture diagram.

3.1 Environment Abstraction

The environment representation that the behaviour planner receives is mostly continuous and also contains extraneous information. To facilitate the decision making of the Rule-Engine, the representation is filtered and abstracted into a minimal relevant *world abstraction*. This new model is represented by abstracting the information into discrete attributes that accurately describe the environment at a higher level. The attributes are divided into the following types depending on the information source.

Ego's Attributes

The information about the state of the ego vehicle is abstracted to a few relevant attributes, avoiding continuous values as much as possible except for a few attributes such as the current speed. Ego's location on the map is specified using keywords that relate it to relevant features on the map, such as 'approaching', 'at' and 'on', followed by an abstracted location such as 'intersection', 'drive-lane' and 'off-road'. In addition to these, the ego intent to take turns is described using keywords such as 'left', 'right' and 'straight'.

Travel Attributes

Travel attributes are features abstracted from the map pertaining to the current route of the ego vehicle. They include the road's speed limit and details about any relevant features on the map that ego vehicle will encounter. For example, if there is an intersection on ego's path, a set of features are required that specify whether there is a stop sign or traffic lights, along with the state of the traffic lights.

Dynamic Objects' Attributes

Other road users in the environment are referred to as dynamic objects. Like for ego attributes, real-valued attributes are mostly avoided except for their speed and their relative position with respect to the ego vehicle. Moreover, objects identified as vehicles are tagged with additional information such as Boolean values indicating the parked state or the leading role with respect to ego. Furthermore, objects identified as pedestrians use an extended set of discrete localization values such as 'left-along', 'left-towards', 'right-along', and 'right-towards', since their motion is less constrained then vehicles. Also, they are tagged with additional information, such as a Boolean value indicating the standing state.

Deciding the behaviour that an autonomous vehicle must undertake requires an accurate model predicting the dynamic object's intent, location and motion. While intent and motion depend on the sequence of temporal behaviours, the location can be represented in two ways: (i) For most encounters, the road geometry can be neglected such that the location is only described along the ego vehicle trajectory. In this representation, lateral position describes the number of lanes between a dynamic object and the ego vehicle, while the longitudinal position describes the discrete relative position of the vehicle along ego's heading such as 'behind', 'centre' and 'in front'. For any off-road dynamic object or dynamic object driving in a lane that is not parallel to ego trajectory (and thus is not part of the same road that ego travels on), both lateral and longitudinal position is set to null. (ii) The first representation is insufficient when the autonomous vehicle might get close or cross the path of another vehicle, such as near an intersection. In such an event, an alternative location specification is used where each dynamic object has a lateral and longitudinal coordinate using the ego vehicle rear axle as the origin. This representation gives a concrete notion of distance which helps to identify collision risks. In the world abstraction model, the dynamic objects are described using both representations and it is left to the Rule-Engine to make appropriate use of them.

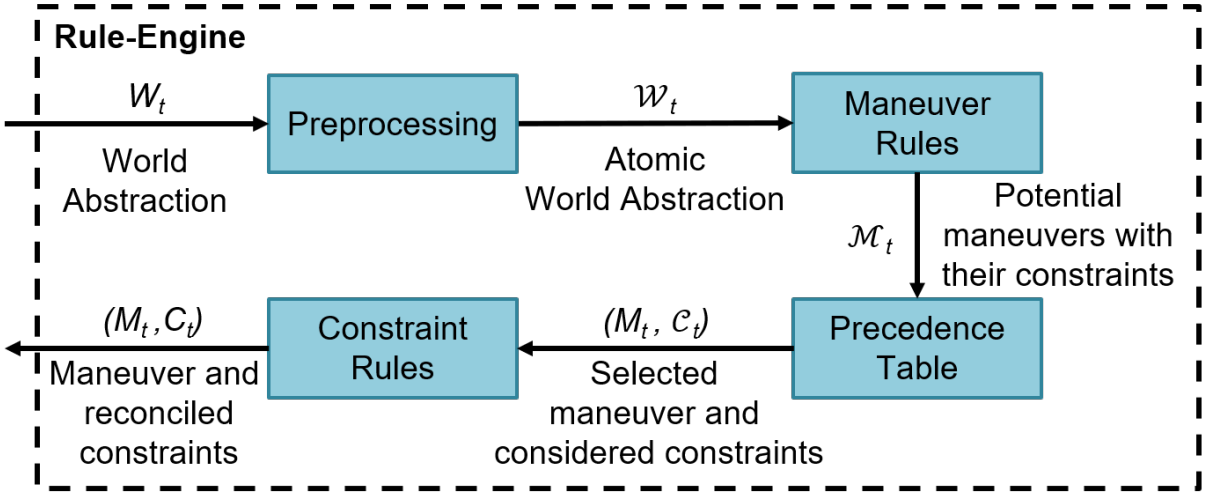


Figure 3.2: High-level architecture of the Rule-Engine.

3.2 Rule-Engine

The Rule-Engine is an implementation of an expert system using two sets of unordered rules:

i. The maneuver rules, each describing a situation that the autonomous vehicle may encounter in the environment. When this situation matches the current world state, the maneuver rule suggests a behaviour composed by a maneuver and a set of constraints restricting its execution in the environment.

ii. The constraint rules, each describing how to reconcile the behaviours produced by the maneuver rules. This set of rules receives in input the constraints of each behaviour containing the maneuver chosen by the precedence table. The rules entail the logic to filter inconsistency in the set of constraints. The constraints retained by this set of rule along with the maneuver selected by the precedence table constitute the final behaviour that ego must initiate during the next time step.

For instance, suppose that ego is driving on a straight line regulated at 50 km/h, that it approaches a crosswalk, and that no dynamic objects are perceived. A maneuver rule might consider only the speed limit and suggest to "track-speed" at "50 km/h". Another maneuver rule might consider the crosswalk and decide to lower the speed by 40%, suggesting to "track-speed" at "30 km/h". If we assume that only those two rules

produced a behaviour, than obviously, the precedence table will select "track-speed" as the maneuver to perform. However, ego cannot drive simultaneously at 50 km/h and 30 km/h. Therefore, a constraint rule might simply tell "Amongst all the retained speed constraints, select the smallest speed". In that case, the constraint rule reconciles the input speeds to 30 km/h, and the final behaviour is "track-speed" at "30 km/h".

At a high level, the Rule-Engine data flow can be decomposed in four stages as depicted in Fig. 3.2. At each time step, t, the Rule-Engine will go over the whole process before committing on the behaviour to initiate.

The preprocessing stage aims to convert the mostly discrete input model into a representation that is more suited for searching purposes. Namely, in the preprocessing stage, the Rule-Engine receives the mostly discrete world abstraction model, W_t, and converts it into a world atomic abstraction model represented using nullable-Boolean attributes, $\mathcal{W}_t$, where discrete attributes have one of the three values: true, false, or null. The null is used to represent the absence of a meaningful Boolean value (because the attribute is not applicable in a given situation—such as dimensions of the next intersection, when there is none) or the Boolean value being unknown (such as the Rule-Engine assuming the validity of an attribute in the given situation, but not having yet received a reliable value—such as the heading of an already detected dynamic object, estimation of which requires longer observation time). The maneuver and constraint rules can check for true, false, and null explicitly, and, in particular, react to the absence of a meaningful value (denoted by null) accordingly.

During the preprocessing, the system reduces each attribute into a set of atomic propositions, refines the model by deriving additional atomic propositions, and carries the temporal proposals of the previous and predicted steps. As an example, the abstract location attribute named 'approaching' could be arbitrarily discretized into the following set of atomic propositions {'approachingIntersection', 'approachingDriveLane', 'approachingOffRoad'}, while a carried temporal attribute could be 'wasRegulatedByAStopSign' that extends the time-free attribute 'isRegulatedStop'. When Ego is in the physical area of the intersection, 'isRegulatedStop' describes the next intersection along Ego's trajectory, while 'wasRegulatedByAStopSign' still describes the road geometry of the current intersection allowing, for instance, the system to disable the constraint rules producing the left turn across path constraint.

The maneuver rules stage evaluates the first set of rules. Each maneuver rule decides a maneuver in the following complete set, $M := \{track\text{-}speed, follow\text{-}leader, overtake, decelerate\text{-}to\text{-}halt, yield, stop, emergency\text{-}stop\}$. When a rule is evaluated, it assesses if the atomic world abstraction, $\mathcal{W}_t$, matches the encapsulated situation. If it does, the rule

outputs a maneuver restricted by a set of constraints. The collection of outputs, $R(\mathcal{W}_t) \vdash \mathcal{M}_t$, where R represents the set of maneuver rules, describes the potential behaviours for the autonomous vehicle during the next time step, t.

To decide which maneuver should be performed, the set of potential behaviours, $\mathcal{M}_t$, is then prioritized based on a precedence table. The precedence is given by the totally ordered maneuver set, M, where *track-speed* has the lowest priority. Amongst the maneuvers in $\mathcal{M}_t$, the one with the highest priority, denoted as M_t, is retained. Thereafter, each behaviour containing this maneuver is converted into a nullable-Boolean representation that we refer to as the considered constraint model, $\mathcal{C}_t$.

Finally, the constraint rules stage corresponds to the evaluation of the second set of rules. Its main purpose is to analyze the dynamic/static objects conditioning the behaviour of the autonomous vehicle during the next time step, t. The output of this set of rules, $R(\mathcal{C}_t) \vdash C_t$, constitutes the restrictions that the autonomous vehicle must enforce when performing the selected maneuver, M_t. The outputted constraints are then paired with the selected maneuver to describe the behaviour to execute, (M_t, C_t).

3.3 Example Scenarios

To better understand the proposed architecture, we will use a list of scenarios that an autonomous vehicle may encounter when travelling in an urban environment. We will define a scenario as a sequence of behaviours that allows the autonomous vehicle to travel from an initial point to a target location. By convention, we will indicate Ego's initial position as the starting point of the scenario and represent the target location by a green target. Throughout this thesis, the scenarios presented will be discussed thoroughly and will serve to depict the data flow.

Figure 3.3 presents the first scenario where Ego is on a two-way road and must navigate to the target location. In this scenario, this road stretch is limited to 50 km/h and no dynamic object will interfere, allowing the autonomous vehicle to drive freely. Therefore, Ego will have to repeatedly execute the "track-speed" maneuver until the target location is reached. In addition, it will also have to consider the speed limit as a constraint to establish its cruising speed.

Figure 3.4 shows the second scenario in which Ego is stopped behind a parked vehicle, V_1. It will seek to pass this vehicle as well as the next parked vehicle, V_3, while keeping a safety gap to the lead vehicle, V_2. This scenario has a road structure like the previous one, since it is also a two-way road with a speed limit of 50 km/h.

Figure 3.5 presents the third scenario in which Ego is approaching an intersection regulated by a stop sign in all directions. The autonomous vehicle will have to anticipate that the leading vehicle, V_1, will want to stop at the stop line. Thus, it must stop behind the leader and must consider the future position of the pedestrian, P_1, before entering the intersection, regardless of the turn made by the leading vehicle. From a geometric point of view, this junction consists of a pair of two-way roads each with a speed limit of 50 km/h.

The scenarios presented in this chapter demonstrate the need for a behaviour planner within an autonomous vehicle since the situations encountered by Ego influence the maneuver to initiate. The temporal nature of the behavioural planning problem suggests that a good planner must both consider the constraints of the present state, but also anticipate the result of the interactions among the dynamic objects. We will see how to establish these foundations in the next two chapters.

In this chapter, we have classified the properties of the environment into three categories: ego, travel, and the dynamic objects. We have also briefly introduced the four main components of our expert system: the preprocessor, the maneuver rules, the precedence table, as well as the constraint resolution rules. Finally, we illustrated three scenarios on which we will later demonstrate the properties of our system.

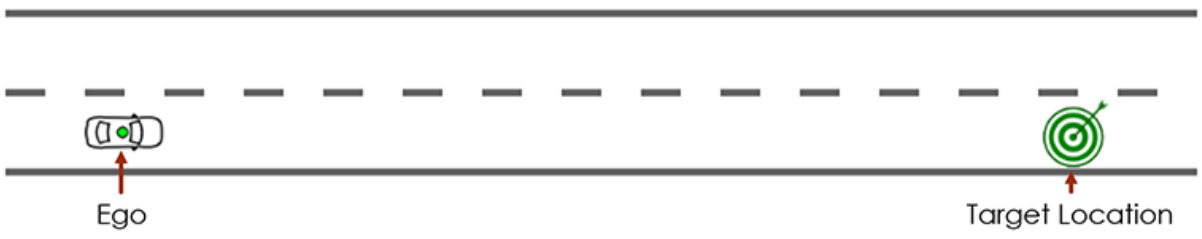


Figure 3.3: [Scenario 1] Ego navigating freely on a two-way road.

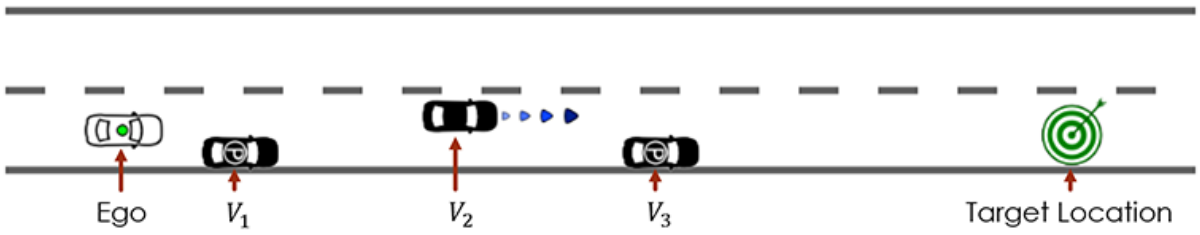


Figure 3.4: [Scenario 2] Ego passing a sequence of parked vehicles.

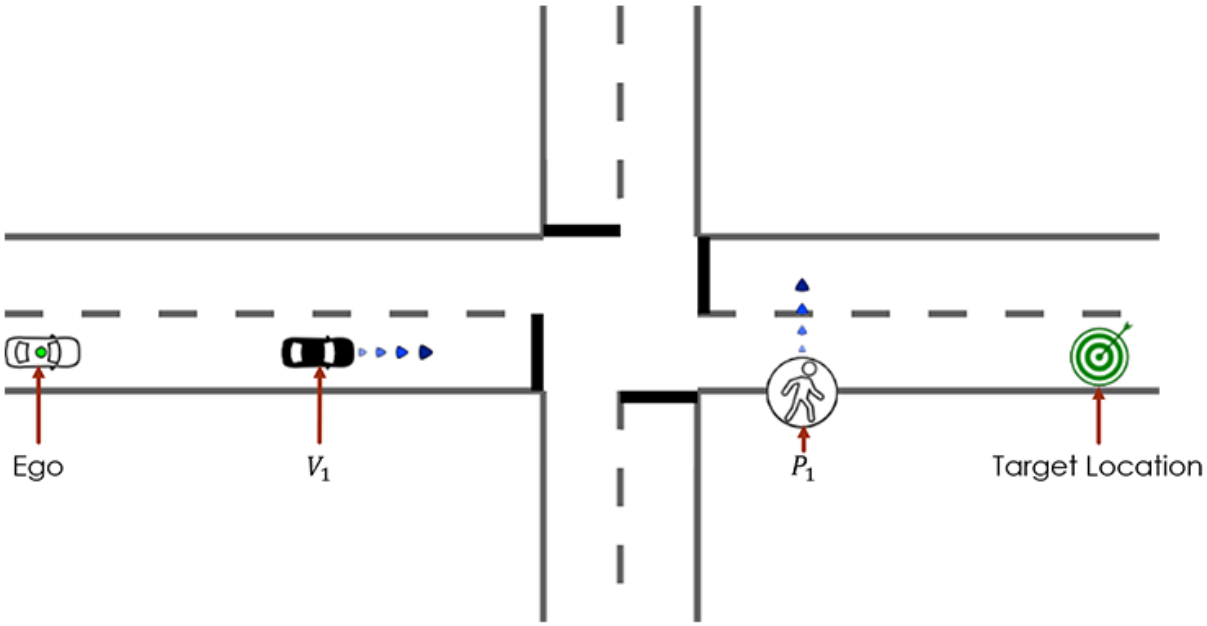


Figure 3.5: [Scenario 3] Ego crossing an intersection regulated with an all-way stop.

Chapter 4

Mid-Level Architecture

In this chapter, we illustrate the Rule-Engine's decision flow using the first scenario where Ego drives freely on a two-way road (Fig. 3.3). We review the four main stages of the expert system, depicted in light blue in Fig. 4.1, by detailing their inputs and outputs represented in light green. To facilitate the understanding of the methodology, the data flow studied is simplified and only highlights the attribute subset that is relevant to the decision-making process. We invite, however, the reader to consult the book of rules appendices B to discover concrete examples.

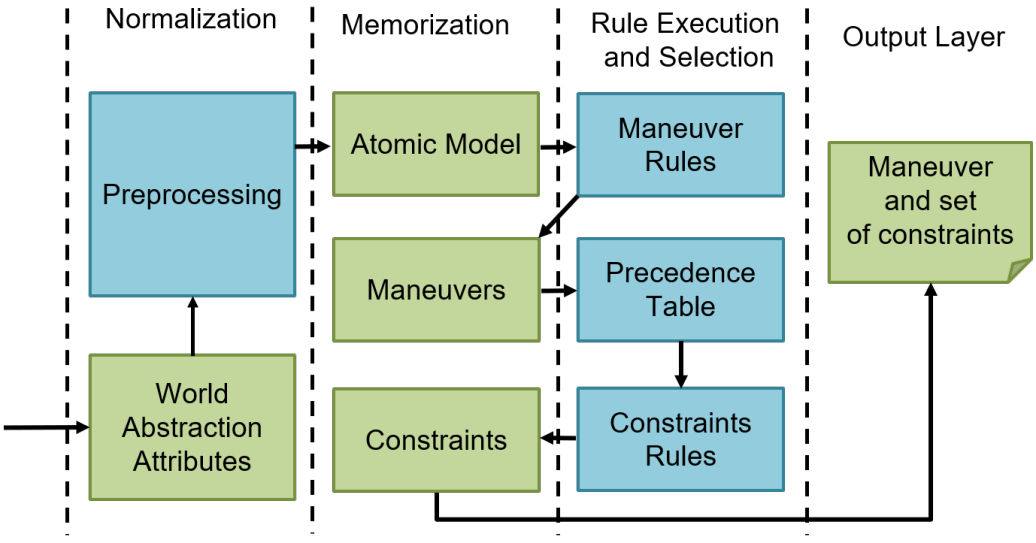


Figure 4.1: Mid-level architecture of the Rule-Engine.

4.1 World Abstraction Model

In Chapter 3.1, we saw that the Rule-Engine receives a data model decomposed into three categories of attributes: Ego, Travel, and Dynamic Objects that we further divided into Pedestrians and Vehicles. Considering the specificities of the first scenario, we can omit the attributes of Pedestrians as well as Vehicles since they will not intervene during the scenario. We will represent this absence of dynamic objects by using the empty list. With that in mind, we will only describe Ego and Travel attributes.

Ego's Attributes

The following list describes the Ego attributes for the inputted model to the Rule-Engine. This list is also synthesized in Listing 1 where each attribute enumerates the set of discrete or continuous values to which it can be valuated. In Listing 1, we use || to denote the logical or.

Listing 1 Ego's Attributes

```
{
    "location": {
      "approaching": "crosswalk" || "intersection" || "",
      "at": "crosswalk" || "intersection" || "",
      "on": "crosswalk" || "drive-lane" || "intersection" ||
            "off-road" || "",
      "timeOfArrival": A_REAL_NUMBER
    },
    "navigation": "left" || "right" || "straight" || "",
    "speed": A_REAL_NUMBER
}
```

Approaching: An enumeration indicating that the ego vehicle is approaching a critical point. For instance, to approach an intersection means that Ego is between 0 and x metres, excluding the bounds, from the stop line, without having passed it, where x is a parameter.

At: An enumeration indicating that the ego vehicle is currently at a critical point. For instance, to be at an intersection means that Ego is about to enter the intersection which can be at a stop line or the end of the lane.

Navigation: An enumeration that gives information to the Rule-Engine about the Ego's next goal such as turning left or right or going straight.

On: An enumeration that provides information on where the ego vehicle is travelling. For example, to be on an intersection means that Ego is beyond the stop line and therefore in the physical area of the intersection.

Speed: A real number that represents the speed of Ego in kilometres per hour.

Time of arrival: A real number representing the estimated time in seconds since Ego has arrived at a specific abstract location. This property will be used to determine which road users have the right of way.

Travel Attributes

The following list describes the travel attributes for the inputted model to the Rule-Engine. This list is also synthesized in Listing 2 where each attribute enumerates the set of discrete or continuous values to which it can be valuated. In Listing 2, we use || to denote the logical or.

Distance: A distance in metres to some location or object of interest considering the road geometry, e.g., along the lane centre line.

Intersection Center X: The actual distance, in metres using base-link frame, between Ego and the width centre of the intersection. In Figure 4.2, this attribute is depicted by the longitudinal distance of the turquoise arrow.

Intersection Center Y: The actual distance, in metres using base-link frame, between Ego and the height centre of the intersection. In Figure 4.2, this attribute is depicted by the lateral distance of the turquoise arrow.

Intersection Distance: The actual distance, in metres, that Ego will have to travel to get to the location atIntersection of the nearest intersection. In Figure 4.2, this attribute is depicted by the orange arrow.

Intersection Length: The length of the intersection is the longitudinal distance, in metres, from the end of the location atIntersection to the end of the location onIntersection assuming that Ego would navigate straight. In Figure 4.2, this attribute is depicted by the green arrow.

Intersection Width: The width of the intersection is the lateral distance, in metres, from the end of the location atIntersection to the end of the location onIntersection along the crossing road from left to right. In Figure 4.2, this attribute is depicted by the red arrow.

Left Traffic Light: The left light corresponds to the signal dedicated to left turns. For example, the left turn priority.

Main Traffic Light: The main light represents the general priority level of the traffic light if the left and right lights are omitted, otherwise it represents the priority for vehicles crossing in a straight line.

Regulation: A traffic control device indicating whom to prioritize at a junction and what behaviour is expected.

Right Traffic Light: The right light corresponds to the signal dedicated to right turns. For example, the right turn priority.

Special Traffic Light: Represents an additional indicator for special vehicles such as an ambulance, a bus, a fire truck and a taxi.

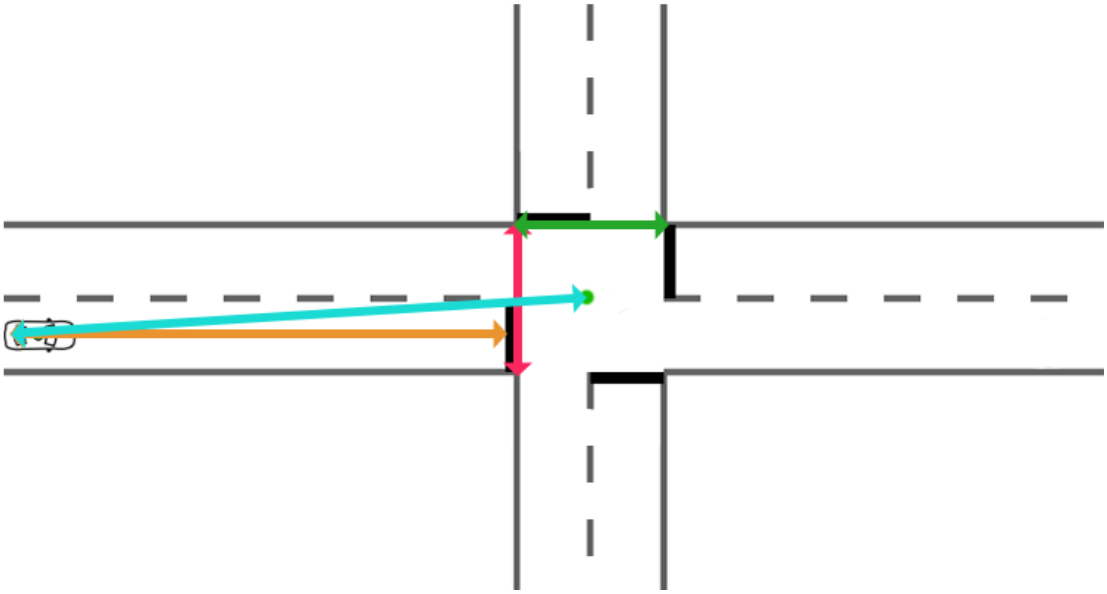


Figure 4.2: Travel attributes for an intersection.

Speed Limit: A real number that represents the speed limit in kilometres per hour that Ego would like to follow and should not exceed. Among others, it will be used to determine if we want to overtake a vehicle.

Time till clear: A set of real numbers representing the estimated time in milliseconds where the crosswalk will be freed of any pedestrian.

Time till pedestrian enters: A set of real numbers representing the estimated time in milliseconds where at least one pedestrian will be on the crosswalk region.

Listing 2 Travel Attributes

```
{
    "crosswalk": {
        "distance": REAL_NUMBER,
        "timeTillClear": [REAL_NUMBER, ..., REAL_NUMBER],
        "timeTillPedestrianEnters": [REAL_NUMBER, ..., REAL_NUMBER]
    },
    "intersection": {
        "center": {
            "x": A_REAL_NUMBER,
            "y": A_REAL_NUMBER
        },
        "distance": A_REAL_NUMBER,
        "length": A_REAL_NUMBER,
        "width": A_REAL_NUMBER,
    },
    "regulation": "stop" || "yield" || "",
    "speedLimit": A_NATURAL_NUMBER,
    "trafficLight": {
        "left": "red" || "yellow" || "green" || ""
        "main": "red" || "yellow" || "green" || "green-blink" ||
                "yellow-blink" || "red-blink" || "",
        "right": "red" || "yellow" || "green" || "",
        "special": "bus-cab" || "emergency" || ""
    }
}
```

The attributes that we have listed correspond to the operational-design domain and the perception capacities of our autonomous vehicle. The model remains rather simple and hides advanced concepts of the other modules such as perception, tracking, local planning etc. Such a model allows newcomers to quickly grasp the behavioural planning concepts required to operate in urban areas. However, it is expected that the set of attributes can be improved by more accurate representations such as the use of lanelets as well as by new actions such as overtaking dynamic objects and changing lanes.

Model Integrity

To enforce a proper usage of the World Abstraction model presented in this section, the knowledge expert can add a special type of rule aimed solely at validating the input data integrity. This set of rules can, for instance, introduce a series of *emergency-stop* maneuvers preventing the autonomous vehicle to initiate any alternative maneuver when a problem is detected during the stage of data ingestion. For example, these rules could verify that the speed of dynamic objects cannot be negative or that, when a group of attributes require to have meaningful values such as when Ego is navigating nearby an intersection, the following attributes must also be declared: intersection height and width, the x and y coordinates of the central point of the intersection, etc. Therefore, this superfluous set of rules facilitate the integration of the Rule-Engine with the other components of the autonomous stack. When this development stage is achieved, such as when the usage of the world abstraction model is aligned with the semantics of its ontology, this set of rules can be safely removed without causing disparities in the sequence of behaviours undertaken by the autonomous vehicle. Furthermore, in Chapter 8 when we will have studied the misbehaviour diagnosis capability of the Rule-Engine, we will see that the model integrity rules also provide improved diagnosis and promote independence between teams, because component teams can benefit from analysis and debugging capabilities without solely relying on the knowledge expert.

Scenario 1: Model Instance

Now that we have defined the attributes of the data transfer object (DTO), we can look at an example of model instantiation. Listing 3 represents the values, W_t, submitted as input to the Rule-Engine when Ego is at its initial location.

Listing 3 [Scenario 1] Model Instance

```
{
    "ego": {
        "location": {
          "approaching": "",
          "at": "",
          "on": "drive-lane",
          "timeOfArrival": NULL
        },
        "navigation": "",
        "speed": 0
    },
    "pedestrians": [],
    "travel": {
        "crosswalk": NULL,
        "intersection":NULL,
        "regulation": "",
        "speedLimit": 50,
        "trafficLight": NULL
    },
    "vehicles": []
}
```

4.2 Preprocessing

At every time step, t, the world abstraction, W_t, is converted into an atomic world abstraction, $\mathcal{W}_t$, by a preprocessing phase. This process begins by discretizing the natural/real values of W_t into a series of ternaries (nullable Boolean) using thresholding. A thresholding function returns null when the value of the attribute is undefined, or either true or false depending on whether the value of the attribute matches a test clause or not. Each threshold comes from empirical testing either in a simulated environment or a closed-course environment.

Similarly, the discrete attributes are also discretized to ternaries. To do so, each value in the enumeration type of an attribute of that type is converted to a nullable Boolean

attribute. Again, when the value of an attribute is undefined, the ternary is set to null and when the value is known, then the ternary is either true or false.

When every attribute is Boolean, the preprocessor combines several attributes creating new derived propositions. This addition of derived attributes simplifies the management of the rule base as well as reduces the overall rule activation frequency as described later in Chapter 8.

Listing 4 [Scenario 1] Atomic World Abstraction


```
{
    "ego": {
        "exceedSpeedLimit": false,
        "hasLowSpeed": false,
        "hasPrecedence": false,
        "hasSpeed": false,
        "location": {
          "approachingCrosswalk": false,
          "approachingIntersection": false,
          "atCrosswalk": false,
          "atIntersection": false,
          "onCrosswalk": false,
          "onDriveLane": true,
          "onIntersection": false,
          "onOffRoad": false
        },
        "navigationLeft": false,
        "navigationRight": false,
        "navigationStraight": false
    },
    "travel": {
        "regulationStop": false,
        "regulationYield": false
    }
}
```

For example, if we take the attribute *location.approaching* from the Ego category, each value recognized by this enumeration becomes a Boolean, such as *location.approachingCrosswalk* and *location.approachingIntersection*. For numeric values such as *speed* of the Ego category, several Booleans will be extracted according to the rules' needs. For example, we extract *hasSpeed* (i.e., speed is nonzero) and *hasLowSpeed* (i.e., speed is below a threshold). Finally, some propositions are derived, such as *exceedSpeedLimit* of the Ego's atomic model. This proposition is derived from the *speed* in the Ego category and *speedLimit* from the Travel category.

At the end of this conversion process, the generated atomic world abstraction includes a subset of the atomic world abstraction model propositions, $\mathcal{W}_t \subseteq \mathcal{W}$. Let's apply this process to the world abstraction, W_t, of our first scenario. We will obtain the Boolean variable assignment depicted by Listing 4.

The preprocessing stage also includes several other discretization mechanisms that increase noise robustness and allow the rules to manage the dynamic objects more efficiently. However, they are not needed to understand the studied scenario and are therefore omitted until Chapter 5.

4.3 Maneuver Rules Evaluation

One of the design goals for the Rule-Engine is to reduce its workload such that it can replan behaviours at high frequency. To achieve this, $\mathcal{W}_t$ and $\mathcal{W}_{t-1}$ are assumed to be highly correlated such that, if the history of atomic world abstractions constitutes a sequence of logical observations, the outputs of many rules at time t should be identical to those at time $t-1$.

Since each rule, R_i, is expressed with clauses and consequences over a subset $\mathcal{W}_i$ of the propositions that constitute the atomic model, namely $\mathcal{W}_i \subseteq \mathcal{W}$, then the system only needs to evaluate a rule if $(\mathcal{W}_t - \mathcal{W}_{t-1}) \cap \mathcal{W}_i \neq \varnothing$. Here, the difference between $\mathcal{W}_t$ and $\mathcal{W}_{t-1}$ results in key-value pairs that change between these two time steps. This monitoring, which we refer to as the maneuver rule activation function, allows the Rule-Engine to retain in memory the output of a rule when its proposition has not changed.

We denote the evaluation of a maneuver rule on the atomic world abstraction, $\mathcal{W}_t$, as $R_i(\mathcal{W}_t) \vdash \mathcal{M}_{i,t}$. The proposed constrained maneuver $\mathcal{M}_{i,t}$ represents a behaviour that the autonomous vehicle may adopt, partially adopt, or reject. The agglomeration of the output of each rule at time step t, become the set of potential behaviours $\mathcal{M}_t$, which the ego vehicle can initiate.

Formally, the rule clauses are specified in the disjunctive normal form. Thus, let A, B, and C be the literals required for the clauses of a rule, and let D be the consequence of the rule. Any of the following combination of conjunctions and disjunctions are well-constructed rules: $(A \wedge B \wedge C) \vdash D$, $(A \vee B \wedge C) \vdash D$, $(A \wedge B \vee C) \vdash D$, or $(A \vee B \vee C) \vdash D$. When one of the sequences of conjunctions returns true, the rule can deduce the constraint D from the literals and may also use any values from the world abstraction in D.

More concretely, if we return to our scenario, we could model the following rule: When Ego is on a two-lane road then it must *track-speed* to the speed limit regulated on this stretch. In this example, the propositions used for the clauses of the rule are (A) *location.onDriveLane*, (B) *location.approachingCrosswalk*, (C) *location.approachingIntersection*, (D) *location.atCrosswalk*, (E) *location.atIntersection*, all from the atomic Ego category. The value from the world abstraction (F) *speedLimit* of the Travel category is used as a constraint. This rule can be written as $(A \wedge \neg B \wedge \neg C \wedge \neg D \wedge \neg E) \vdash (\textit{track-speed}, F)$. By judiciously memorizing the inputs and outputs of the preprocessing, the activation function will only re-evaluate the rule if one of the ternary A, B, C, D, or E changes or if the discrete value, F, changes. Thus, any other atomic proposition changes will not influence the decision of the behaviour planner and may therefore be neglected.

Scenarios 2 and 3 are examples where the rules will use the clauses to target the dynamic objects constraining the maneuver. Therefore, we will omit these details until Chapter 5.

4.4 Precedence Table

A maneuver M_t is chosen from $\mathcal{M}_t$ by following the totally ordered set of the precedence table, $M := \{\textit{track-speed}, \textit{follow-leader}, \textit{overtake}, \textit{decelerate-to-halt}, \textit{yield}, \textit{stop}, \textit{emergency-stop}\}$, where *track-speed* has the lowest priority. This ordering is based on conservatism to ensure safety and ease of rule creation. For instance, the conservatism prioritizes deceleration and standstill maneuver since collision at low velocity are usually less lethal.

This conservatism is extended such that if there is no achievable maneuver, that is, $\mathcal{M}_t$ is the empty set, M_t falls back to *emergency-stop*. In our system, *emergency-stop* is the safest maneuver since it emits an audible signal telling the safety driver to take over. In addition, the fallback formulation allows the knowledge expert to only focus on the rule that allows safe progress rather than exhaustively listing the risky situations. This significantly reduces the number of required rules and increases the trustworthiness in the system since the knowledge expert can control the fallback behaviours when Ego encounters

an unhandled case. For example, when the ego vehicle navigates on a road segment with high collision risk, such as an intersection, the knowledge expert can produce rules based on pessimism by only describing the conditions allowing Ego to enter the intersection. Otherwise, the rules can also be modelled optimistically, such as the track-speed rule we have created previously for two-way roads. In that optimistic case, since the track-speed rule acts greedily, the remaining cases to describe are the events that constrain Ego's driving.

For now, consider the track-speed rule that we created. This rule does not have a disjunction, which implies that all the conjunctions must be satisfied to get a non-empty output. Note that the atomic world abstraction, $\mathcal{W}_t$, we obtained in Chapter 4.2 makes the clause true and therefore the rule produces the *track-speed* maneuver and the 50 km/h constraint. Since this is the only rule declared so far, we can assume that this is the only option available in the system and therefore it takes precedence. By applying the aforementioned principle of greediness, we can anticipate that this rule will fire as long as Ego does not reach *location.approachingCrosswalk*, *location.approachingIntersection*, or *location.onOffRoad*. If any of these locations were reached, then the system would apply *emergency-stop* since no other rule is present and the proposed maneuver set would be empty.

Even if this example is simplistic, it provides an insight on how the rules work. By the greedy nature of the rule created, we can anticipate that this rule also applies to scenario 2 where Ego is on a two-lane road and must pass two parked vehicles by maintaining a safe distance from the leading vehicle. However, in this scenario, a rule suggesting the overtake maneuver constrained by the two vehicles to overtake as well as the leading vehicle will have to be modelled. Thereafter, the system will have the choice between *track-speed*, *follow-leader* and *overtake* and given the precedence table, M, we can anticipate that the *overtake* maneuver will be chosen.

Once a maneuver M_t is chosen, the constraints produced by the maneuver rules that yield the tuple $(M_t, \_)$, where $\_$ denotes any value, are merged together and preprocessed into an atomic considered constraint set, $\mathcal{C}_t$. This second preprocessing stage follows the same idea as the preprocessing of the world abstraction presented in this chapter. In the next chapter, we will briefly exemplify the constraints preprocessing when analyzing a scenario with conflicting opportunities. In the simple scenario under study in this chapter, since we only have one rule modelled and since that rule produces the intended *track-speed* maneuver, there is nothing to merge and this preprocessing is not needed.

4.5 Constraint Rules Evaluation

At this stage, the Rule-Engine knows the maneuver to undertake and now seeks to choose the most appropriate, reconciled restrictions in the considered constraints model, $\mathcal{C}_t$. When there is only one candidate constraint, the choice is trivial since it suffices to apply it. Nevertheless, when the system has several conflicting options, it will seek to reconcile constraints by applying a set of conservative rules.

For example, the rules covering the *track-speed* maneuver will seek to establish the lowest speed limit among those suggested. The *follow-leader* maneuver will seek to establish the lowest speed limit proposed, but at the same time will require to establish a safe following distance to the dynamic object. The *overtake* maneuver will look for one to several vehicles to overtake, the existence of a leading vehicle as well as the lowest speed limit suggested. The *decelerate-to-halt* maneuver, for its part, will want to identify the closest deceleration target such as a stop line, an end of the lane, a left turn across the path, or a dynamic object. Finally, the *yield*, *stop*, and *emergency-stop* maneuvers do not require any particular constraints, but still attempt to identify the causality of this stop, such as the dynamic object to yield to.

In the sample scenario, we are facing the trivial case where the system has only one constraint available, namely, the speed limit of 50 km/h. Thus, since there is nothing to filter, the system will apply it. However, we could imagine the case where the ego vehicle is approaching a school zone limited to 30 km/h with dedicated rule for the speed limit in school zones. Therefore, both rules will suggest the *track-speed* maneuver and the system now has to choose between 30 km/h or 50 km/h. By conservative modelling, the vehicle would begin to regulate its speed limit to reach the 30 km/h.

This chapter has illustrated the decision-making process of the Rule-Engine in a scenario that does not involve any dynamic object. We started our journey by listing the attributes of the Ego and Travel categories that were essential to understanding the scenario. We then motivated the conversion of the data transfer object into the data search object in the preprocessing stage. Subsequently, we saw how to model a rule and illustrated the output of the maneuver rules. We also discussed the first level of reconciliation of our inferential process, namely the precedence table. This table allows mainly to select the maneuver to execute from a list that may be inconsistent. This process ended with the reconciliation of the constraints embedded in the behaviours including the chosen maneuver. Those constraint were derived by the maneuver rules, but need to be reconciled by the constraint rules. This reconciliation produces a consistent set of constraints. Thereafter, the chosen maneuver and the reconciled set of constraints are merged defining the behaviour to execute during the next time step.

Chapter 5

Low-Level Architecture

Now that we know the basics of the Rule-Engine's decision-making process, we are ready to evaluate a more complex example where the system must deal with noisy data and partial observability. To do this, we will study the third scenario depicted in Figure 3.5 where Ego wants to cross an intersection regulated by an all-way stop while a leading vehicle navigates in front of it and a pedestrian crosses concurrently at the intersection.

As in the previous chapter, we will review the full data flow by observing the system's inputs and outputs in the four main phases, namely the preprocessing, the maneuver rules evaluation, the precedence table, and the constraint rules evaluation. However, this example will require a deeper understanding of the concepts presented since the decision-making will have several conflicts to reconcile.

5.1 Dynamic Objects Abstraction

In Chapter 4.1, we saw the Ego and Travel attribute categories of the input model received by the Rule-Engine. In this section, we describe the dynamic object attributes which can be decomposed in Pedestrians (Listing 5) and Vehicles (Listing 6) categories. As usual, we end the section by showing the model valuation when Ego is at the starting position of the third scenario.

Dynamic Object Attributes

Approaching: An enumeration indicating that a pedestrian/vehicle is approaching a critical point. For instance, for a pedestrian/vehicle to approach an intersection means that he/she/it is between 0 and x metres from the physical area of the intersection, where x is a parameter.

At: An enumeration indicating that a pedestrian/vehicle is currently at a critical point. For instance, to be at an intersection means that the pedestrian/vehicle is about to enter the intersection.

Distance x: According to the base-link of the ego vehicle, this distance in metres is positive when the pedestrian/vehicle is somewhere in front of Ego's rear axle.

Distance y: According to the base-link of the ego vehicle, this distance in metres is positive when the pedestrian/vehicle is somewhere to the left of the center of Ego's rear axle.

Heading (orientation): This attribute represents the direction of the pedestrian's/vehicle's velocity vector relative to Ego's path. This direction is used to determine potential collision risk, while considering the road geometry. The discrete values of this attribute are illustrated in Figure 5.2 (right). In Figure 5.1, this heading is along-ego since the velocity vector of each vehicle is following the same path.

Heading (relative orientation): This attribute represents the direction of the pedestrian's/vehicle's velocity vector relative to Ego's orientation. This direction is used to

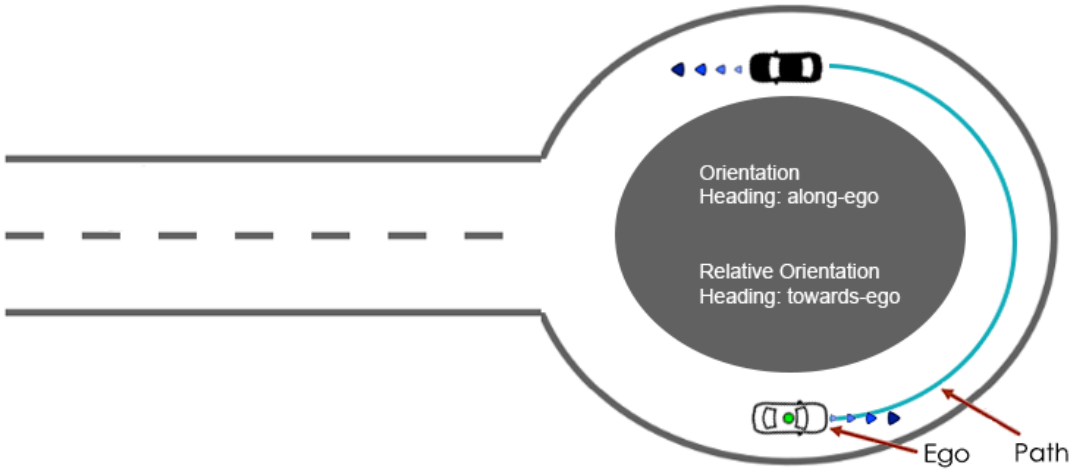


Figure 5.1: Definition of the two heading types.

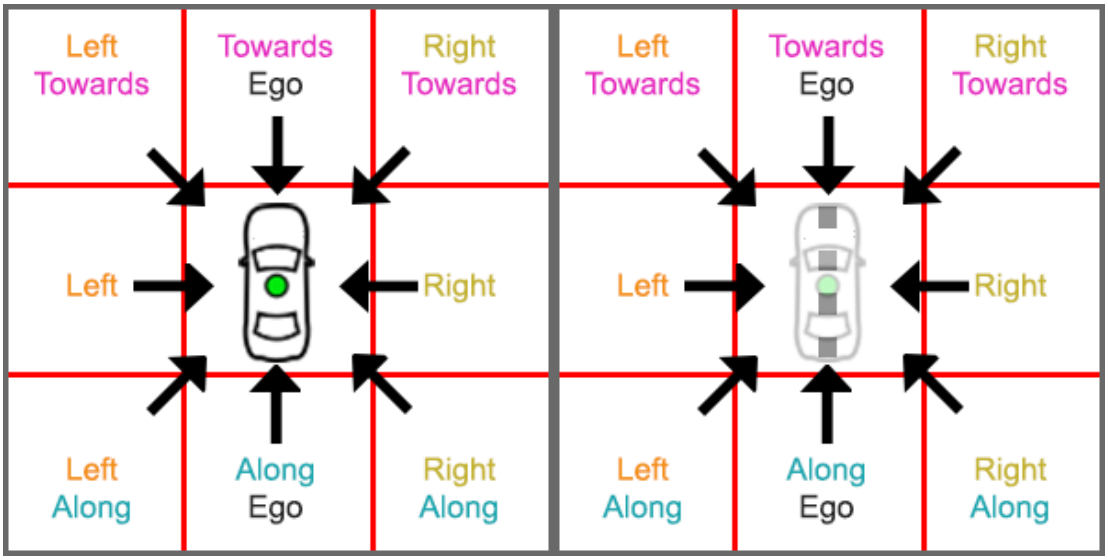


Figure 5.2: Heading (relative orientation) of a dynamic object, shown on the left, is defined by the orientation of its velocity vector (black arrow) relative to Ego's current position. Heading (orientation) of a dynamic object, shown on the right, is defined by the orientation of its velocity vector (black arrow) relative to Ego's future position being the point on Ego's future path that is closest to the dynamic object.

determine potential collision risk, independent of the road geometry. The discrete values of this attribute are illustrated in Figure 5.2 (left). In Figure 5.1, this heading is towards-ego since the velocity vector of each vehicle have opposite direction.

Id: This attribute is a unique identifier that allows maintaining pedestrian's/vehicle's identity over time.

Is leading: This Boolean attribute, when true, indicates that the pedestrian/vehicle obstructs Ego in front, but is also moving in the same direction as Ego (but it also could be temporarily stopped). This target must be the one closest to the ego vehicle. There cannot be multiple leaders.

Is obstructing: Boolean attribute that, when true, indicates that the pedestrian/vehicle is crossing the trajectory of Ego or is leading the ego vehicle. If several entities obstruct the trajectory, only the one closest to Ego is marked. There cannot be multiple obstructions.

Lateral orientation: If we divide the area around Ego into a 3x3 grid (see Figure 5.3), we can determine a relative horizontal position with respect to the column in which the pedestrian/vehicle is located, i.e., whether the pedestrian/vehicle is on Ego's path or adjacent

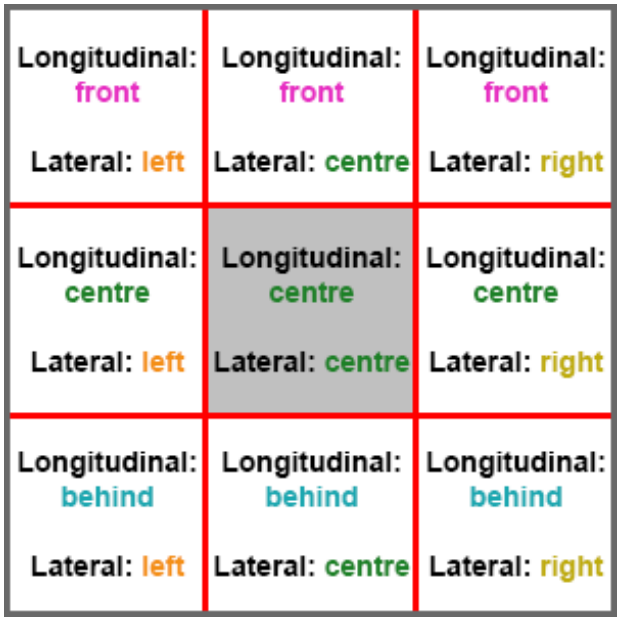


Figure 5.3: Definition of longitudinal and lateral orientation where Ego is always located in the gray square and where the distances between the ego vehicle and each of the four red lines are hyperparameters.

to it on the left or right.

Longitudinal orientation: If we divide the area around Ego into 3x3 grid (see Figure 5.3), we can determine a relative vertical position with respect to the row in which the pedestrian/vehicle is located, i.e., whether the pedestrian/vehicle is in front, next to, or behind Ego.

On: An enumeration providing information on where the pedestrian/vehicle is travelling. For example, to be on an intersection implies that the pedestrian/vehicle is on the roadway.

Speed: This real number attribute represents the pedestrian/vehicle speed in kilometres per hour.

Time of arrival: This real number attribute represents the estimated time in seconds since the pedestrian/vehicle has arrived at a specific abstract location. This property will be used to determine which road users have the right of way.

Listing 5 Pedestrian Attributes

```
[{
    "id": A_UNIQUE_NUMBER || A_UNIQUE_STRING,
    "distance": {
        "x": A_REAL_NUMBER,
        "y": A_REAL_NUMBER
    },
    "isLeading": true || false,
    "isObstructing": true || false,
    "location": {
        "approaching": "crosswalk" || "intersection" || "",
        "at": "crosswalk" || "intersection" || "",
        "on": "crosswalk" || "drive-lane" || "intersection" ||
              "off-road" || "",
        "timeOfArrival": A_REAL_NUMBER
    },
    "orientation": {
        "heading": "left" || "right" || "along-ego" ||
                   "towards-ego" || "left-towards", ||
                   "left-along" || "right-towards" || "right-along",
        "lateral": "left" || "right" || "center",
        "longitudinal": "front" || "behind" || "center"
    },
    "relativeOrientation": {
        "heading": "left" || "right" || "along-ego" ||
                   "towards-ego" || "left-towards", ||
                   "left-along" || "right-towards" || "right-along"
    },
    "speed": A_REAL_NUMBER
}]
```

Vehicle Specific Attributes

Is parked: A Boolean which represents a stationary vehicle subject to remain in this position when the traffic will move forward.

Regulation: A traffic control device indicating whom to prioritize at a junction and what behaviour is expected.

Listing 6 Vehicle Attributes

```
[{
    "id": A_UNIQUE_NUMBER || A_UNIQUE_STRING,
    "distance": {
        "x": A_REAL_NUMBER,
        "y": A_REAL_NUMBER
    },
    "isLeading": true || false,
    "isObstructing": true || false,
    "isParked": true || false,
    "location": {
        "approaching": "crosswalk" || "intersection" || "",
        "at": "crosswalk" || "intersection" || "",
        "on": "crosswalk" || "drive-lane" || "intersection" ||
              "off-road" || "",
        "timeOfArrival": A_REAL_NUMBER
    },
    "orientation": {
        "heading": "left" || "right" || "along-ego" ||
                   "towards-ego",
        "lateral": "left" || "right" || "center",
        "longitudinal": "front" || "behind" || "center"
    },
    "regulation": "stop" || "yield" || "",
    "relativeOrientation": {
        "heading": "left" || "right" || "along-ego" ||
                   "towards-ego" || "left-towards", ||
                   "left-along" || "right-towards" || "right-along"
    },
    "speed": A_REAL_NUMBER
}]
```

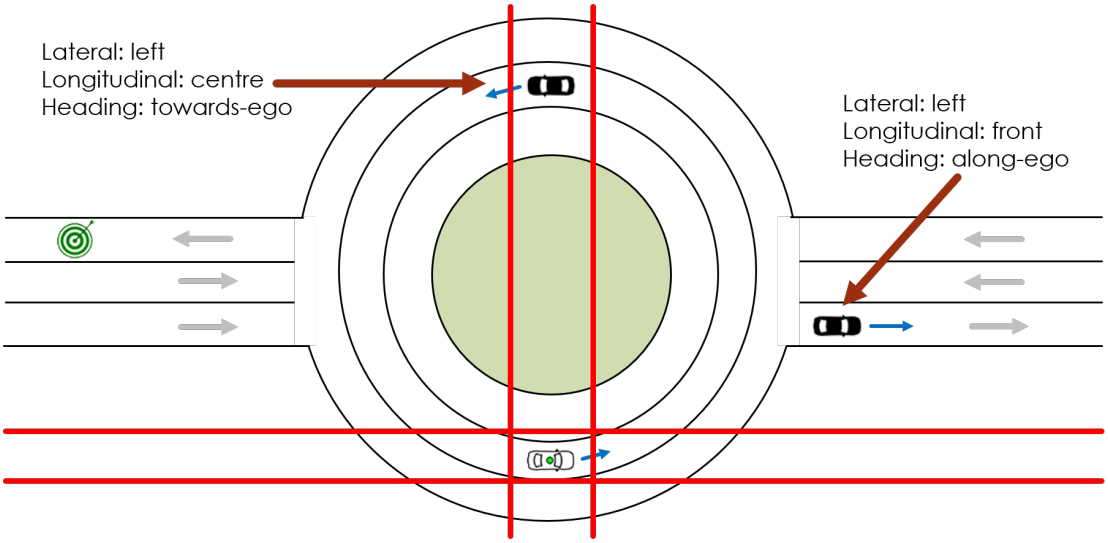


Figure 5.4: Visualization of the relative (egocentric) orientation coordinate system, with two sample classified dynamic objects.

Note that we apply the nine-grid partitioning of the space around Ego both to the *relative orientation* (relative to Ego's position and represented by the `relativeOrientation` block in Listing 6) and the (path-relative) *orientation* (relative to Ego's future position and represented by the `orientation` block in Listing 6).

Both orientation and relative orientation consist of heading and lateral/longitudinal position; however, since the `distance.x` and `distance.y` positions are already part of the input, we do not include the lateral/longitudinal positions in the `relativeOrientation` block.

Figure 5.4 shows the coordinate system used for the relative (egocentric) orientation of dynamic objects (represented by the red lines). The figure also shows two example cars and their classification using the coordinate system.

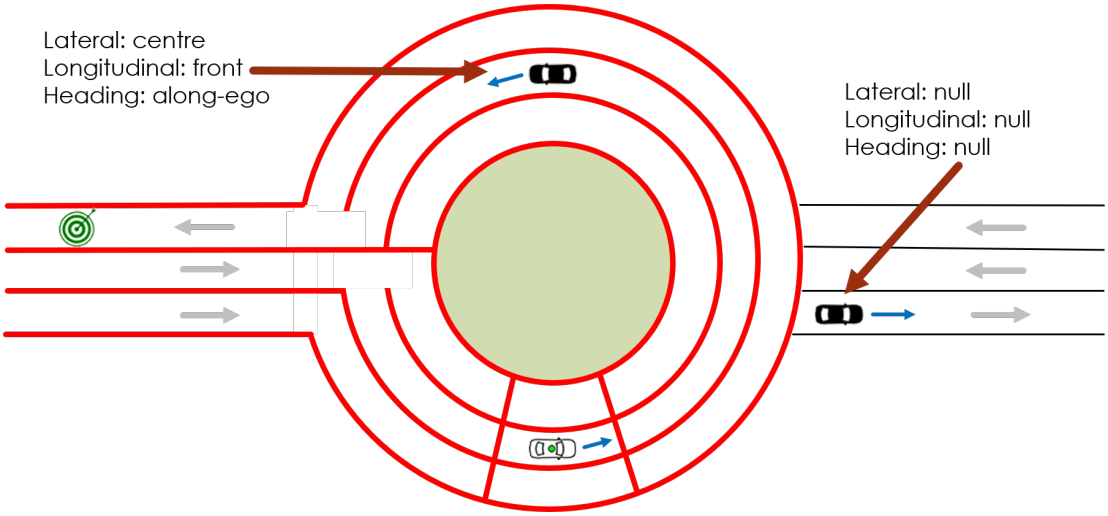


Figure 5.5: Visualization of the (path-relative) orientation coordinate system, with two sample classified dynamic objects.

Similarly, Figure 5.5 illustrates the coordinate system used to determine the (path-relative) orientation of dynamic objects. The coordinate system is defined relative to the sequence of road sections that Ego is traveling on. In our example, Ego arrives from the left and travels around the roundabout towards the green goal point. The figure also shows two sample cars and their classification. Note that the right car is outside the road segments travelled by Ego, and thus its orientation coordinates are undefined, i.e., all three attributes are null.

Scenario 3: Model Instance

Now that the attributes of each category of the world abstraction model, W_t, are known, we will apply these schemas to scenario 3 when Ego is at the initial location as depicted in Figure 3.5. In this scenario, we arbitrarily give to the pedestrian the identifier P_1 and to the vehicle the identifier V_1. Considering this, we obtain the valuation of the model depicted in Listing 7.

Listing 7 [Scenario 3] Model Instance

```
{
    "ego": {
        "location": {
            "approaching": "intersection",
            "at": "",
            "on": "drive-lane",
            "timeOfArrival": NULL
        },
        "navigation": "straight",
        "speed": 50
    },
    "pedestrians": [{
        "id": "P1",
        "distance": {
            "x": 856
            "y": -1.25
        },
        "isLeading": false,
        "isObstructing": true,
        "location": {
            "approaching": "",
            "at": "",
            "on": "intersection",
            "timeOfArrival": NULL
        },
        "orientation": {
            "heading": "left",
            "lateral": "center",
            "longitudinal": "front"
        },
        "relativeOrientation": {
            "heading": "left"
        },
        "speed": 2.25
    }],
```

```
    "travel": {
        "crosswalk": NULL,
        "intersection":  {
            "center": { "x": 849.99, "y": 4.12 },
            "distance": 850,
            "length": 6,
            "width": 6,
        },
        "regulation": "stop",
        "speedLimit": 50,
        "trafficLight": NULL
    },
    "vehicles": [{
        "id": "V1",
        "distance": { "x": 325, "y": 0 },
        "isLeading": true,
        "isObstructing": false,
        "location": {
            "approaching": "intersection",
            "at": "",
            "on": "drive-lane",
            "timeOfArrival": NULL
        },
        "orientation": {
            "heading": "along-ego",
            "lateral": "center",
            "longitudinal": "front"
        },
        "relativeOrientation": {
            "heading": "along-ego"
        },
        "speed": 35
    }]
}
```

5.2 Temporal Difference

Previously in Chapter 3.2, we mentioned that the preprocessing extracts historical and predicted attributes, and subsequently performs a temporal difference. However, in our minimalist study of the data flow in Chapter 4, we omitted these details that did not influence the decision-making process. However, in this section, we define what this temporal difference accomplishes.

At first, the system computes additional derived attributes by comparing the propositions at time t with that of time $t-1$. During this comparison, the atomic and raw world abstraction are compared to derive new attributes allowing the Rule-Engine to remember important historical information about the perceived environment. For instance, imagine that the memorized atomic world abstraction at time $t-1$ contains the proposition *regulatedStop* = *True* and that the world abstraction at time t contains the proposition *on* = *'intersection'*. Then, the system can derive the following proposition *wasRegulatedByAnIntersectionStop* = *True* that can be appended to the world abstraction at time t. Following this principle, the atomic world abstraction memorized at time t will constitute a time-free snapshot where the history of some state changes is abstracted into ternary attributes, which simplifies the development of the behavioural test suite as described in Chapter 6.

Ego's Temporal Attributes

Last maneuver: The maneuver initiated at the latest time step that has not produced an *emergency-stop* is used to remember, for instance, that the ego vehicle is performing an overtake and therefore it currently is allowed to travel in the opposite lane.

Stop begin at: The time at which the mandatory stop was first performed at the current intersection.

Stop time elapsed: The time period elapsed between the stop begin and the current time.

Travel Temporal Attributes

Crosswalk is conflicted: A ternary coming from a prediction algorithm indicating if Ego will collide with at least one pedestrian crossing the street at a crosswalk during the projected time period where Ego is expected to be on the crosswalk.

Was regulated by an intersection stop: A ternary remembering if the intersection on which Ego is travelling was regulated by a stop sign, such that it does not idle in the middle of the road to perform a left turn across the path.

After that, the system performs the temporal difference operation that we define as an attribute union. As you will see at the end of Chapter 5.5, the number of propositions is large even when there is only one vehicle and one pedestrian involved in the environment. In a real situation, the expected number of vehicles (including parked vehicles and misdetections) and the expected number of pedestrians can be considerably larger. Knowing that an autonomous vehicle wants to maximize the number of exchanges with the behavioural planner to increase the robustness of the sequence of behaviours derived from the driving policy, it is essential to minimize the computation time required to process each request.

The first step toward this minimization is to reduce the amount of data that the Rule-Engine requires for preprocessing and analysis to derive the next behaviour to initiate. To achieve this goal, suppose that for each attribute $a = (p, v)$, p is a proposition and v is its value (true, false, or null). Then, let the set of propositions in W_t be denoted by P_t. To encode temporal information, the system performs the following temporal difference on the world abstraction model, $W_t \leftarrow \{(p, v) \in W_t \mid v \vee \neg v\} \cup \{(p, v) \in W_{t-1} \mid p \in P_{t-1} \setminus P_t\}$. Namely, the condition $v \vee \neg v$ is used to extract the non-null attributes from the atomic world abstraction.

To exemplify the temporal difference, suppose the input received is $W_t = \{a, \neg b, \overset{\circ}{c}\}$, where $\overset{\circ}{c}$ denotes a null value, and that the memory state is $W_{t-1} = \{\neg a, b, c, d, \neg e\}$, where a, b, c, d, and e are, to simplify the example, atoms. Then, $W_t \leftarrow \{a, \neg b\} \cup \{d, \neg e\}$ since $\overset{\circ}{c}$ is null in W_t and the literals d, and e are not included in P_t.

The second step of this temporal difference constitutes the foundations of a memoization scheme. For instance, recall the previous example where $W_t \leftarrow \{a, \neg b\} \cup \{d, \neg e\}$. If we compare the attributes of this valuation of the world abstraction model with those of the previous step, we can easily discover that $\{a, b, c\}$ are the propositions that have changed at the current time step. Therefore, the system can use this information to memorize the output produced by each rule and only reactivate the subset of rules using at least one of these propositions in at least one of the disjuncts of its clauses. Moreover, it should also be noted that when the communication is minimal between the machine interface and the Rule-Engine, the proposition subset should always correspond to the attributes of the input model such as in the example of this section.

5.3 Long Short-Term Memory

Long short-term memory is a layer of the preprocessing stage that allows the Rule-Engine to remove dynamic objects from its memory. It should be remembered that the rule-based system performs incremental updates and therefore the absence of a dynamic object from the received scene can mean that (i) its attributes remain unchanged, or (ii) it is out of bound of the decision-making system, or (iii) it is temporarily obstructed by a dynamic or static object. To optimize memory management, this layer allows the knowledge expert to establish rules such as an update absence time or a distance to travel before deleting a dynamic object in the memory. If we assumed perfect perception, a dynamic object that has constant speed (including being stationary) might not produce any update for extended periods of time, and thus be deleted from the memory. However, we do not face this problem since perception is not perfect, and causes attributes such as speed and relative position to oscillate around their true value, and these changing values will be reported as updates. This in turn, will prevent the system from deleting these constant-speed or stationary visible dynamic object. Therefore, this layer decreases the system noise sensitivity allowing Ego to behave smoothly even when it travels in a partially observed environment.

Another goal of long short-term memory is to reconcile a small quantity of noisy data. In our system implementation, perception would provide only very noisy heading estimate for slowly moving dynamic objects, where the heading would change at random over time. This layer thus allows memorizing a history of values to establish whether the environment perceived at time step t is coherent with its history. For example, a vehicle that has decelerated from 10 km/h to 5 km/h in the history cannot suddenly have a heading that would suggest a reversed trajectory. Therefore, even if the world abstraction suggests that the value of an attribute has changed, the long short-term memory may overwrite it with a value deemed more likely. Indeed, one can design closely accurate overriding functions only based on empirical observation and analysis, since the approximated value depends of the sensor capabilities and the recovery mechanisms implemented in the autonomous vehicle stack.

5.4 Grouping Operations

Suppose that D_i is a dynamic object that Ego wants to locate using a relative orientation, where the middle of Ego's rear axle is the origin. Also suppose that Ego wants to locate D_i according to its path. Achieving these goals requires defining the set depicted in Listing 8.

Listing 8 Location of dynamic objects

```
{
    (...),
    "dynamicObjects": [{
        "identifier": "D1",
        (...)
        "orientation": {
            (...),
            "lateralCentre": true,
            "lateralLeft": false,
            "lateralRight": false,
            "longitudinalBehind": false,
            "longitudinalCentre": false,
            "longitudinalFront": true
        },
        "relativeOrientation": {
            (...),
            "lateralCentre": true,
            "lateralLeft": false,
            "lateralRight": false,
            "longitudinalBehind": true,
            "longitudinalCentre": false,
            "longitudinalFront": false
        }
    }, (...)],
    (...)
}
```

The problem with the object representation in Listing 8 is that each rule that checks for the presence of objects with a certain property will need to redundantly filter the collection. In addition, the memoization performs poorly because this entire large collection is flagged as changed whenever any attribute of its member object is updated, and thus the collection will trigger reevaluation of all rules that operate on it. Denoting R as the number of rules, these operations require $O(R|D|)$, which decreases the inferential process run time efficiency. A method to attenuate this problem is to flatten the dynamic object set using the Boyce-Codd Normal Form (BCNF) [11]. BCNF is a normal form commonly used in relational databases. It involves flattening all nested attributes, such as arrays and collections. This flattening allows rules to register more specific interest on attributes of subsets of dynamic objects. In our implementation, BCNF uses identifiers to reference each dynamic object, generating the set depicted in Listing 9.

Listing 9 Location of dynamic objects using BCNF

```
{
    (...),
    "dynamicObjects": {
        "D1": { (...) },
        "D2": { (...) },
        (...)
    },
    (...)
}
```

The normalization of Listing 9 creates a relational data schema ensuring that each proposition is accessible in constant time. As a result, the dynamic object set can be targeted directly by the rules such as "IF D_1 is [...]". While being time efficient, BCNF is limiting as some atomic propositions used in the clauses require to be quantified. To overcome this drawback, the preprocessing phase ends with grouping operations. Grouping avoids bounding the number of elements in the environment and enhances the dissimilarities amongst the propositions of the dynamic object set as depicted in Listing 10.

Although it may seem counter-intuitive to apply BCNF and then recreate collections, this process, in addition to allowing constant time access, avoids the run time complexity of redundantly finding a common collection of dynamic object amongst the rules (i.e. multiple rules requiring to find the subcollection representing the incoming traffic). Furthermore, it

also improves the memoization efficiency since the system can monitor the changes of atoms over the subcollections. For instance, let B be the *orientation.behind* subcollection, monitoring a predicate such as $(\exists d \in B_t \wedge \exists d' \in B_{t-1} | d.id = d'.id \wedge d.hasSpeed \neq d'.hasSpeed)$ is necessarily more efficient than monitoring it over the whole collection of dynamic objects, since $B \subseteq D$.

In the example studied in this section, we use the terminology of dynamic objects to simplify the notation, but in our system implementation we split them in pedestrian and vehicle since Ego takes different precautions when interacting with either category of these dynamic objects. For the remaining chapters of this thesis, we will use a subset of the 58 grouping operations performed by the Rule-Engine since these operations are only meant to boost the run time efficiency.

Listing 10 Improved location of dynamic objects using BCNF subcollections

```
{
    (...),
    "dynamicObjects": {
        "orientation": {
            "behind": ["D4", "D5"],
            "front": ["D1"],
            "left": ["D3", "D4"],
            "right": ["D2", "D5"]
        },
        "relativeOrientation": {
            "behind": ["D1"],
            "front": ["D2", "D5"],
            "left": ["D4"],
            "right": ["D3"]
        },
        "references": {
            "D1": { (...) },
            "D2": { (...) },
            [...]
        }
    },
    (...)
}
```

5.5 Preprocessing

We saw in the previous subsections of this chapter that the preprocessing stage is decomposed into several micro-tasks aiming to deduce new predicates, recover from noisy observations, and reshape the dataset used by the succeeding processes. We can now show the data format following the complete execution of this stage where this time, we decided to use “all”, “behind”, “front”, “left”, “leading”, “right” and “offRoad” as grouping operation.

Listing 11 [Scenario 3]: Complete Preprocessing Valuation


```
{
    "ego": {
        "exceedSpeedLimit": false,
        "hasLowSpeed": false,
        "hasPrecedence": false,
        "hasSpeed": true,
        "lastManeuver": {
            "decelerateToHalt": false,
            "emergencyStop": false,
            "followLeader": true,
            "overtake": false,
            "stop": false,
            "trackSpeed": false,
            "yield": false
        },
        "location": {
            "approachingCrosswalk": false,
            "approachingIntersection": true,
            "atCrosswalk": false,
            "atIntersection": false,
            "onCrosswalk": false,
            "onDriveLane": true,
            "onIntersection": false,
            "onOffRoad": false
        },
        "navigationLeft": false,
        "navigationRight": false,
        "navigationStraight": true
    },
    (...)
```

```
    (...)
    "references": {
        "P1": {
            "hasPrecedence": true,
            "hasSpeed": true,
            "isLeading": false,
            "isObstructing": true,
            "location": {
                "approachingCrosswalk": false,
                "approachingIntersection": false,
                "atCrosswalk": false,
                "atIntersection": false,
                "onCrosswalk": false,
                "onDriveLane": false,
                "onIntersection": true,
                "onOffRoad": false
            },
            "orientation": {
                "headingAlongEgo": false,
                "headingLeft": true,
                "headingLeftAlongEgo": false,
                "headingLeftTowardsEgo": false,
                "headingRight": false,
                "headingRightAlongEgo": false,
                "headingRightTowardsEgo": false,
                "headingTowardsEgo": false,
                "lateralCentre": true,
                "lateralLeft": false,
                "lateralRight": false,
                "longitudinalBehind": false,
                "longitudinalCentre": false,
                "longitudinalFront": true
            },
            (...)
        },
        (...)
    }
    (...)
```

```
    (...)
    "references": {
        "P1": {
            (...)
            "relativeOrientation": {
                "headingAlongEgo": false,
                "headingLeft": true,
                "headingLeftAlongEgo": false,
                "headingLeftTowardsEgo": false,
                "headingRight": false,
                "headingRightAlongEgo": false,
                "headingRightTowardsEgo": false,
                "headingTowardsEgo": false,
                "lateralCentre": true,
                "lateralLeft": false,
                "lateralRight": false,
                "longitudinalBehind": false,
                "longitudinalCentre": false,
                "longitudinalFront": true
            }
        },
        "V1": {
            "hasLowSpeed": false,
            "hasPrecedence": true,
            "hasSpeed": true,
            "isLeading": true,
            "isObstructing": false,
            "isParked": false,
            (...)
        }
    }
    (...)
```

```
    (...)
    "references": {
        [...]
        "V1": {
            (...)
            "location": {
                "approachingCrosswalk": false,
                "approachingIntersection": true,
                "atCrosswalk": false,
                "atIntersection": false,
                "onCrosswalk": false,
                "onDriveLane": false,
                "onIntersection": false,
                "onOffRoad": false
            },
            "orientation": {
                "headingAlongEgo": true,
                "headingLeft": false,
                "headingRight": false,
                "headingTowardsEgo": false,
                "lateralCentre": true,
                "lateralLeft": false,
                "lateralRight": false,
                "longitudinalBehind": false,
                "longitudinalCentre": false,
                "longitudinalFront": true
            },
            "relativeOrientation": {
                "headingAlongEgo": true,
                "headingLeft": false,
                "headingRight": false,
                "headingTowardsEgo": false,
                "lateralCentre": true,
                "lateralLeft": false,
                "lateralRight": false,
                "longitudinalBehind": false,
                "longitudinalCentre": false,
                "longitudinalFront": true
            }
        }
    }
    (...)
```

```
    (...)
    "pedestrians": {
        "all": [ "P1" ],
        "behind": [],
        "front": [ "P1" ],
        "left": [],
        "leading": "",
        "right": [ "P1" ],
        "offRoad": []
    },
    "travel": {
        "intersection":  {
            "centerPointInFront": true,
            "isObstructed": {
                "crossing": true,
                "leftTurn": false,
                "rightTurn": false
            }
        },
        "regulationStop": true,
        "regulationYield": false,
        "wasRegulatedByAnIntersectionStop": false
    }
    "vehicles": {
        "all": [ "V1" ],
        "behind": [],
        "front": [ "V1" ],
        "left": [],
        "leading": "V1",
        "parked": [],
        "right": [],
        "offRoad": []
    }
}
```

5.6 Maneuver Rules Evaluation

In the scenario under study, Ego must undertake several maneuvers before reaching the opposite side of the intersection. These maneuvers include: establish a deceleration point (*decelerate-to-halt*); wait for the leading vehicle to enter the intersection (*yield*); make a full stop for three seconds (*stop*); wait for precedence (*yield*); enter the intersection (*track-speed*); and finally, reach the marker when leaving the intersection (*track-speed*). Each of these maneuvers requires to model a set of rules allowing Ego to drive safely. In this thesis, we will only describe the rules to accomplish the first maneuver: **establish a deceleration point**. The remaining rules are given in the rule book B.

A set of rules should generalize to many situations. Thus, even if the anticipated stopping point must be behind the leading vehicle in the specific scenario, we must consider all the desirable stopping locations knowing that Ego intends to cross the intersection. Considering only the intersection entrance, we can already identify the following possibilities: stop at the stop line, and stop behind the leading vehicle at the stop line. On the other hand, we can also improve our modelling by employing courtesy rules such as: "When a pedestrian crosses a junction on our way, it is better to avoid blocking the intersection and thus, potentially allowing vehicles without precedence to cross meanwhile". Note that these courtesy rules apply regardless of whether the intersection is regulated by a stop sign or not. Therefore, we will add the following possibilities: stop at the end of the lane and stop behind the leading vehicle at the end of the lane.

Now that we have an idea of the range of our maneuvers, we will list the predicates used: (A) *ego.location.approachingIntersection*, (B) *travel.regulatedStop*, (C) *vehicles.leading*, (D) $v \rightarrow$ *references[v].location.approachingIntersection*, (E) $v \rightarrow$ *references[v].location.atIntersection*, (F) *pedestrians.all*, (G) $p \rightarrow$ *references[p].location.onIntersection*, and (H) $p \rightarrow$ *references[p].isObstructing*. Unlike the previous chapter, some of these predicates are quantifiable, such as the predicate functions D, E, G and H. Therefore, the dynamic objects on the scene should be appropriately quantified in the rules.

$$(A \wedge B) \vdash (\textit{decelerate-to-halt}, \alpha) \tag{5.1}$$

$$(A \wedge B \wedge (\exists v \in C | D(v)) \vee A \wedge B \wedge (\exists v \in C | E(v))) \vdash (\textit{decelerate-to-halt}, \beta) \tag{5.2}$$

$$(A \wedge (\exists p \in F | G(p) \wedge H(p)) \vdash (\textit{decelerate-to-halt}, \omega) \tag{5.3}$$

$$(A \wedge (\exists v \in C | D(v)) \wedge (\exists p \in F | G(p) \wedge H(p))) \vdash (\textit{decelerate-to-halt}, \lambda) \tag{5.4}$$

$$(A \wedge (\exists v \in C | E(v)) \wedge (\exists p \in F | G(p) \wedge H(p))) \vdash (\textit{decelerate-to-halt}, \lambda) \tag{5.5}$$

Here, we should notice that the Equations 5.4 and 5.5 are meant to be a single rule in disjunctive normal form like Equation 5.2, but to improve readability we have split it in two equations.

According to Listing 11 which lists all the properties after the preprocessing stage, all these rules match the current state and therefore, they produce the outputs α, β, ω, and λ, which are the following set of values.

Listing 12 [Scenario 3]: Maneuver rule outputs

```
α: {
    "abstractLocation": "stop-line"
}

β: {
    "abstractLocation": "stop-line",
    "leadingVehicle": "V1"
}

ω: {
    "abstractLocation": "end-of-lane"
}

λ: {
    "abstractLocation": "end-of-lane",
    "leadingVehicle": "V1"
}
```

5.7 Precedence Table

As mentioned in the previous subsection, each rule in our example matches the World Abstraction, W_t. Thus, when the precedence table seeks the priority maneuver, there is only one option, making the choice of the *decelerate-to-halt* maneuver trivial. Nevertheless, since we have several constraint outputs disagreeing, we can illustrate the second layer of preprocessing.

Listing 13 [Scenario 3]: Considered Constraints Input Model


```
{
    "abstractLocation": {
        "endOfLane": true,
        "stopLine": true
    },
    "leadingVehicle": true,
    "leadingPedestrian": false,
    "maneuver": {
        "decelerateToHalt": true,
        "emergencyStop": false,
        "followLeader": false,
        "overtake": false,
        "stop": false,
        "trackSpeed": false,
        "yield": false
    }
}
```


This preprocessing is like the one we saw earlier for the input model of the Rule-Engine: it consists of merging the received proposals, transforming them into propositions, and adding the selected maneuver as a proposition. To do this, the preprocessor applies the same strategies as previously established such as to transform each discrete value into a ternary and to occasionally merge several ternaries creating derived propositions aiming to reduce the activation frequency of the constraint rules. As we have already seen the whole theory to understand these concepts, we can immediately examine the data generated for our current example shown in Listing 13.

5.8 Constraint Rules Evaluation

Recall that the purpose of the constraint rules is to filter the propositions issued by the maneuver rules such that the end behaviour constitutes a safe reconciliation of the given constraints. In the previous chapter, we had already illustrated the *track-speed* constraints resolution mechanism searching for the minimum cruising speed amongst the generated upper bounds. Now, let's use the nullable-Boolean constraint's model, $\mathcal{C}_t$, to formulate the following rules: "When the selected maneuver is *decelerate-to-halt* and Ego can stop at the stop line, it should be included in the final constraint" (Equation 5.6), "When the maneuver is *decelerate-to-halt* and a leading vehicle exists, the vehicle should be included in the final constraint" (Equation 5.7), and "When the selected maneuver is *decelerate-to-halt* and there is no stop line but there is the end of the lane, the end of the lane should be included in the final constraint" (Equation 5.8).

As we have seen, rules are built to include or exclude information from the input model. More concretely, let the predicates: (A) *maneuver.decelerateToHalt*, (B) *abstractLocation.stopLine*, (C) *leadingVehicle*, and (D) *abstractLocation.endOfLane* be attributes of the considered atomic constraint model (see Listing 13) as well as (E) *leadingVehicle* be the referential string of the considered non-atomic constraint model (see Listing 12). With these predicates, we can build the following rules:

$$(A \wedge B) \vdash \{\ \text{“abstractLocation”: “stop-line”}\ \} \tag{5.6}$$

$$(A \wedge C) \vdash \{\ \text{“leadingVehicle”: E}\ \} \tag{5.7}$$

$$(A \wedge \neg B \wedge D) \vdash \{\ \text{“abstractLocation”: “end-of-lane”}\ \} \tag{5.8}$$

When we apply these rules to the atomic input model, $\mathcal{C}_t$, and perform the union of retained propositions, we obtain the behaviour that Ego will initiate: (*decelerate-to-halt*, { “abstractLocation”: “stop-line”, “leadingVehicle”: “V1” }). In this case, the final behaviour corresponds exactly to the rule in Equation 5.2. However, it is also possible that the selected behaviour is the complete or partial union of several maneuver rules. For instance, suppose that the active maneuver rules were Equation 5.1, 5.3, 5.4 and 5.5. According to the constraint rules, the selected behaviour remains unchanged (*decelerate-to-halt*, { “abstractLocation”: “stop-line”, “leadingVehicle”: “V1” }). Therefore, this behaviour is the partial union of Equation 5.1, 5.4 and 5.5 since in the proposition α, the abstract location has been retained while the leading vehicle was unspecified, and in the proposition λ, the abstract location has been rejected while the leading vehicle was retained.

In this chapter, we saw the properties of the dynamic objects inputted to the Rule-Engine. We applied them on the third scenario depicted in Figure 3.5. This scenario illustrates an interaction amongst Ego, a leading vehicle and a pedestrian nearby an intersection. This scenario allowed us to review the stages of preprocessing, maneuvers rules, precedence table and constraint resolution rules. In particular, we have seen that the preprocessor performs temporal difference operations, long-short term memory with potential attribute overwrite, grouping operations and uses a memoization scheme to accelerate the deduction of the maneuver to be executed at the next time step. We then refined the design of the rules by adding functions to iterate over collections of dynamic objects. We illustrated this design process by defining four rules controlling the Ego behaviour when approaching the stop line of an intersection regulated by an all-way stop. Next, we detailed the merging process of the precedence table by agglomerating the constraints of the four rules created. This set of inconsistent constraints was further filtered by the constraint rules until a consistent constraint set is obtained. Afterward, we defined the end behaviour by creating the tuple composed with the maneuver retained by the precedence table as well as the constraints filtered by the constraint rules. This concludes the description of the decision-making process of our expert system. In the next chapter we will discuss how to qualify a rule set aiming for developing better rules.

Chapter 6

Tool Assisted Quality Assurance Method

In the three previous chapters, we viewed a design of an expert system applied to behavioural planning for urban self-driving. Nevertheless, one of the deficiencies commonly attributed to this type of architecture is that it is difficult for a human to build a book of rules covering exhaustively an operational design domain. Although the objective of this thesis is not to produce a complete rule set, we present a rule production methodology that allows us to increase the capabilities of our system while the operational design domain is growing.

An effective methodology is essential since the number of collisions amongst rules in an unordered rule-based system, such as the Rule-Engine, grows exponentially with the number of rules. As a result, it becomes unmanageable for a human to comprehend the impact of creating or altering a rule. Fortunately, we implemented a test-driven technique that can address this challenge. In this tool-assisted technique, the developers can use their intuition to improve the rules until the generated set supports the operational design domain, while, concurrently, the machine checks if all the situations tested satisfy the expected behaviours and suggests new edits by providing a rationale on what went wrong.

In this chapter, we will discuss how the tool ensures the desired behaviour of the rule set, the metrics used to qualify the rule set, as well as its integration with the Rule-Engine architecture.

6.1 Unit Test Method

The primary purpose of this tool-supported method is to ensure that the Rule-Engine gradually reaches a comprehensive and correct set of rules. To do this, the tool assures that the built rules support their test cases. Whenever a test fails, the tool assisted-method uses the traceability of the Rule-Engine to identify the set of rules along with their matching clauses reported in the failing test case. This allows the developer to focus only on the outputted subset of rules and clauses. While incrementally editing the knowledge base, this process is repeated until the test suite is satisfied.

In our system, we define a test case as a valuation of the World Abstraction Model, $\mathcal{W}_k$, where the internal state of the rule-based system must match the label produced by the knowledge expert. Each label corresponds to a quadruplet, (W_k, R_m, M^*, C^*), where $\mathcal{W}_k$ is a valuation of the World Abstraction Model, R_m is the maneuver rule number that should survive to the system filtering (i.e. the constraints, C_m, outputted by the maneuver rule R_m correspond exactly to the retained constraints, $C_m = C^*$ subject to C^* being derived from R_m), M^* is the desired maneuver, and C^* is the desired constraints to be respected, as associated with the maneuver. Later, in Chapter 7.4, we will see that R_m is helpful when identifying gaps amongst test cases during the knowledge acquisition.

For this verification to be gradual, the knowledge expert builds the labels by applying the principles of Test-Driven Development (TDD). Its purpose is to create a behavioural test suite, $\mathcal{B}$, consisting of labels covering the extent of the operational design domain in which the autonomous vehicle must drive. Since, in an expert system using an unordered set of rules, each rule acts concurrently, these tests are designed to ensure that the rule set produces the right maneuvers and constraints. Thus, the tests check the integration amongst the rules rather than their standalone behaviour.

As previously mentioned, in an unordered rule-based system, adding a new rule may influence the precedence of any subset of its previous rules. For example, even if a situation was once covered by rule A, the inclusion of a new rule that uses a subset of the attributes of rule A may be enough to change the domain in which, in a collective evaluation, the maneuver and constraints produced by rule A survives to the system filtering. Thus, for each modification, the knowledge expert wants to be convinced that the modification still allows the autonomous vehicle to achieve the behaviours already included in the test suite as well as the new ones.

It is clear that the knowledge expert cannot cover all valuations of the World Abstraction Model, W, since this model includes continuous attributes such as positions and velocities as well as an indefinite number of dynamic objects. Therefore, the knowledge ex-

pert looks for an incomplete test suite that will be refined during the development process of the various modules integrated in the autonomous vehicle.

The test choice is intuitively correlated with the rule set constituting the Rule-Engine. Since the rules are defined in the disjunctive normal form, each disjunct describes a situation to be assessed. Moreover, as each rule only uses a subset of the World Abstraction Model propositions, it is unclear whether we should instantiate each attribute of the World Abstraction Model to describe a complete environment or whether the value of the remaining attribute should be left unspecified. For instance, a disjunct of a rule premise can be used to define one or more test cases. A disjunct is a conjunction of constraints on attributes, often equalities, and any test case derived from it will respect these constraints. When deriving a test case, we need to decide what to do with the attributes that are not mentioned in the disjunct. We could either create minimal test cases, which try to omit the unmentioned attributes (sometimes the wellformedness of a model might require some of the unmentioned attributes to be included), or test cases that include additional unmentioned attributes. When defining a test case that includes additional unmentioned attributes, the test case increases the chance to match additional rules, either existing or future, which would make defect localization difficult. On the other hand, minimal test cases may sometimes represent contrived situations that do not occur in reality. As a result, the test designer needs to find a balance between these concerns.

To illustrate the contrived situations claim, suppose we design a rule using the following predicates: (A) *ego.location.onIntersection* and (B) *ego.hasPrecedence* such that $(A \wedge B) \implies a$. The rule is well-formed and uses a subset of the predicates from the World Abstraction Model. Unfortunately, the predicate *ego.hasPrecedence* means that Ego has the precedence at a stop or yield sign. Since there does not exist such signal on the physical area of an intersection, then this combination of attributes is already unattainable. Sometimes, the designer can also create a rule that does not violate the meaning of any attributes but that remains unattainable. Introducing the predicate (C) *ego.location.onOffRoad*, suppose the following pair of rules $(A) \implies \alpha$ and $(A \wedge \neg C) \implies \beta$. Since *onIntersection* implies to be on the physical area of an intersection, it is obvious that Ego cannot be off-road simultaneously. Therefore, even if the ontology is properly used, we are still in a situation where the behaviour α will always be overridden by the behaviour β or vice-versa depending on the precedence of these maneuvers.

Beyond covering maneuver rules, a knowledge expert also needs to check the constraint rules. For this task, the system compares the quintuplet, $(W_k, R_m, R_c, M^*, C^*)$, where R_m is now a **set** of maneuver rule numbers, R_c is a set of constraint rule numbers, whereas $\mathcal{W}_k$, M^* and C^* remain unchanged. Here, the knowledge expert does not seek to isolate a situation targeted by a single rule but rather wants to consider the interaction of multiple

rules having the same level of precedence. For example, the third scenario that we described in the previous chapter is a suitable example since four rules with the same precedence (i.e., *decelerate-to-halt*) were juxtaposed in the decision-making.

Finally, when the behavioural test suite, $\mathcal{B}$, is well defined and includes both test types, the knowledge expert has a test methodology to probe the behaviours of the autonomous vehicle in polynomial time such as $\mathcal{O}(|\mathcal{B}| * p)$, where p is the mean polynomial time required by the Rule-Engine to process each query.

6.2 Quantitative Evaluation

The behavioural test suite, $\mathcal{B}$, is an approximation of the desired behavior-defining function that can be covered by several distinct rule sets. Therefore, the second goal of our tool assisted method is to compare two sets of rules based on the efficiency of their execution and their maintainability as stated in the design objectives of Chapter 2. This comparison focuses primarily on the interaction among the rules. The metrics defined in this chapter are used to assess the most desirable rule set generated by Algorithm 1 of Chapter 7.5.

With respect to efficiency, our method considers a single metric which is the number of activations of a rule defined as follows.

Activation: Number of times a rule R_i is evaluated such that the value of at least one of the predicates included in its clauses has changed.

This metric identifies the frequency at which the rules must be evaluated. For instance, the rule created in Chapter 4.3 depended on the following six attributes: (A) *location.onDriveLane*, (B) *location.approachingCrosswalk*, (C) *location.approachingIntersection*, (D) *location.atCrosswalk*, and (E) *location.atIntersection* from the atomic Ego category and, finally, the value from the world abstraction (F) *speedLimit* of the Travel category, used as a constraint. In comparison, we can consider another rule which combines all the Ego location category attributes in a new derived predicate called (G) *location.onlyOnDriveLane*. With this alternative design, the rule can be expressed as Equation 6.2, which is simpler than Equation 6.1.

$$(A \wedge \neg B \wedge \neg C \wedge \neg D \wedge \neg E) \implies (\textit{track-speed}, F) \tag{6.1}$$

$$G \implies (\textit{track-speed}, F) \tag{6.2}$$

From the efficiency point of view, the designer must determine the fastest implementation. For example, the boolean *location.onlyOnDriveLane* could be obtained using: **IF** *location.approaching* is empty **AND** *location.at* is empty **THEN** *location['onlyOn' + location.on]* = true. In this case, the method evaluates the entire behavioural test suite, $\mathcal{B}$, checking whether it is better to execute Equation 6.1 each time one of the five monitored boolean values changes or to incorporate the derived predicate by performing the two comparisons of the derived clause at each time step, t, such as in Equation 6.2. Intuitively, we see that Equation 6.2 is significantly less active since it evaluates to true for only one of the 2^5 boolean combinations of Equation 6.1. Nevertheless, we still run both versions to determine whether using the derived predicate is an actual improvement.

One could wonder why we use the activation metric rather than the execution time of the behavioural test suite. Knowing that the activation metric is used to avoid rule re-evaluation, it requires a temporally correlated observation sequence to determine the actual execution time saving. However, the behavioural test suite, is not temporally correlated since the tests aim to evaluate the autonomous vehicle behaviours in specific independent situations. Thus, the observation history during the execution of the suite is not a realistic approximation of the expected histories during actual driving. Therefore, to overcome this limitation, we have setup the following weight matrix P using the problem decomposition described in Chapter 7.1 where we define the cost of accessing an attribute using the average number of dynamic objects (i.e., $\vec{\alpha}$, and $\vec{\beta}$) encountered during public drive.

$P :=$

Predicate Type	P1	P2	P3	P4	P5	P6	P7
Ego	1	1	1	1	1	1	1
Travel	1	1	1	1	1	1	1
Pedestrian	α_1	α_2	α_3	α_4	α_5	α_6	α_7
Vehicle	β_1	β_2	β_3	β_4	β_5	β_6	β_7

$C_i :=$

Predicate Type	P1	P2	P3	P4	P5	P6	P7
Ego	e_1	e_2	e_3	e_4	e_5	e_6	e_7
Travel	t_1	t_2	t_3	t_4	t_5	t_6	t_7
Pedestrian	p_1	p_2	p_3	p_4	p_5	p_6	p_7
Vehicle	v_1	v_2	v_3	v_4	v_5	v_6	v_7

To approximate the computation cost of a rule set, let $\vec{a_b}$ be an indicator vector that specifies whether a rule has been activated during a behavioural assessment. Also, let C_i be a sparse matrix telling the number of Ego, Travel, Pedestrian, and Vehicle atomic attributes included in the clause of a rule R_i for each of the seven subproblems described in Chapter 7.1. Thus, the computation cost of the rule set can be approximated with

Equation 6.3 which compensates for the lack of temporality and the restricted number of dynamic object on the scene.

$$\sum_{b} \sum_{i=1}^{|R|} a_{b,i} \sum_{j=1}^{4} \sum_{k=1}^{7} P_{j,k} C_{i,j,k} \tag{6.3}$$

In addition to evaluating the run time effectiveness of the rule sets, the method establishes a maintainability score for each rule set covering the behavioural test suite, $\mathcal{B}$. We define maintainability as the ease for the knowledge expert to modify the knowledge base. The maintainability of the rules can be estimated by observing whether the rules that matched the valuation of the World Abstraction, $\mathcal{W}_k$, contributed to the final behaviour determined by the Rule-Engine. This aspect can be decomposed into the following four metrics computed for $\mathcal{B}$:

Support: Number of times the output of the rule R_i was included in the adopted behaviour.

Redundancy: Number of times a rule R_i leads to the same constrained maneuver as another rule R_j during the same situation.

Greediness: Number of times the output of rule R_i was prioritized and therefore masks the output of another rule R_j.

Rejection: Number of times the output of rule R_i is masked by the output of another rule R_j after the prioritization.

At first glance, we already know that the rule sets being compared must each cover the behavioural test suite, $\mathcal{B}$, and the suite must cover the rules in each set before probing rule maintainability. Thus, one could wonder about the utility of calculating the number of supports since each rule is necessarily supported. Even though this is a fair intuition, the support metric helps to establish the importance of each rule. For instance, a system with a balanced number of supports amongst the rules does not rely on the trustworthiness of a single rule. Therefore, it includes a natural redundancy mechanism to restore a viable state if an observation was to be misdetected.

To illustrate this claim, imagine that the ego vehicle is approaching an intersection regulated by a stop sign. The rules could be constructed such that the intersection state is only observed once Ego starts its mandatory stop since it necessarily must stop before getting the option to enter the intersection. Unfortunately, suppose the stop sign is obstructed so that Ego does not anticipate the need to decelerate. If the system is well

balanced in its supports, an alternative observation such as "a vehicle not controlled by a stop sign is approaching" will allow Ego to slow down/yield and therefore make a stop even if the mandatory stop situation should have taken precedence if it was detected successfully. Thus, this model with the additional support for the situation is more robust and desirable since the rule "If Ego encounter a stop sign, it must stop" alone would be too brittle.

Further, the number of supports is also used to assess if the rule has at least one unique support. For instance, if we take the Equations 6.1 and 6.2 we can conclude that the two rules have exactly the same domain and range. However, in a large-scale design, the number of rules could be significant to a point where it is difficult to detect these redundancies. To establish whether a Rule A could be removed, it must be ascertained whether an agglomeration of one or more rules resulting in the same decision covers the domain of Rule A. With the metrics mentioned above, we can establish that a rule is not needed and can be removed if its support number is equal to its redundancy number.

Furthermore, the greediness and rejection metrics allow us to establish the level of coupling amongst the rules. Naturally, a rule set is more easily extensible if the number of situations activating multiple rules simultaneously is minimal. For example, if for a valuation of World Abstraction, $\mathcal{W}_k$, the maneuver rules are voting for *decelerate-to-halt*, *yield*, *stop*, and *emergency-stop*, then it becomes very difficult to change the resulting behaviour since a significant number of rules have to be analyzed because of the precedence table and the constraint resolution rules succeeding it.

It is also of interest to note that greediness necessarily implies the presence of rejection. Nevertheless, these metrics do not constitute a one-to-one relationship. That is, for example, for a given decision, there could be two counts of greediness for five rejections. Therefore, we define the coupling as the sum of these two metrics, which encapsulate the ease for the knowledge expert to modify existing behaviours.

Now that we have gained insight on the proposed metrics, we can describe the impact of the Chapter 5.8 statement "it is also possible that the selected behaviour is the complete or partial union of several maneuver rules". Since the behaviour derived from the driving policy does not have to correspond exactly to one of the maneuver rule suggested constraints, we refine our notion of support by adding that the support of a behaviour is the subset of constraints with minimum cardinality that suffice to generate the behaviour. As an example, let's return to the third scenario where the maneuver rule constraints were α, β, ω, and λ. Knowing that the final decision exactly matches β and that α is a non-strict subset of β, we determine that only β supports the decision while all others are rejected. This means that the maneuver rule output pair (α, β), even if it produces the same derived

behaviour, does not count as a support. Thus, the rule generating β has greedy support while the rules generating α, ω, and λ count as rejected options. There is no redundancy in this example since no other rule produces exactly β. In other words, if at another time step the candidate constraints would have been α, β, ω, and α (i.e., two distinct rules producing α), there would still be no redundancy since α is not the behaviour retained.

Finally, consider the second scenario where Ego wants to pass two parked vehicles, V_1 and V_3. Even if we did not concretely define the rules modelling this scenario, suppose that our alternatives are (A) passing V_1 (B) passing V_3, and (C) passing V_3 considering the leading vehicle V_2. Moreover, suppose that after the constraint rules, the adopted behaviour is passing V_1 and V_3 considering the leading vehicle V_2. In this example, the union of A and C becomes our support even if the union of A, B, and C would also have been a candidate. We retain the union of A and C since this union is the smallest subset supporting the decision. With respect of this union, A and C are greedy supports while B is a rejection. The only exception where we will accept several alternatives as a behavioural support is when several unions have the same cardinality. Let us arbitrarily define the union of A, B, and C as well as the union B, D, and E generating the same end behaviour which is indistinguishable since they each have a cardinality of three. From this perspective we shall say that A, B, C, D, and E are supports while all of them except B are redundant.

Our restrictions on computation of the metrics are primarily used to provide an unambiguous definition that can be applied to qualify a rule set in practice. These definitions, although debatable, avoid emphasizing negligible information when comparing two rule sets. For example, it is appropriate to prefer a set of rules where the decisions are minimally redundant, and we want to avoid being skewed by the intrinsic redundancies generated by decisions that are not retained and therefore superfluous. On the other hand, since our method penalizes equally the greediness, redundancy and rejection metric, we can arbitrarily label all constraints subset with higher cardinality than the supporting subset as a rejection even if some of them could have been labelled as redundancies. We will see in the next chapter how to use these metrics to help us model our knowledge base.

6.3 Test Interface

This chapter ends with additional guidance on the design of the behavioural test suite, $\mathcal{B}$. On the one hand, we have previously mentioned that the behavioural test suite, $\mathcal{B}$, is not temporally correlated. This requirement stems mainly from the fact that the Rule-Engine does not simulate a continuous interaction among the dynamic objects since most of the

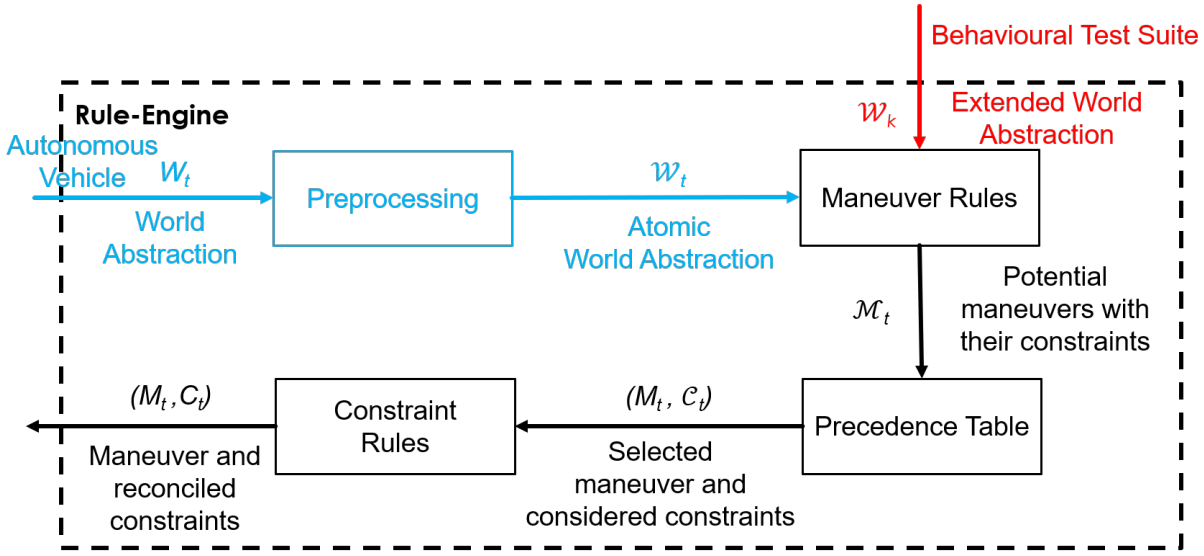


Figure 6.1: Splitted High-level architecture diagram.

received attributes are abstract and therefore already reflect an imprecise situation. This imprecision is caused by the existence of several environment representations that can be abstracted using the same attribute valuation. Thus, the mandate of the tool assisted technique is to build a sample test suite to validate the behaviour of the autonomous vehicle while abstracting from the concrete meaning of each situation and their temporal sequence.

To avoid having to model realistic situation sequences, the behavioural test suite, $\mathcal{B}$, must be able to remove the effects of the preprocessing that corrects the attributes based on the observation's history. To achieve this, we created a process that sends a World Abstraction, W_t, where each attribute has a null value between each behavioural assessment. This empty entry suffices to remove any kind of correlation between the behavioural assessment since the disparity between this state compared with the observation history is too significant for the Rule-Engine to successfully reconcile the data.

To ensure that each component of the Rule-Engine is properly tested, we allowed the behavioural test suite, $\mathcal{B}$, to submit a time-free snapshot $\mathcal{W}_k$ (in red in Figure 6.1) of the World Abstraction Model. The attributes in $\mathcal{W}_k$ included in the World Abstraction Model, W_t, (in blue in Figure 6.1) are first processed conventionally (Listing 14) and generate an Atomic World Abstraction, $\mathcal{W}_t$. Using the restructuring properties wisely, the remaining attributes not included in W_t are fused using the temporal difference operation defined in Chapter 5.2. This restructuring overwrites the values of $\mathcal{W}_t$ which could not be interpreted

Listing 14 Ego's fragment of a world abstraction snapshot submitted to the Rule-Engine containing temporal (W_t) and time-free (W_k) attributes

```
{
    "ego": {
        "location": {
            "approaching": "",              // Wt
            "at": "intersection",           // Wt
            "on": "drive-lane"              // Wt
        },
        "navigation": "straight",           // Wt
        "speed": 0,                         // Wt
        "stopBeginAt": 100,                 // Wk
        "stopTimeElapsed": 3100             // Wk
    },
    [...]
}
```

successfully by the preprocessing stage (Listing 15).

To illustrate this override, let's exemplify this process using the three-second mandatory stop situation. We can break this scenario into three distinct situations: (A) When Ego reaches the stop line, it makes a complete stop; (B) Immediately before 3 seconds, the ego vehicle is still restricted in a stationary position; (C) From 3 seconds and beyond, Ego can resume its journey. Thus, in the behavioural test suite, $\mathcal{W}_k$ will have information on the attributes *ego.stopBeginAt* and *ego.stopTimeElapsed* which are normally calculated internally by the Rule-Engine. This override accelerates the test suite execution, which can then independently test each situation without worrying about the test execution order.

The presented method allows the knowledge engineer to quickly compare alternative designs when extending a rule base. Even if the test suite is insensitive to temporal attributes, for the tool assisted quality assurance method to be realistic, the knowledge expert must frequently compare the results of the behavioural test suite with concrete urban self-driving scenarios. The goal is to assess if the test suite includes enough examples involving various combination of rules such that the metrics approximate fairly the reality. The process is covered in the next chapter.

Listing 15 The preprocessing giving precedence to the attributes of W_k on Ego's fragment.

```
{
    "ego": {
        "exceedSpeedLimit": false,
        "hasLowSpeed": false,
        "hasPrecedence": false,      // Derived from Wt
        "hasPrecedence": true,       // Derived from Wk + Overridden
        "hasSpeed": false,
        "lastManeuver": {
            "decelerateToHalt": false,
            "emergencyStop": false,
            "followLeader": false,
            "overtake": false,
            "stop": false,
            "trackSpeed": false,
            "yield": false
        },
        "location": {
            "approachingCrosswalk": false,
            "approachingIntersection": false,
            "atCrosswalk": false,
            "atIntersection": true,
            "onCrosswalk": false,
            "onDriveLane": true,
            "onIntersection": false,
            "onOffRoad": false
        },
        "navigationLeft": false,
        "navigationRight": false,
        "navigationStraight": true,
        "stopBeginAt": null,         // In the wiped history
        "stopBeginAt": 100,          // Overridden by Wk
        "stopTimeElapsed": null,     // In the wiped history
        "stopTimeElapsed": 3100      // Overridden by Wk
    },
    [...]
}
```

In this chapter, we have developed a series of metrics to qualify the efficiency and maintainability of a set of rules. We first explained that the efficiency metric must correspond to the number of activations since the test suite lack of a temporality notion and therefore does not reflect the reality of autonomous driving. On the other hand, we have partitioned the maintainability metric into the sum of four sub-metrics, namely the number of greediness, redundancies, rejections, and supports. In our way of designing rule sets, we prefer those that balance efficiency and maintainability to fit the design objectives of Chapter 2.

Chapter 7

Rule Set Development

So far, we have discussed an architecture to infer the behaviour that an autonomous vehicle must adopt at the next time step, t. We also described a tool assisted method for testing and comparing multiple rule sets and we have provided insight about how the Rule-Engine integrates with the tool. This chapter describes the overall process of creating the rule base using the test-driven and metric-driven method.

To achieve this, we first discuss a modelling strategy using a pivot point to control the applicability of each rule. Next, we describe the overall rule development process with

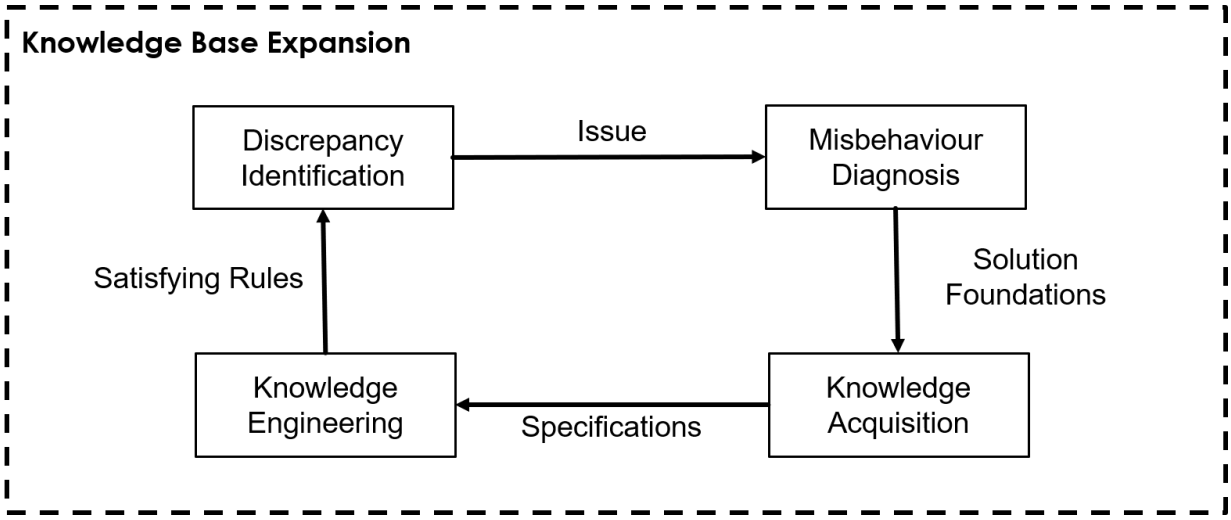


Figure 7.1: Rule Set Development high-level process flow

four stages (in Figure 7.1): Discrepancy Identification identifies one or more inadequate behaviours (issues) in a system using the rule set A; Misbehaviour Diagnosis extracts, from a given issue, an abstraction of the scene that exhibits the issue such that only the essential dynamic objects are included; Knowledge Acquisition compares the abstraction of the scene with the situations involving the same rule or group of rules in the behavioural test suite. During the process, the knowledge expert determines if the misbehaviour is justified by some limitations of the operational design domain or if it is caused by a gap in the behavioural assessments. In other words, Knowledge Acquisition uses the test suite to provide a rationale describing the derivation of each maneuver and constraint matching the world abstraction and causing the misbehaviour; Finally, Knowledge Engineering derives a set of candidate rule sets each satisfying the behavioural test suite and determines with the tool assisted quality assurance method which rule set exhibits the best efficiency and maintainability score. The selected rule set would then become the new implementation of the grown operational design domain specification.

7.1 Modelling Strategy

An unordered set of rules, R, is directly affected by the curse of dimensionality since, given a situation, each rule may vote leading to a worst-case complexity of $2^{|R|}$ combinations that could affect the end behaviour. This number increases significantly during the development of the knowledge base leading to difficulties for a human to understand the impact of each individual rule. Therefore, we will introduce the notion of a pivot attribute to partition the input space and reduce the worst-case complexity.

The idea is to partition the input space with a discrete attribute used to decide if a

Table 7.1: Rule-Engine problem space

Problem	Description
P1	Ego drives on a two-lane road excluding intersections
P2	Ego is approaching an intersection with a mandatory stop
P3	Ego is at a stop line
P4	Ego is on an intersection after a mandatory stop
P5	Ego approaches an intersection without a mandatory stop
P6	Ego is about to enter an intersection without a mandatory stop
P7	Ego is on an intersection without a mandatory stop

rule could vote in a given subspace. In our implementation, we selected Ego's abstract location type as our pivot attribute. This means that each rule created will first declare in which location Ego can be for this rule to hold. Under this assumption, the worst-case complexity can be defined as the maximum subspace complexity. In our experiment, our operational design domain contains the seven subspaces shown in Table 7.1. Naturally, each subspace could be seen as an independent problem to solve as depicted in Figure 7.2.

Figure 7.2 depicts two sets of two lane roads which eventually crosses at an intersection. Suppose that the horizontal road is regulated by a stop sign and that the vertical road is not regulated by a stop sign, and thus the vehicles have precedence. Whenever Ego needs to cross an intersection regulated by a stop sign, it will follow the state flow of the horizontal road. Thus, it sequentially encounters the problem spaces $P1 \rightarrow P2 \rightarrow P3 \rightarrow P4 \rightarrow P1$. Whenever Ego needs to cross an intersection not regulated by a stop sign, it will follow the state flow of the vertical road. Thus, it sequentially encounters the problem spaces $P1 \rightarrow P5 \rightarrow P6 \rightarrow P7 \rightarrow P1$. Although that $P4$ and $P7$ represent the same physical area, their problem spaces are still mutually exclusive by the temporal attribute $wasRegulatedByAStopSign$.

The pivot choice influences the ease of rule set development. For instance, suppose that the Ego vehicle is navigating in the orange area delimited by Problem 5. Also assume that suddenly, Ego is observing that a vehicle with a mandatory stop has a too high velocity to stop at its stop line. In such event, Ego would like to perform a hard brake to avoid the potential collision. This means that the Rule-Engine must decide a target location where the autonomous vehicle can safely reach and stand according to the laws of physics. Unfortunately, if Ego must navigate in multiple problem sets such as the pink area of Problem 6 and the yellow area of Problem 7, this mean that the rules must consistently select the same target location to avoid jerkiness in the sequence of behaviours. The knowledge expert can decide to either create a rule overlapping all these problem sets, or to decompose the rule into a set of specialized rules each applying in only one problem set. While the first option seems more convenient, this solution implies a higher activation frequency since the domain of the rule uses more atoms. On the other hand, the second option can be hard to model since each problem set will require to mathematically derive the proximal bounds involving this behaviour. In our implementation, the location type has proven to be a wise choice since it simplifies this behaviour decomposition.

7.2 Discrepancy Identification

Periodically, the knowledge expert must perform the task of Discrepancy Identification. This task involves confronting the rule-based system with various scenarios in which the knowledge expert looks for inappropriate behaviours of the autonomous vehicle operating with a given set of rules. Depending on the level of integration of the tests executed, the behaviours will be considered unsuitable if they lead the autonomous vehicle towards a situation with high probability to collide with at least one static/dynamic object, if they neglect some rules of the applicable highway code or if they generate uncomfortable situations for the road users. As long as the knowledge expert is unable to identify one of these situations, the system will be optimistically judged to exhaustively cover the operational design domain.

In our experience, we have found that at least four test types are necessary to give a realistic approximation of the rule set range. Figure 7.3 illustrates this classification using a process flow, which suggests that this stage continues as long as the knowledge expert does not identify an inappropriate behaviour. Thus, our development process is enhanced as the knowledge expert detects behaviours that have not been well generalized by the set

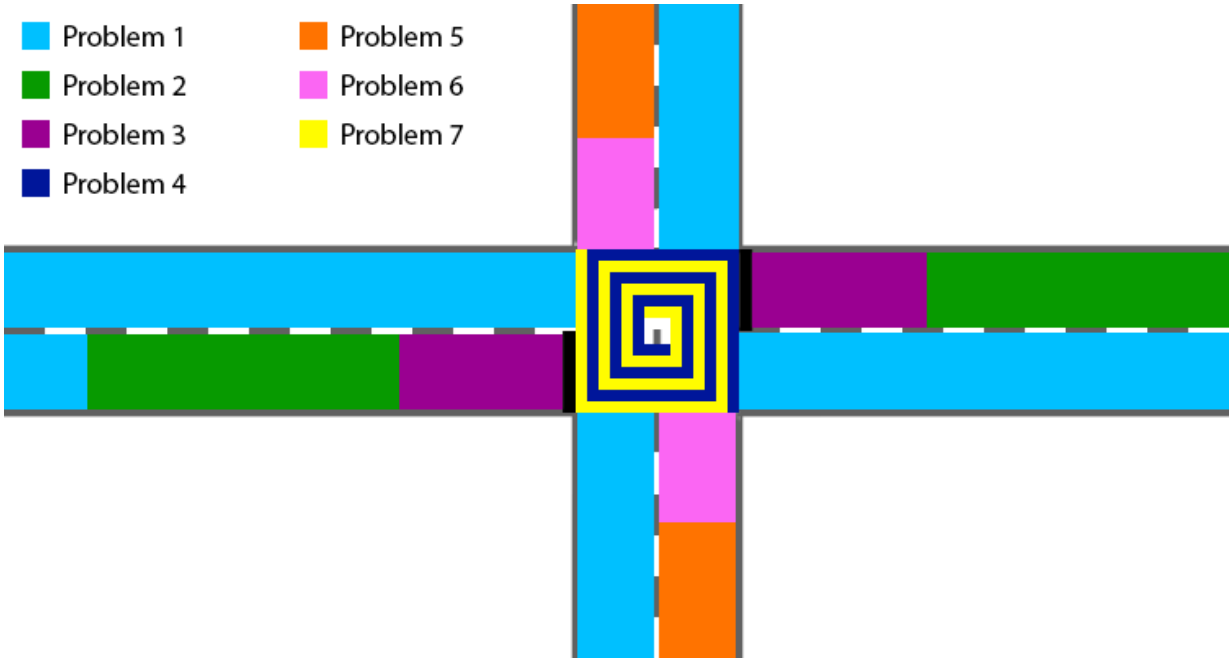


Figure 7.2: Mutual Exclusivity of the Problem Space Decomposition

of rules. This discovery is the entry point to the next stages where the knowledge expert analyzes the misbehaviours.

The first of the four test types performed is the unit tests. These tests only invoke the Rule-Engine by providing the behavioural test suite, $\mathcal{B}$, as an input where the adopted behaviour is compared with the expectation previously hand labelled by the knowledge expert. This test type has the advantage of being fast because the model provided in input is concise and avoids the memoization process, allowing it to ignore the temporal properties of the situations. However, this freedom given to the knowledge expert when declaring the values of each attribute does not enforce the physics of dynamic or static objects and lacks realism in the temporality of the interaction model.

The second test type is the simulation tests. This time, the system is partially integrated with the other components including the tracker, the mission planner, the perception module as well as the local planner. These tests have the advantage of being temporally correlated, more accurately estimating the physic of dynamic or static objects as well as their model of interaction and allow investigating a first layer of integration bugs coming from the assumptions and limitations of each system. However, these tests are done using a completely observable simulation environment and are much slower and expensive to implement since the interaction depends on several systems. During these tests, instead of manually labelling the expected behaviours, the simulator will check whether the autonomous vehicle collides with a static or dynamic object while the knowledge expert will look at the graphical interface to assess if the behaviours of the vehicle controlled by the

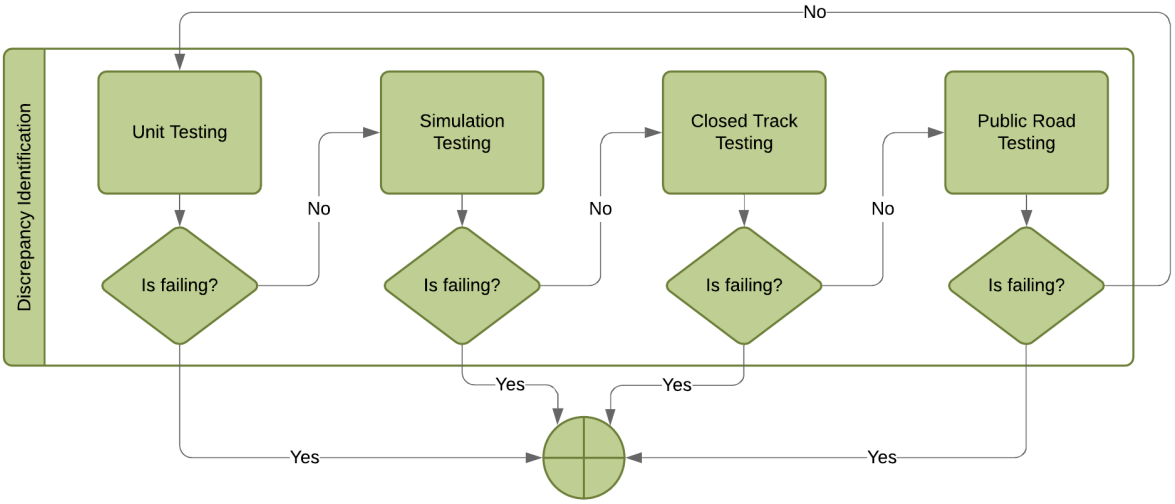


Figure 7.3: Discrepancy Identification Process Flow

rule-based system constitute a sequence of maneuvers expected from a human driver.

The third test type is closed track tests. This time, the system is completely integrated in a vehicle and navigates with the same operating mode as if it would drive in an urban environment. Nevertheless, the environment is completely controlled by the test team, which ensures that safety measures are taken to address the potential behavioural deficiencies that the autonomous vehicle may suffer from. These tests have the advantage of being realistic since the system is fully integrated, estimating the communication needs between the systems, exposing integration bugs as well as minimizing the harmful impact of misbehaving since the test team is aware of the limitations and risks involved in the scenarios. However, these tests are requiring much more human resources and are restricted to evaluate situations where the life of the test personnel is not endangered. Therefore, these tests do not replace those of the previous levels.

Finally, the fourth test type is public road tests. While many driving scenarios have been envisioned and tested during the close track testing, it is almost impossible to cover the range of scenarios that can be found during public driving. Thus, to fully assess the rule base ability to drive in urban areas, several rounds of testing must be performed on public roads. These tests should be executed with a clearly marked autonomous vehicle along with a safety driver and additional safety engineers observing different parts of the autonomous stack. If the tests previously done are well crafted, there should be a limited amount of misbehaviour identified at this level. However, as urban road conditions are difficult to reproduce and because behavioural planning routinely evaluates on a case-by-case basis, these tests are still necessary to assure that the autonomous vehicle is really able to drive in an urban environment.

7.3 Misbehaviour's Diagnosis

Once issues have been identified, the knowledge expert seeks to analyze them. We have seen previously that the data set inputted to the Rule-Engine can be difficult to understand when several dynamic objects are sharing the same environment. Remember that the five pages presenting the data after preprocessing of Chapter 5.5 only involved two dynamic objects and imagine the amount of data to be analyzed when there are more than twenty objects on the scene, such as suggested in Figure 7.4. Given the magnitude of this scenario, we can anticipate that several rules will detect opportunities to undertake maneuvers. Nevertheless, it is also foreseeable that most of the information is superfluous.

Imagine that Figure 7.4 is one of the issues reported in the previous stage. For simplicity, assume that Ego has perfect perception of its environment. Suppose that the

desired maneuver is *overtake* but that the behaviour produced was *emergency-stop*. Using only the information mentioned so far, it can be difficult to explain the logical reasoning behind the *emergency-stop* maneuver. To better understand the implication of each attribute in the decision-making process, the Rule-Engine will first sanitize the environment by only preserving the dynamic objects that has generated the analyzed maneuver. For example, if we would like to understand the *emergency-stop* maneuver, we might notice that the following rule $(\exists v \in vehicles.front | v.orientation.headingTowardsEgo \wedge v.orientation.lateralCenter) \implies (emergency\text{-}stop, v)$ is the only rule to vote for this maneuver type in the issue reported. Thus, by identifying v in this equation, the system can filter the dynamic objects included in this environment by dropping the irrelevant observations such as depicted in Figure 7.5.

In this reduced situation, the cause for the *emergency-stop* maneuver is much easier to deduce. Currently, our autonomous vehicle anticipates that the vehicle turning left can potentially collide with it because the vehicle has a relatively high velocity and its heading is roughly pointing in the opposite direction to Ego. This finding constitutes the foundations of the knowledge required to improve the set of rules.

Finally, the knowledge expert must also ensure that the expected maneuver is produced

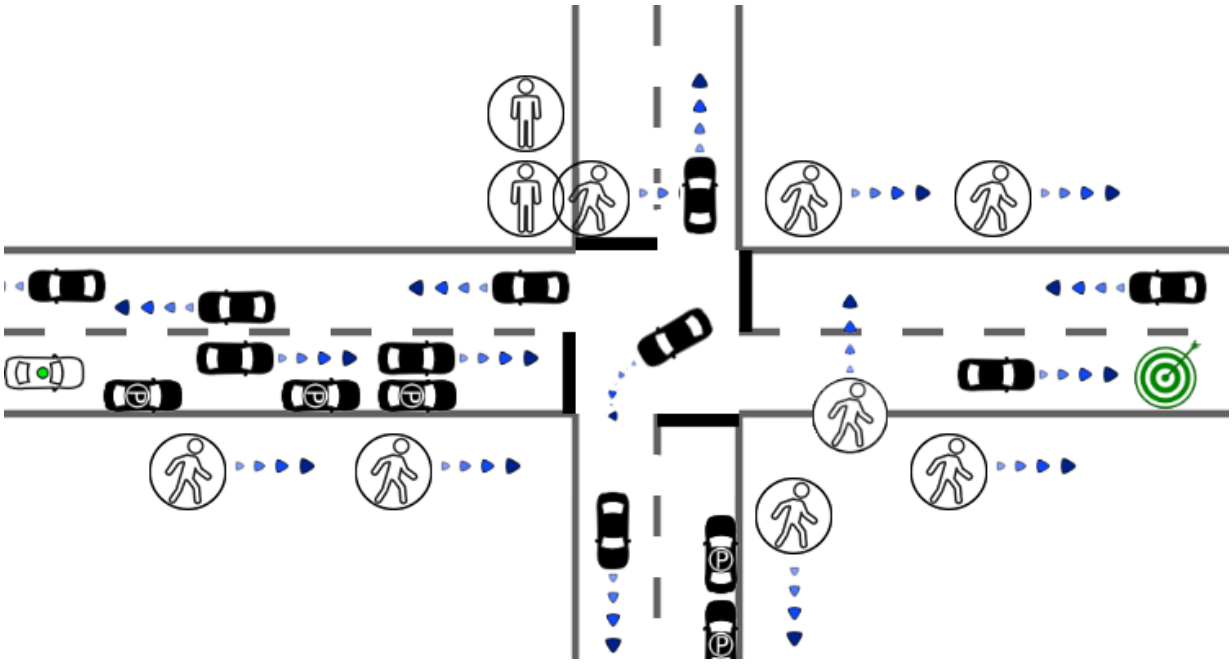


Figure 7.4: [Scenario 4] Unwanted emergency-stop in a busy intersection.

by the Rule-Engine. In the case of Figure 7.4, we can imagine that the rules producing *overtake* are the same as in the third scenario discussed in the previous chapter. Nevertheless, in the case where the desired maneuver would not have been in the maneuvers produced, the knowledge expert must write the draft of a rule to infer the desired maneuver even if this new option is not yet appropriately prioritized.

7.4 Knowledge Acquisition

With the decomposition of the issues, the knowledge expert has several microscopic maneuver-oriented situations each involving a subset of the elements present in the initial environment. These microscopic views are now ready to be compared with those of the behavioural test suite, $\mathcal{B}$, to validate whether the deductions reflect the knowledge acquired or whether there remain gaps. To achieve this, let us remember that in our behavioural test suite, $\mathcal{B}$, we associated each test with a rule number or rule group numbers allowing later to split it into concepts. The knowledge expert therefore uses the rule numbers involved in the decision-making process to determine whether the inferred maneuver is covered in this test space.

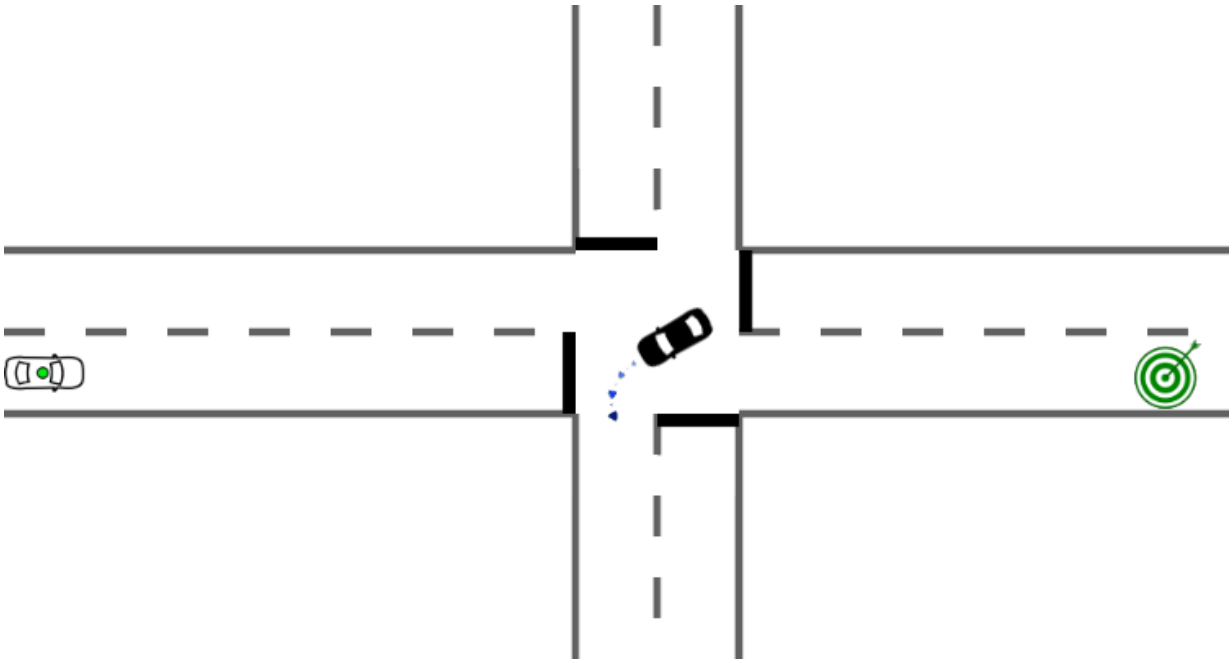


Figure 7.5: [Scenario 4] Emphasis on the emergency-stop maneuver in a busy intersection.

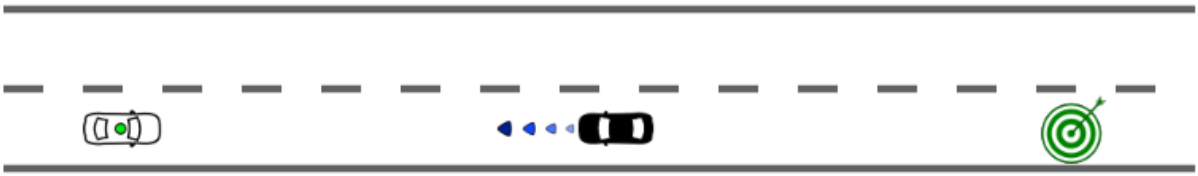


Figure 7.6: [Scenario 4] Closely related emergency-stop behaviour.

In the event that the rules are covered, the knowledge expert has a proof that the issued behaviour is necessary or indistinguishable from another desired behaviour. Otherwise, the knowledge expert must identify the patterns not covered in the microscopic view. If we take the example of the *emergency-stop* maneuver shown in Figure 7.5, we can identify that the targeted rule was initially used to cover the case of Figure 7.6. While studying the desirability of the maneuver produced in these two cases, we can induce that the situation expressed in Figure 7.6 does not generalize well to that of Figure 7.5 since the autonomous vehicle should not be impacted by a vehicle turning in an intersection. Therefore, the knowledge expert can improve the behavioural test suite, $\mathcal{B}$, by adding new examples or by altering the misgeneralizations. In our example, it suffices to specify that the behaviour in the situation described in Figure 7.6 must exclude the vehicles in the physical area of an intersection. This change will remove the *emergency-stop* maneuver from the options considered in the original issue thus allowing the precedence table to select another maneuver.

This process is repeated for each rule involved in the decision-making process. We can see that this approach allows us to design progressively a more complete test suite defining the behaviour that an autonomous vehicle must adopt in a composition of situations. It is, however necessary, to show that the behavioural test suite, $\mathcal{B}$, is consistent, that is, there exists a satisfying set of rules. In other words, the test suite is free from contradictions. An example of contradiction could be "When Ego wants to avoid a collision, it must accelerate" with "When Ego wants to avoid a collision, it must decelerate promptly". Since the premises are the same and the conclusion varies, we see that the behavioural test suite, $\mathcal{B}$, is unsatisfiable and therefore is incorrect.

7.5 Knowledge Engineering

The knowledge engineering stage aims to update the rule set to make it satisfy the behavioural test suite, $\mathcal{B}$, edited during knowledge acquisition. To ensure that generalization is adequate, the knowledge expert has to draft the required rules before letting the system deduce the missing predicates. These drafts mainly stem from the fact that the behavioural test suite, $\mathcal{B}$, consists of representative cases and therefore lacks a large bank of examples which help to build an adequate generalization of the predicates required in the updated rules. However, when these drafts are constructed, the system can automatically fix their unsatisfiability by introducing a minimum of new predicates in the rules' clauses.

In relation to the metrics presented in Chapter 6.2, the coupling metric defined as the sum of the number of greediness, redundancies, rejections and supports is used to approximate the maintainability of the rule sets. The algorithm proposed in this thesis implements a variant of Hill-Climbing that minimizes the number of new attributes to introduce in the rules. Hill-Climbing starts with the implemented rules supporting the previous version of the operational design domain and add one attribute per step in a rule causing a behavioural defect. The algorithm continues to derive new attributes until it satisfies the test suite and that adding a new attribute in one of the previously edited rules does not improve the efficiency/maintainability trade-offs. Hill-Climbing is a reasonable choice under the assumption that the implemented rules supporting the previous version of the operational design domain should be correlated with the rules required by the updated version. For this reason, we can start the algorithm with the current rule set instead of starting with an empty knowledge base.

Algorithm 1 presents the skeleton of the knowledge engineering task. To ensure the requirement of introducing a minimum number of predicates, the for loop (Algorithm 1, lines 6-22) derives from a set of rules a new version of one of its rules where a single new predicate is introduced in one of the rule disjuncts. To decide which disjunct to choose, the algorithm relies on an error signal corresponding to the existence of an unsatisfied test case. When there exists an unsatisfied test case (Algorithm 1, lines 9-18), the goal of the algorithm is to satisfy this test case. When there does not exist an unsatisfied test case (Algorithm 1, lines 19-21), the goal of the algorithm is to improve the trade-off between efficiency and maintainability.

Let's first focus on the existence of an unsatisfied test case. In such event, one of the failing test cases is randomly selected (Algorithm 1, line 9). Because we labelled each test case with a rule number in our behavioural test suite, we already know which rule should ensure that behaviour. To decide the procedure that will lead the algorithm closer to satisfy

Algorithm 1 Maneuver Rule Set Improvement

Input:

B, the behavioural test suite
R, the set of rules to improve
$\mathcal{W}$, the world abstraction atomic model

Output: An improved set of rules, R', satisfying the behavioural test suite, B

1: $previousRuleSets \leftarrow \varnothing$
2: $candidateRuleSets \leftarrow \{R\}$
3: **while** $previousRuleSets \neq candidateRuleSets$ **do**
4: $\quad previousRuleSets \leftarrow candidateRuleSets$
5: $\quad candidateRuleSets \leftarrow \varnothing$
6: $\quad$ **for** $R' \in previousRuleSets$ **do**
7: $\quad\quad \epsilon \leftarrow \{(W, \_, M^*, C^*) \in B | R'(W) \neq (\_, M^*, C^*)\}$
8: $\quad\quad$ **if** $\epsilon \neq \varnothing$ **then**
9: $\quad\quad\quad \delta \leftarrow selectARandomTest(\epsilon)$
10: $\quad\quad\quad (W, R_i, \_, \_) \leftarrow \delta$
11: $\quad\quad\quad (\mathcal{M}, \_, \_) \leftarrow R'(W)$
12: $\quad\quad\quad$ **if** $R_i \notin \mathcal{M}$ **then**
13: $\quad\quad\quad\quad pivot \leftarrow getPivotPredicate(W)$
14: $\quad\quad\quad\quad newRuleSet \leftarrow (R' \setminus R_i) \cup (R_i \vee pivot)$
15: $\quad\quad\quad\quad candidateRuleSets \leftarrow candidateRuleSets \cup newRuleSet$
16: $\quad\quad\quad$ **else**
17: $\quad\quad\quad\quad candidateRuleSets \leftarrow candidateRuleSets \cup StrengthenARule(\mathcal{W}, \delta, R')$
18: $\quad\quad\quad$ **end if**
19: $\quad\quad$ **else**
20: $\quad\quad\quad candidateRuleSets \leftarrow candidateRuleSets \cup R' \cup OptimizeRules(\mathcal{W}, R')$
21: $\quad\quad$ **end if**
22: $\quad$ **end for**
23: $\quad candidateRuleSets \leftarrow keepKBestRuleSets(candidateRuleSets, k)$
24: **end while**
25: **return** $keepKBestRuleSets(candidateRuleSets, 1)$

Algorithm 2 Strengthen A Rule

Input:

δ, a failing test case

R', the set of rules to improve

$\mathcal{W}$, the world abstraction atomic model

Output: Many new candidate rule sets

$newRuleSets \leftarrow \varnothing$
$(W, \_, M^*, C^*) \leftarrow \delta$
$(\mathcal{M}, \_, \_) \leftarrow R'(W)$
$R_j \leftarrow selectARandomGreedyRule(\mathcal{M}, M^*, C^*)$
for $disjunct \in matchingDisjuncts(W, R_j)$ **do**
    for $predicate \in \mathcal{W} \setminus (W \cup disjunct)$ **do**
        $newRule \leftarrow (R_j \setminus disjunct) \vee disjunct \wedge predicate$
        $newRuleSet \leftarrow (R' \setminus R_j) \cup newRule$
        $newRuleSets \leftarrow newRuleSets \cup newRuleSet$
    end for
end for
return $newRuleSets$

Algorithm 3 Optimize Rules

Input:

R', the set of rules to improve

$\mathcal{W}$, the world abstraction atomic model

Output: Many new candidate rule sets

$newRuleSets \leftarrow \varnothing$
for $R_k \in findUpdatedRules(R')$ **do**
    for $disjunct \in findUpdatedDisjuncts(R_k)$ **do**
        for $predicate \in \mathcal{W} \setminus disjunct$ **do**
            $newRule \leftarrow (R_k \setminus disjunct) \vee disjunct \wedge predicate$
            $newRuleSet \leftarrow (R' \setminus R_k) \cup newRule$
            $newRuleSets \leftarrow newRuleSets \cup newRuleSet$
        end for
    end for
end for
return $newRuleSets$

the selected test case, we first need to understand if the behaviour is at least derived. To do so, we look in the potential maneuvers (before the precedence table filtering), if the rule number appears in the potential behaviours. When it is not the case, then we want to force the rule to match the assessed situation by creating a new disjunct using only the pivot predicate (Algorithm 1, lines 13-15). This pivot could be directly extracted from the assessed test case, therefore providing a guarantee that the behaviour of this rule will appear in the list of potential behaviours. At this time, if the new rule set already satisfies the behavioural test suite, it would mean that there exists an abstract location where only one behaviour can be performed. If this is expected, than the algorithm terminates and the rule set is returned, but most of the time, this situation should be unrealistic. Thus, it is expected that the newly created rule will cause many new failing test cases.

Increasing the number of failing test cases do not mean that the rule set is worst. It simply means that there is more work to do before reaching a better set of rules. To complete the rule we have previously created, the algorithm falls in the case where it can find a failing test case where the intended rule number appears in the potential behaviours, but was not retained (Algorithm 1, lines 16-18). This implies that at least one other rule acts greedily over the intended behaviour. In such event, the algorithm selects one of the greedy rules (Algorithm 2, line 4) and add a new predicate in one of the disjunct matching the failing test case (Algorithm 2, lines 5-11). By doing this update, the disjunct will eventually stop matching the assessed behaviour, but it may require more than one iteration of the main loop (Algorithm 1, lines 3-24). As you can guess, the choice of predicate is subject to influence the behavioural test suite satisfaction. By exhaustively trying all the predicates not included in the failing test case and not already in the selected disjunct (Algorithm 2, lines 6-10), the algorithm uses the new number of failing test cases as a signal to decide the k most suited predicates (Algorithm 1, line 23). We retain only the best k predicates (which can be referred as candidate rule sets) since this algorithm is exponential and thus, we want to restrict our exploration with the most promising rule sets.

Eventually, the goal of satisfying the test suite is met (Algorithm 1, lines 9-18) and thus, we are left with k satisfying rule sets. From there, we do not want to terminate yet since the overall performance of our rule sets might be improvable. Thus, we continue to introduce new predicates by focussing only on the updated disjuncts of the updated rules (Algorithm 3, lines 2-10). The goal of this expansion is to decide if mutual exclusivity achieves a better performance. When possible, the algorithm attempts to reach mutual exclusivity by adding one predicate at the time. The new predicate decreases the efficiency since each time the attribute value changes, the rule is activated. Meanwhile, the predicate improves maintainability since the condition for the rule to be true is restricted. At some point,

adding new predicates will only decrease the performance, and thus, $keepKBestRuleSets$ (Algorithm 1, line 23) will return the candidate rule sets corresponding exactly to those of the previous iteration. This stability is the termination criterion of our algorithm (Algorithm 1, line 3) that can now return the best of the candidate rule sets (Algorithm 1, line 25) as our improved version of the inputted rule set.

Figures 7.5 and 7.6 constitute an example of a trivial rule improvement that can be performed by this algorithm. Assuming that the world reprensentation entails in Figure 7.5 is added in a test case where the intended behaviour is *track-speed* at 50km/h, the algorithm starts with this test case as the only failing test case. It exhaustively derives rule sets with a new predicate in the only disjunct of the *emergency-stop* rule acting greedily over the *track-speed* maneuver and discover that **ego.location.onIntersection = false** can satisfy the test suite in one step. Thereafter, the algorithm continues expanding the only rule and disjunct it has previously updated, and discover that further expansions decrease the metrics. Thus, the algorithm reach stability and return the improved set of rules.

The example studied in this section was very abstract because we wanted to emphasize the utility of the test suite and the explainability mechanism. Let's now consider a toy example where we can concretely see what the algorithm is doing. Let's assume that the available propositions in the World Abstraction Atomic Model, $\mathcal{W}$, are $(A)ego.location.atIntersectionRegulatedByStopSign$, $(B)ego.hasPrecedence$ and $(C)ego.hasPerformedAFullStop$.

For the convenience of this demonstration, let's assume that the precedence table is represented by the following set, $\{track\text{-}speed, stop, decelerate\text{-}to\text{-}halt\}$, such that *track-speed* has the lowest priority.

Let's also suppose that the behavioural test suite is defined as follows.

$$
\begin{aligned}
B_1 :=& (A \wedge \neg B \wedge \neg C) \implies (decelerate\text{-}to\text{-}halt, stop\text{-}line) \\
B_2 :=& (A \wedge \ B \wedge \ C) \implies (track\text{-}speed, 50km/h) \\
B_3 :=& (A \wedge \ B \wedge \neg C) \implies (stop, time\text{-}buffer)
\end{aligned}
$$

Also suppose that we start the algorithm with the following rule set.

$$
\begin{aligned}
R_1 :=& (A \wedge \neg B) \implies (\textit{decelerate-to-halt}, \textit{stop-line}) \\
R_2 :=& (A \wedge \ \ B) \implies (\textit{track-speed}, 50km/h) \\
R_3 :=& (\qquad\quad) \implies (\textit{stop}, \textit{time-buffer})
\end{aligned}
$$

For simplicity, let's assume that each B_i associated rule number is R_i. In addition, let's assume that we have represented, $B_i := W_i \implies (M_i^*, C_i^*)$. Because we wanted to emphasize the relationships, we did not use the tuple notation represented earlier, but $B_i = (W_i, R_i, M_i^*, C_i^*)$ can be deduced easily.

In the current setup, you can imagine that the first two test cases were supported by the first two rules and a third test case was designed during the knowledge acquisition when the knowledge expert realized that Ego was performing stop and go. Thus, the knowledge expert added a new rule with an empty condition producing the expected behaviour. Since the clause is empty, the rule can never evaluate to true and thus, is currently useless.

In the following algorithm trace, we use $k = 1$ as the maximum number of rule sets to preserve. In practice, this number performs poorly when the rules disjuncts require introducing many new predicates before reaching a satisfying state and when the number of predicates in the World Abstraction Atomic Model is large. We use $k = 1$ because it simplifies the trace and does not affect the generation of the improved rule set.

During the first iteration, B_1 and B_2 are satisfied leaving B_3 as the only failing test case. Therefore, the algorithm enters the clause body where it wants to satisfy the test suite (Algorithm 1, lines 9-18). The $\mathcal{M}$ of this test case contains $\{R_2 \implies (\textit{track-speed}, 50km/h)\}$, thus R_3 is not a member of the potential behaviours. The algorithm then extracts the pivot predicate that, in this case, is $(A)\textit{ego.location.atIntersectionRegulatedByStopSign}$, and creates the following new version of R_3.

$$
R_3 := (A) \implies (\textit{stop}, \textit{time-buffer})
$$

Since this rule set is currently the only option that the algorithm has generated so far, the $\textit{keepKBestRuleSets}$ function (Algorithm 1, line 23) does nothing and we are entering the second iteration since the rule set is not identical to the rule set we started with. This time, given the precedence of the maneuvers, B_1 and B_3 are satisfied and thus, the algorithm selects B_2 has the failing test case. In this assessment, the $\mathcal{M}$ corresponds to $\{R_2 \implies (\textit{track-speed}, 50km/h), R_3 \implies (\textit{stop}, \textit{time-buffer})\}$. As we can see, the intended behaviour is in the potential behaviour list and thus, the algorithm falls in the

clause body where it wants to force R_3 to stop matching in the environment representation. Since in B_2, the predicates B and C are used, there is only $\neg B$ and $\neg C$ for potential predicate that the algorithm can use. Thus, it generates the following two versions of R_3.

$$R_3' := (A \wedge \neg B) \implies (stop, time\text{-}buffer)$$
$$R_3'' := (A \wedge \neg C) \implies (stop, time\text{-}buffer)$$

According to *keepKBestRuleSets* with $k = 1$, the algorithm must filter out one of these alternatives. Since R_3' leads to one failing test case (B_3) and that R_3'' satisfies the test suite, then R_3'' is selected. Again, the produced rule set is different from the previous step leading to another iteration of the main loop. This time, since the behavioural test suite is satisfied, the algorithm goes over each updated rule and disjunct and continue to expand the clause generating the following two new version of R_3.

$$R_3''' := (A \wedge \neg C \wedge \ B) \implies (stop, time\text{-}buffer)$$
$$R_3'''' := (A \wedge \neg C \wedge \neg B) \implies (stop, time\text{-}buffer)$$

Thus, the algorithm currently has three candidate rule sets. Respectively, $\{R_1, R_2, R_3''\}$, $\{R_1, R_2, R_3'''\}$, and $\{R_1, R_2, R_3''''\}$. Naturally, R_3''' will be filtered out since it does not satisfy the behavioural test suite. Whether R_3'' or R_3'''' is more desirable cannot currently be determined in our example. R_3'' is more efficient since it relies on fewer parameters while R_3'''' is more maintainable because it is mutually exclusive with R_2. If R_3'' is retained, then the algorithm will terminate since this is the same rule set as in the previous iteration. If R_3'''' is retained, the algorithm will continue to expand, but since there is no more predicate available, it generates the same rule set and terminates.

In any case, we have depicted that our algorithm can improve the inputted rule set such that the new rule set satisfies the behavioural test suite. In a more complex scenario, the branching factor of this algorithm is expected to be exponential, but the pruning done by *keepKBestRuleSets* keeps the algorithm manageable.

Chapter 8

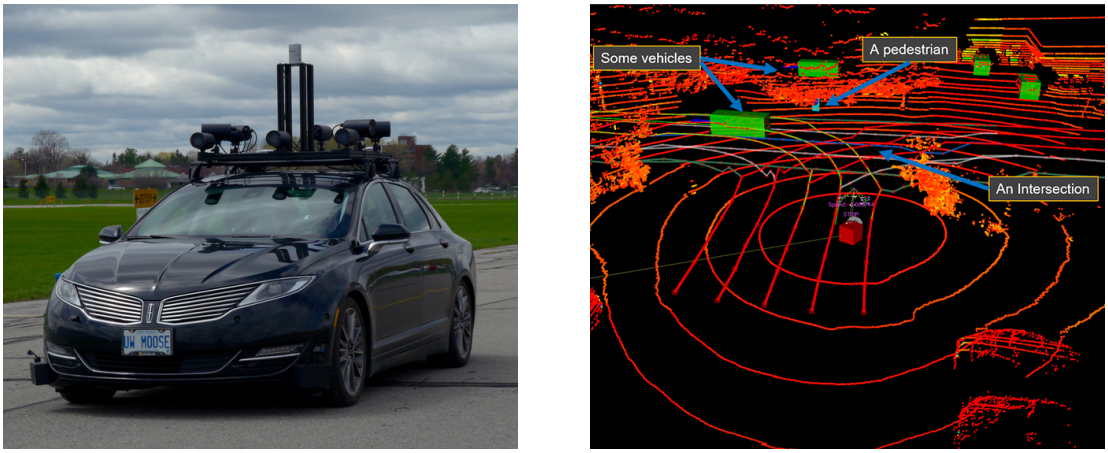
Empirical Evaluation

The proposed architecture and methodology were evaluated through a practical application in the autonomous vehicle of the University of Waterloo depicted in Figure 8.1. This methodology was proven to be effective in driving on populated public roads in the city of Waterloo [12].

The public drive was performed on a network of two-lane commercial roads constrained by four-way intersections, and t-intersections with precedence varying between the ego vehicle as well as other dynamic objects. Due to the planned mission through the road network, the ego vehicle had, for instance, to effectively handle unprotected left-hand turns. The roads selected for the public drive had a myriad of parked vehicles impeding Ego's progress. In that sense, the rules were implemented to ensure that Ego was able to safely overtake parked vehicles by encroaching on the lane of oncoming traffic similarly to a human-controlled vehicle.

8.1 Implementation

The machine interface used to abstract and discretize the data coming from the various sensors acts as a middleware between the Rule-Engine and the Local Planner. The sensors used include cameras, lidar, dashboard and embedded computers within the autonomous vehicle. The machine interface is implemented in C++ and depends on the Kinetic version of the Robot Operating System (ROS). Its data exchanges are mainly done using ROS topics except for the Rule-Engine which uses web sockets to transfer JSON streams.



(a) Autonomoose (b) Lidar Point Cloud Map

Figure 8.1: Autonomoose, the University of Waterloo autonomous vehicle used to perform the 100 kilometres of autonomous urban driving with the rule-based system behaviour planner.

The Rule-Engine has been developed following the functional principles of a backend web application. It runs in a terminal where it opens intranet protocols that serve JSON streams. The expert system is intended to be used in conjunction with an interpreter such as Google's V8 engine which can be used on any operating system. The rule-based system was developed following the semantic standard proposals of ECMA Script 2016-2017 [13] and uses polyfills to ensure uniform behaviours regardless of the interpreter used. The application takes advantage of the destructuring and restructuring techniques popularized by functional languages. The system data is stored via the Reducer View Action (RVA) [14] design pattern using the Redux library.

8.2 Knowledge Base

In the context of our experience, in addition to solving the seven problems presented in Chapter 7.1, we have also developed the set of data integrity rules introduced in Chapter 4.1. This set of rules aims primarily to facilitate the integration of the Rule-Engine with the other components of the autonomous stack since the mechanisms of *Misbehaviour Diagnosis* presented in Chapter 7.3 can be used to produce diagnostics enabling greater cohesion and promoting autonomy between teams that can benefit from analysis and debugging

capabilities without solely relying on the knowledge expert.

During the numerous demonstrations of autonomous urban driving in the region of Waterloo, the modified Lincoln MKZ depicted in Figure 8.1 was using a set of 75 rules whose distribution is shown in Figure 8.2. From that figure, we can see that 52% of the rules enforce an *emergency-stop* maneuver to delimit the operational design domain. However, 30 of these rules cover integrity issues which can be removed when the integration is complete, and the incoming model is considered well-formed since it will lead to the same sequence of outputted behaviours. Thus, we can consider that an autonomous vehicle can drive freely in an urban environment with as few as 45 rules. This practical demonstration shows the feasibility of expert systems to solve problems involving a significant number of uncertainties both in the integration limitations of a series of logical components and in the quantification of inputted noise.

Moreover, Figure 8.3 provides an overview of the number of disjuncts used in the maneuver rules. From the graphic, we can see that most of the rules contains less then three disjuncts which means that the attributes used in the clauses of these rules do not significantly overlap. Furthermore, the graphic also depicts that one rule has 21 disjuncts which is far away from the other rules. This rule entails the concept of overtaking in which there is many clauses all relying on the same subset of attributes. Therefore, the necessity to partition this rule remains debatable from the efficiency perspective.

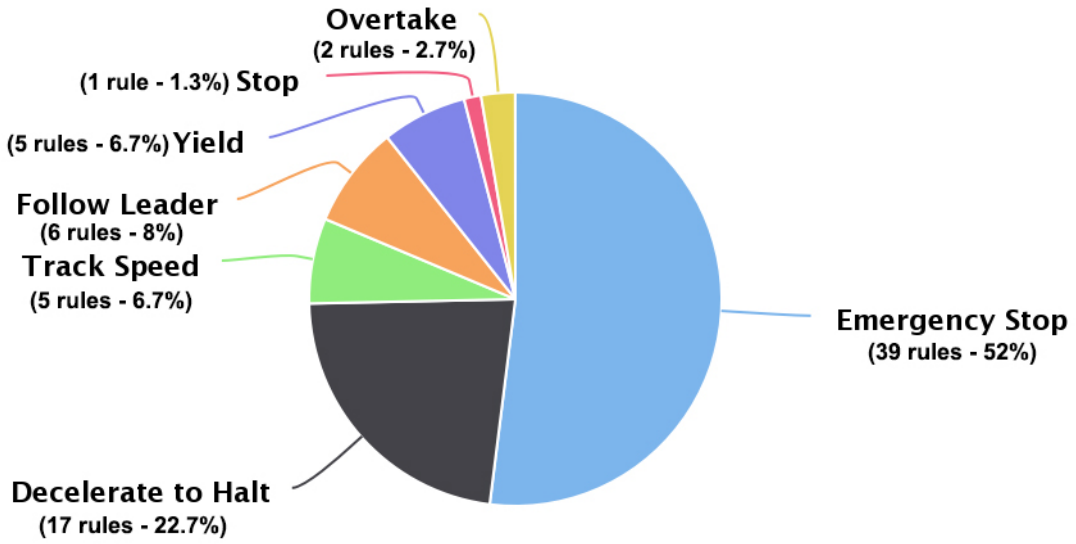


Figure 8.2: Rule distributions used during the public drive

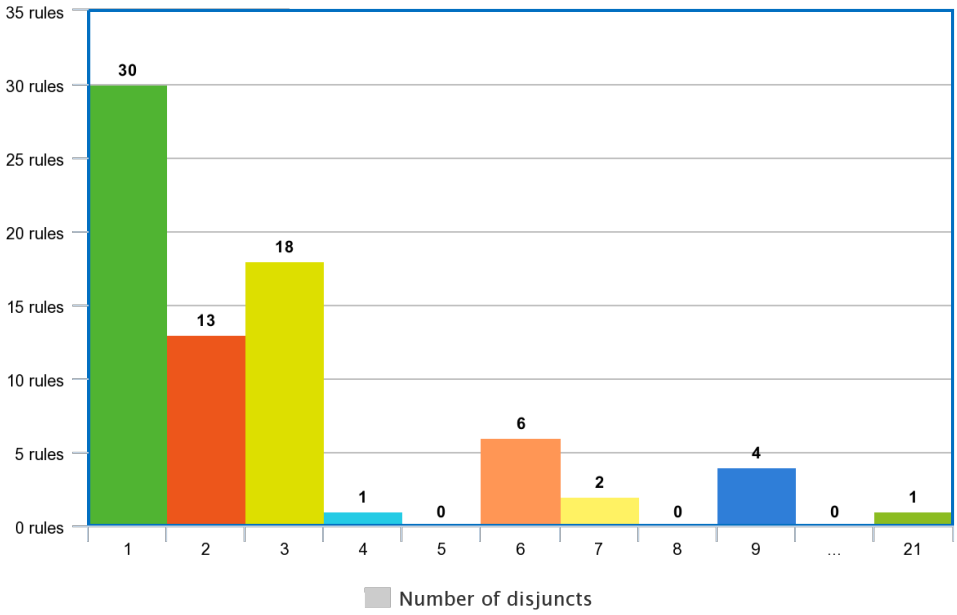


Figure 8.3: Number of disjuncts per rule in the maneuver rule set

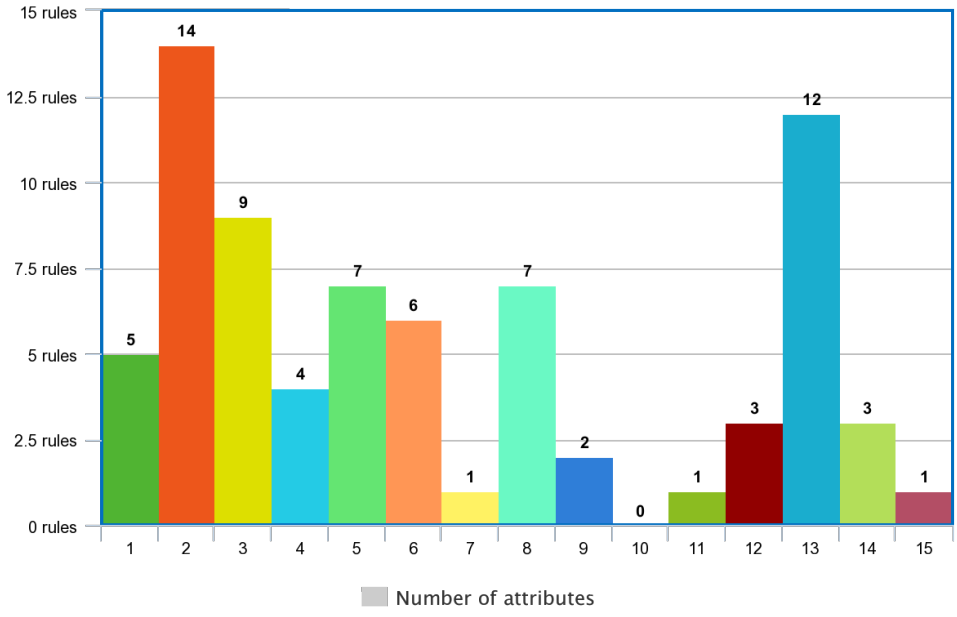


Figure 8.4: Number of attributes per rule in the maneuver rule set

Finally, Figure 8.4 provides an overview of number of attributes used in the maneuver rules. From the graphic, we can mainly observe two distributions. The first one have its median at 3 attributes and the second have its median at 13 attributes. The former are usually rules involving only Ego's with potentially a leading vehicle while the latter are usually rules involving interaction with numerous dynamic objects such as nearby an intersection or during an overtaking maneuver.

8.3 Timed Memoization Efficiency

The memoization scheme introduced in Chapter 4.3 was tested through 110 kilometres of public driving in the areas of Bathurst Drive, Colby Drive, and Fire Tower Road. The activity was monitored while the vehicle was in full autonomy in 31 different scenarios involving the problems $P1$, ..., $P7$. On average, the memoization scheme reduced the activation of more than two thirds of the 75 rules.

Figure 8.5 summarizes the system activity of one of these scenarios, highlighting several advantages of this scheme. In the depicted scenario, the autonomous vehicle is required to perform the following maneuvers: (i) follow a human-controlled vehicle, (ii) stop at an intersection, (iii) wait for precedence at the stop line, and (iv) handle jaywalking pedestrians. In the figure, the term *vote* means that a rule has been activated and produced a behaviour to add in the memory whereas *revoke* means that a rule has been activated but did not produced a behaviour requiring the system to wipe the propositions associated with this rule in its memory. Finally, the term *total activity* is simply the sum of the number of *votes* and *revokes*.

As supported by the Figure 8.5, the nature this scheme results in peaks of high activity followed by an extended period of low activity in which the system can computationally recover. The periodic damping cycles support the idea that it would be possible to significantly increase the number of requests and/or number of rules to evaluate before observing a loss of performance.

Figure 8.5 also depicts that the average number of activated rules voting at any time step is less than one which means that the constraints' rules execution is mostly trivial and intrinsically demonstrating the low coupling as well as the memoization schemes ability to avoid recomputation.

Finally, around the ticks 200 to 260, a strong presence of repeated peaks can be observed. This event happens when the environment perceived is noisy and the discrete parameters oscillate between multiple values. In these situations, the safety driver must

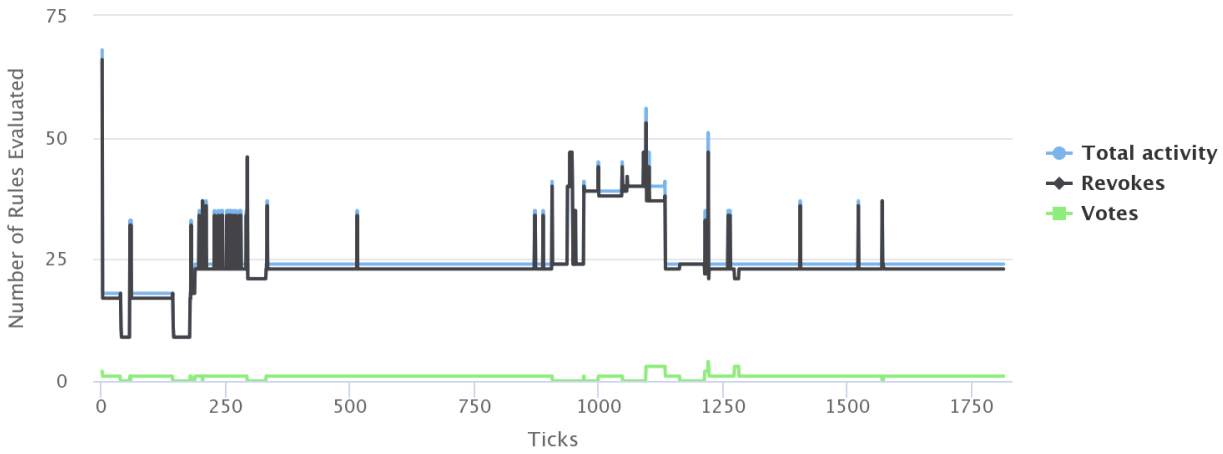


Figure 8.5: Memoization effect on a scenario involving ego following a leading vehicle through an intersection with random encounter on the opposite lane.

be warned since the proposals submitted to the Rule-Engine does not allow to choose a consistent sequence of behaviours to initiate. Thus, the memoization scheme can also be used to identify inconsistencies and prevent dangerous situations.

8.4 Timeless Memoization Efficiency

One might be interested in knowing the effect of the memoization scheme if the hypothesis of temporal correlation amongst a sequence of observations is missing. Using the behavioural test suite, we can simulate this lack of correlation since these tests are performed by resetting the value of all attributes between each query. Thus, in each assessment of the behavioural test suite, the temporal difference depicts the average number of rules required to evaluate an observation.

In Figure 8.6, we executed the set of rules used during the public drive on the 685 uncorrelated tests constituting the behavioural test suite. The figure shows that, on average, the system must evaluate two thirds of its rules and cannot benefit from a low activity period which directly affects the scaleability of the system. With the reported data, it can be said that the memoization scheme and the temporal difference operations reduce the run time by a third. This performance reflect the ability of the system to react in a critical situation such as when perception is so noisy that the dynamic objects are flicking in the inputted environment. More generally, we can anticipate that increasing the fre-

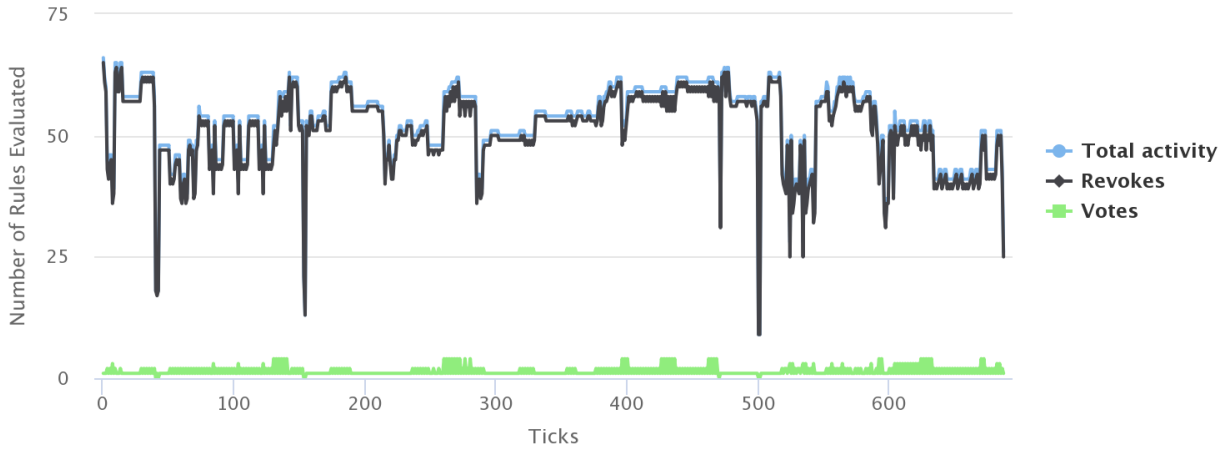


Figure 8.6: Activation function on the behavioural test suite using the same rule set as during the public drive.

quency of the system driving on public roads will also increase the correlation amongst the observations, thus improving the effect of the memoization.

8.5 Qualitative Analysis

To prove the viability of the expert system within its operational design domain, the autonomous vehicle was tested in a 110 kilometre drive on public roads in full autonomy. During this test it required a total of 58 interventions by the safety driver with an average time of over 5 minutes and 30 seconds between interventions. These disengagements were due to three main contributing factors: the driving scenario encountered outside of the operational design domain; the conservative design of the system resulting in a deadlock at some intersections due to perceptual noise corrupting behaviour interpretation of dynamic objects on the road; the misdetections of dynamic objects in the environment. During the public drive, the Rule-Engine deployed in autonomous vehicle was able to serve queries up to a speed of 300 Hertz. Knowing that the average speed of a component on this stack runs at 10 Hertz, the rule-based system is far from being the bottleneck of this integration. While we acknowledge that the empirical data is unable to provide guaranties of rule quality, based on the results obtained, we have shown the suitability of the rule-based architecture within an actual autonomous vehicle and its capability to support incremental development expanding the system's operational design domain.

Chapter 9

Method Discussion

This chapter discusses the fulfillment of the design objectives presented in the Chapter 2 of this thesis. In particular, we explain how we improve the interpretability of the inferential process, how we maximize the coverage of the behavioural test suite while minimizing the number of situations to model, how we measure the maintainability of the domain knowledge as well as how we approximate the system performance outside a virtual environment, then finally, how we design a synthesis process to elicit the missing requirements as well as to adjust the rule base when deficiencies are detected.

9.1 Expert System

We briefly discuss the how our expert system design addresses interpretability, efficiency, and maintainability.

There are several ways to formalize and structure knowledge. Taking the example of databases, one could wonder the advantages of object versus relational structures. On the one hand, the object structures have the advantage of being easier to comprehend. They allow the knowledge expert to interpret the data more efficiently since relevant information is self contained inside each entity. On the other hand, relational structures avoid redundancies in the model to the detriment of a join requirement when accessing data. Relational structures make the repeated updating more efficient since the data is processed atomically.

For example, imagine that our model incorporates, for each dynamic object, a notion of relative location of dynamic and static objects within the environment. Thus, each

object is used differently by the various entities. When one of these objects is updated, it is therefore necessary to propagate this information in each entity using it. These updates are expensive because they require synchronisation in several collections and are subject to cascade effects where an update can cause another. This procedure complicates the development of a database since it becomes difficult to identify which portion of a cascading update causes a discrepancy, or sometimes, even an infinite update cycle.

The problem described above also affects ordered knowledge bases. In these, the knowledge derived by a rule is immediately accessible by the rules succeeding it. This forward chaining allows building generic-to-specialized rules since the hierarchy enriches the knowledge during the execution. However, this expansion hampers maintainability of the system since the knowledge expert must consider the permutations among the rules when updating the knowledge base. Further, some ordered rule-based systems rerun previous rules ensuring that the facts discovered later also benefit from the previous forward chaining. This loop corresponds to the cascading effect described for the databases and entails the same identification complexity.

As part of the design of a behaviour planner for an autonomous vehicle, it is necessary that the reactivity of the expert system should be the main focus since it must respond to a continuous information flow. In this perspective, a relational structure is more suited to the demand. Nevertheless, a competing need is the interpretability of the inferential process.

To meet these competing demands, our approach is to model a volatile database somewhere between the object and the relational structure. To achieve this, the input data is flattened using the Boyce Codd Normal Form (BCNF), but we allow entities to have nested properties providing syntactic sugar that helps interpretability. In the knowledge base, we want to preserve the beneficial property of forward chaining, which allows us to design generic to specialized rules. However, to avoid the problems of permutations and cascade effects, we suggest developing a unordered rule set hierarchy. In this setup, members of the same rule set all have access to the same input data, but the derived knowledge is combined.

The inferential process suggested is subject to the production of inconsistent knowledge. In contrast with the ordered inferential process, rule ordering cannot be used to express rule priority. Thus, taking the example of autonomous vehicles, a rule could declare that it is better to maintain the speed, while another rule can claim the necessity to decelerate. Clearly, these objectives are competing and cannot be achieved in parallel. Thus, to restore the consistency after executing a rule set, we propose implementing a filtering policy that, in our system, consist of taking the most conservative action. This policy retains only

a portion of the derived knowledge and thus narrows the number of properties to be considered by the next rule sets.

Since some of the derived knowledge is discarded, it is better to reduce the computation costs of the inferential process by minimizing the number of rules to evaluate. To do so, we used a memoization technique where the temporal difference between two iterations identifies the altered input propositions in this time interval. Thus, we can preserve the output of each rule as long as no proposition is found in the intersection between the temporal difference and the propositions used in the clauses of the rule.

Finally, a last property that favors the traceability, and thus interpretability, of the inferential process is the identification of matching clauses, commonly called backward chaining. When a knowledge expert models a knowledge base, it is natural to elaborate the rules by enumerating the premises required to deduce a behaviour. Nevertheless, when trying to understand why the system behaves in a specific way, several premises are associated to the same conclusion. To identify the causality of a behaviour, we dynamically rewrite a rule set by creating a rule for each disjunct in the rule set disjunctive normal form (DNF). Thus, by partitioning the conjunctions of this normal form, the system will be able to identify the premises involved in the decision-making process without restricting the expressiveness of the rules.

9.2 Behavioural Test Suite

We now summarize how the test-driven approach and the proposed form of the behavioral test suite aid maintainability and correctness.

The problem of behavioural planning is deemed to be defined on a case-by-case basis [9]. Given the extent of the scenarios that the autonomous vehicle may encounter, it is natural to want to demonstrate that the driving policy induced by the planner is safe and adapted to each situation. While we recognize the importance of this goal, we believe that this demonstration should be primarily covered in system-level simulation testing. A key objective in the rule development is not only to correctly cover the tested situations, but to avoid producing very narrowly focused rules. Instead, the focus should be on capturing the representative corner cases that delineate the domain in which a rule holds. Recall that we chose to use a rule-based system for its interpretability property. Thus, it would be counterproductive to overwhelm it with narrowly-focused rules, each representing very specific examples. On the contrary, the knowledge expert wants to use the traceability of backward chaining to understand discrepancies in the inductive process and therefore,

the usage of such representatives naturally filters the amount of information that must be reviewed during knowledge acquisition.

The discrepancies to identify are (i) the cases that are not well generalized, as well as (ii) the contradictory specifications that demonstrate an incompleteness of the model used in the inductive process. To facilitate the understanding of the discrepancies in the inductive process, we first want to achieve temporal independence. Given that behaviours of an autonomous vehicle are strongly influenced by an action/reaction system with the users in the environment, the driving policy is time sensitive. Therefore, to reduce effort in unit testing, we would like to assess the behaviours of the autonomous vehicle without having to describe a realistic sequence of interaction which require tuning the attributes given as input. By taking a snapshot of the temporal attributes and allowing the expert-system to override its memory with this snapshot, we create a time-free extension of the input model enhancing the testing process. Therefore, the test suite can use the time-free model to describe standalone assessment which can run in parallel and require fewer attribute tuning.

This time-free implementation makes sense when we consider our main goal of minimizing the number of situations included in the behavioural test suite. This minimization objective is explained by the fact that we will later compare the misbehaviour of the autonomous vehicle with the representative cases to elicit the missing requirements. Thus, we want to avoid biasing the analysis and design with irrelevant attributes. On the contrary, we believe that a test only needs to specify the most important attributes, even if the specification does not describe a real concrete situation. The objective of this model is to tightly cover the disjunctive normal form (DNF) of the rules matching the situation. Thus, in the context of our experience, we judge that our behavioural test suite is effective if (i) it exhaustively represents the disjoint minimum requirements of each rule, (ii) it approximates the range of the feasible join amongst the rules, and if (iii) it describes understandable situations for the knowledge expert.

The remaining properties desired for the behavioural test suite are primarily to ensure the maintainability of the rule sets. On the one hand, we want to define a pivot attribute that is used in each rule. This attribute allows us to manage the complexity of the interleaving amongst the rules since its range of values partitions them, thereby exposing the feasible joins. As part of our implementation, the retained pivot attribute is the road geometry, since it is a natural pivot decomposing the problem of autonomous driving. On the other hand, we also want to associate the rules with one or more representatives. This extra classification allows us to expose to the knowledge expert the representative cases covered by the rule such that he can decide if the causality of a misbehaviour comes from a misgeneralization.

9.3 Rule Base Metrics

A challenging task of rule-based system development is to evaluate the performance and maintainability of its knowledge base. Assuming that the advice given previously has been followed, the specification encapsulated by the behavioural test suite that the system seeks to satisfy contains a small number of representative samples allowing us to quickly certify the specification support. Nevertheless, a specification of the driving policy, by its definition, may allow several distinct rule sets that support its requirements. From this perspective, we had to develop a set of heuristics to determine which rule set is more suitable.

To do so, we have defined the two main optimization axes. The first is the performance metric, motivated by the need to allow a high frequency re-evaluation of the driving policy. Faster execution enables the autonomous vehicle to quickly adapt to behavioural and environmental changes, thus aiding a more reactive driving style. The second is maintainability of the rule set, which we define as the ease of updating an existing rule or adding a new rule that would get precedence in the inferential process. The proposed heuristics are approximations of the real system qualities because in the case of the performance, it is difficult to anticipate the actual scenarios that the autonomous vehicle will met, whereas for maintainability, the specific modifications are difficult to predict.

We approximate the performance of a rule set supporting the behavioural test suite by associating an execution cost for each activation of a rule. This cost is derived from the number of attributes used in the clauses multiplied by the average number of dynamic objects in which the attribute is defined. This average number comes from empirical observations made by agglomerating statistics of Ego's driving on public roads. Furthermore, since the situations tested in the behavioral test suite are not temporally correlated, we empty the volatile memory between each situation assessment. This deletion does not necessarily mean that the system will reevaluate the complete set of rules, since it can still memorize that no rule is active when it starts its deductive process. Thus, the system can choose to activate only the rules depending on the predicates included in the premise of the tested situation. Although this approximation overestimates the reactive capabilities of the system in an environment where the sequences of situations are temporally correlated, this approximation is nevertheless a good indicator of the system's reaction time when an unpredictable situation occurs. For instance, this unpredictability can occur when the perception algorithm is uncertain about the components within the environment. We believe that it is in these critical situations that the reactive capabilities of a system become an asset.

In terms of maintainability, we want to approximate the interaction amongst the rules.

To do this, we empirically compute the number of rules that generated a deduction during the behavioural test suite execution. This count allows us to define the notion of redundant rules and to quantify the superfluous deductions produced. Thus, a rule set generating fewer deductions (and thus leading to less greediness and rejection) is said to be more maintainable since the domain of the rules is narrower.

From that, we can claim that there exists a trade-off between the performance and maintainability metric. For example, if we want to create a rule that only supports the situation described by a behavioural assessment, the rule would depend on a maximum number of predicates. Thus, the rule would activate many times during the execution of the behavioural test suite since the probability that one of these predicates is included in another assessment increase significantly. Therefore, if we take the maintainability metric and push it to the extreme then the system performance approaches its worst-case, reducing the rule set desirability. Similarly, suppose we want to minimize the number of predicates used by rules to maximize the performance metric. Even if the rules are activated less often, the probability of a rule producing a deduction increase due to the lack of constraints. Once again, we see the trade-off where maximizing the performance affects the maintainability of the system. Thus, a good rule set must balance the collective domain of its rules by reconciling these two objectives.

9.4 Rule Set Development

Previously, we have seen heuristics comparing sets of rules satisfying the behavioural test suite. The final element of our method is an approach to find such sets. In the context of our experience, we propose that this task is done through program synthesis. Since the knowledge expert carefully designed a behavioural test suite consisting of representative situations labeled with the desired behaviour, it is feasible to produce many distinct rule sets that satisfy the specification.

Unlike supervised learning, we do not want to generalize potentially contradictory examples, but rather learn to recognize incomplete models as well as forgotten constraints or fallacies in the rules. This thesis proposes a semi-automatic synthesis process that takes advantage of both the analytical skills of the knowledge expert and the exploratory and validation capabilities of a computer system. Assuming that the majority of the subsystems of a autonomous vehicle are imperfect and that their development cycle is continuous, an interactive program synthesis algorithm using representative cases will better adapt its driving policy compared to automatic methods that are typical of machine learning. For instance, since the behavioural test suite is built to delimit the scope of the rules, it is

often enough to modify these human-understandable boundary clauses to entail the new requirements. As an example, assume that a new perception system is deployed such that it can detect dynamic objects farther. In such event, we might want to adjust the level of conservatism in the rules allowing smoother sequence of behaviours. Therefore, the knowledge expert only have to reconsider the clauses of the rules dealing with dynamic object crossing Ego's path, which is only a subset of the rules. Therefore, simulation and testing can focus only on dynamic objects crossing scenarios and can assume that the remaining rules are still accurately modeling the operation design domain.

The proposed development process begins with the identification of a discrepancy. In most cases, this process corresponds to fulfilling an objective not covered by the current operational design domain or the identification of a misbehaviour during simulation testing, closed track testing or public driving. Thereafter, the backward chaining of the expert system is queried to identify the causality of the discrepancy. Since the environment in which the misbehaviour has occurred may contain an exorbitant amount of attributes, it is essential to use the backward chaining process to filter them such that each deduction can be associated with a microenvironment. These microenvironments are thereafter used to identify whether the behaviour of the autonomous vehicle is justified or if changes to the knowledge base are required.

In the event that modifications are required, the knowledge expert wants to understand whether the behaviour is the result of a misgeneralization of a situation already included in the behavioural test suite or whether it is a new situation to incorporate. To facilitate this deduction, the backward chaining mechanism must associate the erroneous deduction with similar examples in the behavioural test suite. Remember that we suggested that this test suite is classified by the pivot attribute, the maneuver and the rule number. Thus, the algorithm should have the ability to perform the relevant associations such that the knowledge expert can identify the problem.

For each problem that the knowledge expert solves, we suggest that he first models the draft of the new rule. This manual modeling activity gives the opportunity to set up the prominent attributes that should be used as well as annotate the overall objective of the rule such that it can be revised later. This annotation is essential since if all the rules were in their disjunctive normal form (DNF) and no hierarchy or grouping would have been modeled, the exponential deduction growth combined with the lack of correlation amongst the rules would decrease the global interpretability of the system. Nevertheless, it is imminent that the knowledge expert unintentionally omits certain attributes mandatory to the satisfaction of the specification due to the complexity of the modeling process. This is where the rule set generator algorithm takes over and completes the rules until they produce a certificate showing that the specification is supported by the rule set. Along

this process, multiple sets of rules are generated and the performance and maintainability metrics are used to order the rule sets to select the most promising one. As a result, the knowledge expert only needs to do a sanity check and confirm that the system's proposal is well-founded and can be safely deployed.

Chapter 10

Related Works

10.1 Expert Systems

Traditionally, expert systems comprise an inferential mechanism and a knowledge base, typically specified through a set of rules [15]. A popular approach to implement such system is forward chaining and the Rete family of algorithms [10]. In such systems, a set of ordered rules operates over and mutates a workspace, which is a network (in Latin "rete") of facts. The network represents the buildup of compound facts used in the rule conditions from more basic facts, and allows reusing the basic facts when computing more complex ones. As part of the Rete algorithm, facts are recomputed by change propagation, avoiding unnecessary computations. The rules in a forward chaining system are typically ordered and may be fired in response to previous rule firings, possibly even the same ones. In this thesis, the approach differs from classical forward chaining by using two sets of unordered rules, as depicted in Fig. 3.2. In other words, the chaining is constrained by rule stratification, where the rules from the second set follow the rules form the first set. Using unordered and stratified rules avoids the explosion of permutations that a knowledge expert must analyze when the knowledge base is extended. While ordering is still needed to resolve rule conflicts, our architecture exploits the domain-specific setting and encapsulates ordering in the priority-based filtering stage. To cover the need of workspace mutations, the proposed approach chains a hierarchy of immutable workspaces and rule sets that preserve the traceability of the inference mechanism along the behavioural deduction. The proposed runtime optimization based on temporal difference is somewhat reminiscent of the Rete spirit. However, Rete optimizes a much more complex forward chaining paradigm than ours, our defined by stratified rules. Yet the Rete ideas to reuse simpler facts to build

larger ones and propagate change in such a network would likely benefit our approach too. In particular, it would allow us to write more maintainable rules without being overly concerned with runtime performance. We leave this extension for future work.

A recurrent shortcoming of expert systems is the difficulty to identify, formalize and structure the domain knowledge [16]. To facilitate this process, Zhang et al. focus on the detection of redundant rules to reduce the size of their knowledge base [15], whereas Suwa et al. concentrate on model-checking to identify contradictions and gaps in the requirements [17]. Other approaches focus on the development of visualization tools that emphasize the relationships amongst the axioms [18]. In this work, the knowledge engineering process does not minimize the size of the rule base but instead focuses on balancing the maintainability and runtime performance. The overall architecture using unordered and stratified rules avoids many of the complexities of more general forward chaining systems. Further, the use of pivot attributes, knowledge base quality metrics, systematically designed unit test suites based on rule premise disjuncts, and tool support for rule synthesis collectively yield an effective method for incremental development of rule bases.

10.2 Behaviour Planning

Early approaches to behaviour planning used finite state machines [19, 20]. These systems construct an automaton over a set of relevant driving states which is evaluated to produce a driving decision. Unfortunately, such systems are difficult to maintain because of the inherent space complexity of the driving problem. Therefore, combination of state machines which effectively decomposes the problem space into easier sub-problems has been studied [21]. In addition to partition the scenario being handled, this hierarchy of state machines can introduce precedence tables to reduce the lack of maintainability [22, 23]. To contrast with the approach presented in this thesis, our hierarchy splits the decision making process into choosing a maneuver and subsequently reconciliating the remaining set of constraints while precedence tables are only applied between these two rule sets.

A recent trend in behaviour planning has been to use machine learning for decision making. For instance, end-to-end machine learning approaches can handle basic driving tasks [5, 24] while *imitation learning* tries to imitate an expert policy to produce a safe and robust driving experience [7]. Imitation learning theory can also be extended to learn from demonstrated trajectories [6, 25], but learning over many examples requires hyper-parameter tuning and long training times. Moreover, the common usage of neural network

to encapsulate the policy often leads to uncertainty of what decisions would be taken for new inputs [26].

In this thesis, the focus is on interpretability which, usually, is a design advantage of declarative systems, for example, Rete family of algorithms [27] and hierarchical probabilistic systems [28].

Overall, the proposed approach differs from the previous works by demonstrating with real-world scenarios how expert system interpretability can be effectively exploited to produce insights for requirements engineering and knowledge acquisition. It also demonstrates how to prevent the autonomous vehicle from navigating in situations in which perceptual denoising cannot be confidently achieved due to the lack of correlation in a sequence of environmental representations. Further, using the proposed analysis framework and the program synthesis algorithms, rule sets can be compared in order to produce a maintainable knowledge base with low coupling.

Chapter 11

Conclusion

In this thesis, we introduced an architecture and a method for building an expert system using two sets of unordered rules. The architecture discretizes the input space using nullable Boolean (true, false, null) and applies a temporal difference to compute historical attributes. We also proposed a strategy to reduce the noise of the perception module by incorporating a tractable long-short term memory which overrides the propositions with values deemed more likely. Then, these atomic propositions are submitted to a first set of rules which evaluates the feasibility of each maneuver in the perceived environment. The proposed maneuvers are filtered using a precedence table based on conservatism to determine which maneuver the Ego vehicle must perform. Then, a second set of rules is used to reconcile the remaining constraints. The finally remaining maneuver and constraints constitute the behaviour that the autonomous vehicle will execute during the next time step, t.

Along with the proposed architecture, we also proposed a method to incrementally build a behavioural test suite supporting the growth of an operational design domain. As part of the method, we proposed a series of metrics to approximate the real-time performance and maintainability of the candidate rule sets satisfying the behavioural test suite. The tool-assisted method consists of four steps. At first, the method identifies the discrepancies in all the testing environments. Then, the Misbehaviour Diagnosis analyzes the scene to identify the attributes of the dynamic and static objects causing the discrepancy. Afterwards, the Knowledge Acquisition uses the rule numbers involved in the inferential process to correlate the perceived situation with those included in the behavioural test suite. This comparison helps the knowledge expert to establish if the misbehaviour is justified by a limitation in the operational design domain or if it is caused by a gap in the behavioural assessments. Finally, the Knowledge Engineering algorithm updates the knowledge base

with the candidate rule set satisfying the behavioural test suite while achieving the best efficiency and maintainability score.

At the end of this thesis, we demonstrated the feasibility of the proposed architecture and method by using both to create a behavioural planner and successfully testing it in the field. In particular, the planner was developed as part of and integrated into Autonomoose, a complete automated driving system software deployed on the research vehicle dubbed "University of Waterloo Moose". The field test involved 110 km of driving in autonomous mode on public roads in Waterloo. During the field test, 58 interventions were required due to perception noise or limitations arising by the short range of the used lidar sensor. Throughout the development of the knowledge base, we demonstrated that the method presented in this thesis can scale with the growing operational design domain. In particular, the operational design domain was incrementally grown by adding new capabilities over time, such as handling additional intersection configurations and crosswalks. However, we experienced some limitations that slowed down the development of the planner. Like in other rule-based systems, modifying the set of rules is time consuming, even with the proposed tool-assisted method. The knowledge acquisition stage is a key bottleneck, where the human expert still has to establish the causality of the misbehaviour. In addition, this thesis did not attempt to solve the problem of hyperparameter tuning. Therefore, once the knowledge base has been updated, the knowledge expert still needs to perform rigorous testing with real-world data to adjust the parameters of the rule set, such as the thresholds used to establish the discrete heading of the vehicle, the coordinate system (longitudinal / lateral relative position threshold and longitudinal / relative path position threshold) used to compute the time to collision, etc.

11.1 Suggestions for Future Work

The presented work opens up several opportunities for extensions to explore in the future.

11.1.1 Program Synthesis

The basic rule synthesis approach presented in Chapter 7.5 could be developed further using more sophisticated optimization techniques than Hill Climbing. For instance, we could use simulated annealing to produce a broader variety of rule sets to compare. In contrast to the possibility of Hill Climbing getting stuck in a local minimum, simulated annealing is more likely to explore a wider variety of rule sets and arrive at a better

solution, at the cost of increased computational effort. As long as the testing speed is not an issue, we could accumulate many distinct instances of the same misbehaviour using the testing environment and use them to ensure that the algorithm has a greater opportunity to generalize. With more representatives of the defects, we could reduce the time spent by the knowledge expert when identifying gaps amongst the requirements. However, whenever this technique becomes a bottleneck, we could also sample the domain with world abstractions near the situation causing the misbehaviour to find a subset of attribute assignments involving the same set of rules. The extrema of this subset could then become the clauses of a new rule inferring the intended behaviour.

11.1.2 Fault Injection

A potentially insightful analysis could be performed by creating mutated copies of the knowledge base and use them to simulate multiple interacting vehicles. For example, if in one instance of the knowledge base we remove the rule specifying *the three mandatory seconds of immobility before crossing an intersection*, the vehicle following this driving policy will perform a stop-and-go that changes the intended precedence at the intersection. Therefore, by disabling some rules, it would be interesting to assess if each instance of the Rule-Engine can avoid collisions properly even when the vehicles do not behave uniformly. Like stated in Chapter 6.2, assuming that a well-modelled rule set should include redundant supports for attenuating the sensitivity to misdetections, this fault injection procedure should help the knowledge expert to model a more robust knowledge base where the faulty behaviours can coexist safely.

11.1.3 Operational Design Domain Growth

With the method presented in this thesis, it is difficult to assess the coverage of the operational design domain. Using the time-free world abstraction model introduced in Chapter 6.3, we believe that a reinforcement learning agent could learn to generate traces where the rules produce an *emergency-stop* maneuver because there was no vote or there was at least a vote (for the *emergency-stop* maneuver) by a rule not ensuring the integrity of the world abstraction. These traces could than be restructured as a lookup table to help the knowledge engineers to identify the disparities amongst the implementation and the operational design domain. Furthermore, this lookup table also describes the atomic propositions causing the autonomous vehicle to enter dangerous or deadlock situations, which should be relevant for the growth of the operational design domain.

11.1.4 Inferential Pipeline

The implementation of the expert system deployed in the autonomous vehicle uses a single thread requiring evaluating the belief of each maneuver rule before selecting a maneuver and filtering the constraints. Although it is conventional to model a rule interpreter by sequentially evaluating the knowledge base, the unordered property of the rule sets combined with the memory state in Boyce-Codd Normal Form allow a variety of parallelization strategies which may increase the run-time efficiency.

Since each rule can be evaluated independently, we can design an inferential pipeline where the evaluation of the knowledge base is performed asynchronously. Moreover, we already shown in Chapter 5.6 that the constrained maneuvers produced by the maneuver rules are agglomerated, and further, preprocessed before evaluating the constraint rules. Therefore, using the memoization suggested in Chapter 5.2 we can update the atomic propositions of any model and only recompute the rules having clauses relying on this literal. Therefore, each rule set can also run asynchronously with a trade-off that the same rule might have to be evaluated more than once because the atomic propositions are updated whenever the belief of a rule in the maneuver rule set is available.

In addition, even if the implementation presented in this thesis only incorporates two sets of rules, it is expected that when the operational design domain will cover lane changes, there could be rule sets inferring the best behaviour to initiate in each lane, and thereafter, another rule set rating the utility of each behaviour allowing the autonomous vehicle to select a behaviour according to safety, comfort and the user driving preferences.

11.1.5 Driving Preference Tuning

Another research area could be to allow drivers to setup the knowledge base according to their driving preferences. Since rule-based systems have the potential of being understandable by non-programmers, we believe that the set of rules can be modified at run time. For instance, Tiugashev implemented an expert system for spacecraft where the astronaut can change the onboard rule interpreter at runtime to explore alternative inferential design when some equipment is not operable [18]. It would be interesting to study how the explainable properties of a rule-based system could be used to educate the autonomous vehicle owners and demonstrate the impact on safety for each of their driving preferences.

References

[1] D. González, J. Pérez, V. Milanés, and F. Nashashibi. A review of motion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 17(4):1135–1145, April 2016.

[2] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1(1):33–55, 2016.

[3] Andrew Bacha, Cheryl Bauman, Ruel Faruque, Michael Fleming, Chris Terwelp, Charles Reinholtz, Dennis Hong, Al Wicks, Thomas Alberi, David Anderson, et al. Odin: Team victortango's entry in the darpa urban challenge. *Journal of field Robotics*, 25(8):467–492, 2008.

[4] N Zimmerman, C Schlenoff, and S Balakirsky. Implementing a rule-based system to represent decision criteria for on-road autonomous navigation. In *2004 AAAI Spring Symp. on Knowledge Representation and Ontologies for Autonomous Systems*, 2004.

[5] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[6] Pieter Abbeel, Dmitri Dolgov, Andrew Y Ng, and Sebastian Thrun. Apprenticeship learning for motion planning with application to parking lot navigation. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1083–1090. IEEE, 2008.

[7] Mayank Bansal, Alex Krizhevsky, and Abhijit S. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *CoRR*, abs/1812.03079, 2018.

[8] A. Adadi and M. Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160, 2018.

[9] Andrea Censi, Konstantin Slutsky, Tichakorn Wongpiromsarn, Dmitry Yershov, Scott Pendleton, James Fu, and Emilio Frazzoli. Liability, ethics, and culture-aware behavior specification using rulebooks. *arXiv preprint arXiv:1902.09355*, 2019.

[10] Charles L Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. In *Readings in Artificial Intelligence and Databases*, pages 547–559. Elsevier, 1989.

[11] K. K. Nambiar, B. Gopinath, T. Nagaraj, and S. Manjunath. Boyce-codd normal form decomposition. *Computers & Mathematics with Applications*, 33(4):1–3, 1997.

[12] Terry Pender. Waterloo's 'Autonomoose' hits 100-kilometre milestone. *Waterloo Region Record*, August 2018.

[13] Nicholas C. Zakas. Ecmascript 6: The definitive guide for javascript developers. *San Francisco, US: no starch press*, pages 1–298, 2016.

[14] Dan. Abramov. Redux. *London, ENG: GitBook*, pages 1–26, 2015.

[15] Yongjie Zhang and Ansheng Deng. Redundancy rules reduction in rule-based knowledge bases. In *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pages 639–643. IEEE, 2015.

[16] Brian R Gaines. Knowledge acquisition: Past, present and future. *International Journal of Human-Computer Studies*, 71(2):135–156, 2013.

[17] Motoi Suwa, A Carlisle Scott, and Edward H Shortliffe. An approach to verifying completeness and consistency in a rule-based expert system. *Ai Magazine*, 3(4):16–16, 1982.

[18] Andrei A Tiugashev. Method of visual construction of real-time knowledge base rules. In *2016 XIX IEEE International Conference on Soft Computing and Measurements (SCM)*, pages 341–344. IEEE, 2016.

[19] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008.

[20] Chris Urmson, Chris Baker, John Dolan, Paul Rybski, Bryan Salesky, William Whittaker, Dave Ferguson, and Michael Darms. Autonomous driving in traffic: Boss and the urban challenge. *AI magazine*, 30(2):17, 2009.

[21] C Shekhar, S Moisan, and M Thonnat. Use of a real-time perception program supervisor in a driving scenario. In *Proceedings of the Intelligent Vehicles' 94 Symposium*, pages 363–368. IEEE, 1994.

[22] Axel Niehaus and Robert F Stengel. An expert system for automated highway driving. *IEEE Control Systems Magazine*, 11(3):53–61, 1991.

[23] Rudolf Gregor, M Lutzeler, Martin Pellkofer, K-H Siedersberger, and Ernst D Dickmanns. Ems-vision: A perceptual system for autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 3(1):48–59, 2002.

[24] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[25] Markus Kuderer, Shilpa Gulati, and Wolfram Burgard. Learning driving styles for autonomous vehicles from demonstration. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2641–2646. IEEE, 2015.

[26] Zachary C Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.

[27] N Zimmerman, C Schlenoff, and S Balakirsky. Implementing a rule-based system to represent decision criteria for on-road autonomous navigation. In *Proceedings of the 2004 AAAI Spring Symposium on Knowledge Representation and Ontologies for Autonomous Systems*, 2004.

[28] Michael Ardelt, Constantin Coester, and Nico Kaempchen. Highly automated driving on freeways in real traffic using a probabilistic framework. *IEEE Trans. Intelligent Transportation Systems*, 13(4):1576–1585, 2012.

APPENDICES

Appendix A

Assumptions and Limitations

Set I – Communication Layer

- The time gap used is sufficient between queries to eliminate the problems of delayed decisions.
- The time gap used is fast enough for the autonomous vehicle to react quickly in the environment.
- The machine interface abstracting the environment communicates at a constant speed with the Rule-Engine by sending promises that can be aborted as needed.
- No device connected to the local network will send requests via the web socket to overload the network or corrupt the temporal logic of the Rule-Engine.
- The machine interface abstracting the environment can handle queries that result in an internal error within the Rule-Engine.

Set II – Data Layer

- Queries sent by the machine interface abstracting the environment use a partial approach where recursive observations are not re-sent or a total approach where all observations are sent, and a null value covers a missing observation.
- The data sent by the machine interface abstracting the environment does not contain any noise from high velocity dynamic object.

Set III – Integration

- The vehicle will behave respecting the decisions issued by the Rule-Engine or it cannot guarantee the phases of temporal logic.
- Only one machine requiring write rights will be integrated by Rule-Engine instance.
- The version of the Rule-Engine used in the autonomous vehicle will start with the command npm run monitor to ensure that the machine can restart in case of a glitch.
- Throughout the autonomous vehicle journey, the npm run monitor script will run indefinitely, without interruption.
- If the Rule-Engine needs to restart, the machine interface abstracting the environment queries will run through the secondary instance of the Rule-Engine.
- The autonomous vehicle responds properly when the Rule-Engine produces a panic decision (emergency-stop)
- There should be no system who overrule the behaviour produced by the Rule-Engine.

Set IV – Localization

- Each dynamic object is located horizontally and vertically in baselink using a measurement in meter.
- Each dynamic object must also be located using a Frenet Frame under the principle of a 3x3 grid.
- The intersection sectors (approaching / at / on) are relative to the lanelets.
- The approaching-intersection sector is large enough to allow the autonomous vehicle traveling at the speed limit to be able to brake in time at the stop.
-

Set V – Interaction with road users

- The autonomous vehicle will not attempt to perform maneuvers where it must drive in the opposite direction.

- The autonomous vehicle will not attempt to perform maneuvers indoors in an enclosed area such as a building, tunnel, enclosed bridge, etc.
- The autonomous vehicle will not attempt to perform maneuvers requiring the vehicles to move in turn.
- The autonomous vehicle will not attempt to perform maneuvers near an intersection with a traffic light.
- The autonomous vehicle will not approach ATVs, bicycles, motorcycles, scooters, snowmobiles, etc.
- The autonomous vehicle will maintain a reasonable distance with any vehicle with a trailer, such as a fifth wheel, a road train, a towing, a van, etc.
- Road users must behave indifferently with the autonomous vehicle and must not try to trap it to test its limits.

Set VI – Safety Driver

- If an emergency vehicle must interact near the autonomous vehicle, the safety driver must take over.
- The safety driver does not kick the steering wheel to trick the Rule-Engine.
- The safety driver must know the Rule-Engine assumptions and limitations and be ready to act accordingly.
- The safety driver should take over if the produced behavior seems to be unsafe in the current environment.

Set VII – Limitations

- The autonomous vehicle must not drive between two lanelets except during an overtake.
- The autonomous vehicle must not drive on lanelets which are prohibited to him.
- The Rule-Engine must not be used on a road with more than one lane driving in the same direction.

- Intersections are reasonably spaced to avoid confusing the signalization of two intersections.
- The Rule-Engine is only used on North American roads.
- The Rule-Engine should not be used in construction zones.

Appendix B

Rules Book

Appendix B - Contents

Book of rules

Set I – Model Integrity

System Id : RMI1
Decision : emergency-stop
Constraint: None
Goal: Make sure there is never more than one leader.

```
SOME-OF {
  many-vehicles {
    isLeading: true
  }

  many-pedestrians {
    isLeading: true
  }

  ALL {
    a-vehicle {
      isLeading: true
    }

    a-pedestrian {
      isLeading: true
    }
  }

}
```

System Id : RMI2
Decision : emergency-stop
Constraint: None
Goal: Make sure that a leading vehicle always go in the same direction as us.

```
SOME-OF {
  a-vehicle {
    orientation: {
      heading: 'not-along-ego',
      longitudinal: 'not-front'
    },
    isLeading: true,
    isParked: false
  }

  a-vehicle {
    orientation: {
      longitudinal: 'not-front'
    },
    isLeading: true,
    isParked: true
  }
}
```

System Id : RMI3
Decision : emergency-stop
Constraint: None
Goal: Obligates the consideration of the nearest spontaneous collision.

```
SOME-OF {
  many-vehicles {
    isObstructing: true
  }

  many-pedestrians {
    isObstructing: true
  }

  ALL {
    a-vehicle {
      isObstructing: true
    }

    a-pedestrian {
      isObstructing: true
    }
  }
}
```

System Id : RMI4
Decision : emergency-stop
Constraint: None
Goal: Require entities to have identifiers.

```
SOME-OF {
  a-pedestrian {
    id: null
  }

  a-vehicle {
    id: null
  }
}
```

System Id : RMI5
Decision : emergency-stop
Constraint: None
Goal: A parked vehicle cannot have a speed.

```
ALL {
  a-vehicle {
    isParked: true,
    speed: > THRESHOLD
  }
}
```

System Id : RMI6
Decision : emergency-stop
Constraint: None
Goal: Each vehicle should be oriented.

```
ALL {
  a-vehicle {
    SOME-OF {
      distance: {
        SOME-OF {
          x: null,
          y: null
        }
      },
      relativeOrientation: {
        heading: ''
      }
    }
  }
}
```

System Id : RMI7
Decision : emergency-stop
Constraint: None
Goal: Each pedestrian should be oriented.

```
ALL {
  a-pedestrian {
    SOME-OF {
      distance: {
        SOME-OF {
          x: null,
          y: null
        }
      },
      relativeOrientation: {
        heading: ''
      }
    }
  }
}
```

System Id : RMI8
Decision : emergency-stop
Constraint: None
Goal: All vehicles with a stop must have a time of arrival.

```
SOME-OF {
  ALL {
    ego {
      location: {
        at: 'intersection',
        timeOfArrival: null
      }
    },
    travel {
      regulation: 'stop'
    }
  }

  a-vehicle {
    location: {
      at: 'intersection',
      timeOfArrival: null
    },
    regulation: 'stop'
  }
}
```

System Id : RMI9
Decision : emergency-stop
Constraint: None
Goal: Ego cannot be simultaneously to several diacritics zones at the same time.

```
ALL {
  ego {
    location: {
      NOT-ALL-DIFFERENT {
        approaching,
        at,
        on
      }
    }
  }
}
```

System Id : RMI10
Decision : emergency-stop
Constraint: None
Goal: Each vehicle cannot be simultaneously to several diacritics zones at the same time.

```
ALL {
  a-vehicle {
    location: {
      NOT-ALL-DIFFERENT {
        approaching,
        at,
        on
      }
    }
  }
}
```

System Id : RMI11
Decision : emergency-stop
Constraint: None
Goal: Each pedestrian cannot be simultaneously to several diacritics zones at the same time.

```
ALL {
  a-pedestrian {
    location: {
      NOT-ALL-DIFFERENT {
        approaching,
        at,
        on
      }
    }
  }
}
```

System Id : RMI12
Decision : emergency-stop
Constraint: None
Goal: Each vehicle should not collide with ego.

```
ALL {
  a-vehicle {
    orientation: {
      lateral: 'center',
      longitudinal: 'center'
    }
  }
}
```

System Id : RMI13
Decision : emergency-stop
Constraint: None
Goal: Each pedestrian should not collide with ego.

```
ALL {
  a-pedestrian {
    orientation: {
      lateral: 'center',
      longitudinal: 'center'
    }
  }
}
```

System Id : RMI14
Decision : emergency-stop
Constraint: None
Goal: Each vehicle should be located somewhere relevant.

```
ALL {
  a-vehicle {
    location: {
      approaching: '',
      at: '',
      on: ''
    }
  }
}
```

System Id : RMI15
Decision : emergency-stop
Constraint: None
Goal: Each pedestrian should be located somewhere relevant.

```
ALL {
  a-pedestrian {
    location: {
      approaching: '',
      at: '',
      on: ''
    }
  }
}
```

System Id : RMI16
Decision : emergency-stop
Constraint: None
Goal: Ego should be located somewhere relevant.

```
ALL {
  ego {
    location: {
      approaching: '',
      at: '',
      on: ''
    }
  }
}
```

System Id : RMI17
Decision : emergency-stop
Constraint: None
Goal: Each vehicle observation must have a unique identifier.

```
ALL {
  a-vehicle {
    id: A_VALUE
  }

  another-vehicle {
    id: A_VALUE
  }
}
```

System Id : RMI18
Decision : emergency-stop
Constraint: None
Goal: Each pedestrian observation must have a unique identifier.

```
ALL {
  a-pedestrian {
    id: A_VALUE
  }

  another-pedestrian {
    id: A_VALUE
  }
}
```

System Id : RMI19
Decision : emergency-stop
Constraint: None
Goal: An identifier cannot be reused to represent a pedestrian and a vehicle at the same time.

```
ALL {
  a-pedestrian {
    id: A_VALUE
  }

  a-vehicle {
    id: A_VALUE
  }
}
```

System Id : RMI20
Decision : emergency-stop
Constraint: None
Goal: A vehicle off road cannot be considered the leader.

```
ALL {
  a-vehicle {
    isLeading: true,
    location: {
      on: 'off-road'
    }
  }
}
```

System Id : RMI21
Decision : emergency-stop
Constraint: None
Goal: A pedestrian off road cannot be considered the leader.

```
ALL {
  a-pedestrian {
    isLeading: true,
    location: {
      on: 'off-road'
    }
  }
}
```

System Id : RMI22
Decision : emergency-stop
Constraint: None
Goal: An intersection must have a positive length and width.

```
ALL {
  ego {
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection',
        on: 'intersection'
      }
    }
  },
  travel {
    intersection: {
      SOME-OF {
        length: this <= 0,
        width: this <= 0
      }
    }
  }
}
```

System Id : RMI23
Decision : emergency-stop
Constraint: None
Goal: An intersection must have a central landmark.

```
ALL {
  ego {
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection',
        on: 'intersection'
      }
    }
  },
  travel {
    intersection: {
      center: {
        SOME-OF {
          x: null,
          y: null
        }
      }
    }
  }
}
```

System Id : RMI24
Decision : emergency-stop
Constraint: None
Goal: The maneuver to be performed at the intersection must always be indicated near an intersection.

```
ALL {
  ego {
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection',
        on: 'intersection'
      }
    },
    navigation: null
  }
}
```

System Id : RMI25
Decision : emergency-stop
Constraint: None
Goal: Each vehicle should have a relative orientation.

```
ALL {
  a-vehicle {
    relativeOrientation: {
      SOME-OF {
        lateral: '',
        longitudinal: ''
      }
    }
  }
}
```

System Id : RMI26
Decision : emergency-stop
Constraint: None
Goal: Each pedestrian should have a relative orientation.

```
ALL {
  a-pedestrian {
    relativeOrientation: {
      SOME-OF {
        lateral: '',
        longitudinal: ''
      }
    }
  }
}
```

System Id : RMI27
Decision : emergency-stop
Constraint: None
Goal: A vehicle going at the same direction as ego cannot obstruct him.

```
ALL {
  a-vehicle {
    isObstructing: true,
    orientation: {
      heading: 'along-ego'
    }
  }
}
```

System Id : RMI28
Decision : emergency-stop
Constraint: None
Goal: Ensures that the at intersection zone cannot contain ego and another vehicle.

```
ALL {
  ego: {
    location: {
      at: 'intersection'
    }
  }

  a-vehicle {
    beliefs: {
      SOME {
        heading: 'along-ego',
        heading: 'untrusted'
      }
    },
    location: {
      at: 'intersection'
    }
    orientation: {
      lateral: 'center',
      longitudinal: 'front'
    }
  }
}
```

System Id : RMI29
Decision : emergency-stop
Constraint: None
Goal: Each vehicle should have a path relative heading.

```
ALL {
  a-vehicle {
    orientation: {
      heading: ''
    }
  }
}
```

System Id : RMI30
Decision : emergency-stop
Constraint: None
Goal: Each pedestrian should have a path relative heading.

```
ALL {
  a-pedestrian {
    orientation: {
      heading: ''
    }
  }
}
```

System Id : RMI31
Decision : emergency-stop
Constraint: None
Goal: A pedestrian crosswalk must have a positive distance landmark.

```
ALL {
  ego {
    location: {
      SOME-OF {
        approaching: 'crosswalk',
        at: 'crosswalk'
      }
    }
  },
  travel {
    crosswalk: {
      SOME-OF {
        distance: null,
        distance: < 0
      }
    }
  }
}
```

System Id : RMI32
Decision : emergency-stop
Constraint: None
Goal: The speed limit should always be defined and positive.

```
ALL {
  travel {
    SOME-OF {
      speedLimit: null,
      speedLimit: < 0
    }
  }
}
```

System Id : RMI32
Decision : emergency-stop
Constraint: None
Goal: The speed limit should always be defined and be positive.

```
ALL {
  travel {
    SOME-OF {
      speedLimit: null,
      speedLimit: < 0
    }
  }
}
```

System Id : RMI33
Decision : emergency-stop
Constraint: None
Goal: The time till a pedestrian enters a crosswalk should always be smaller than the time till the crosswalk is clear.

```
ALL {
  travel {
    crosswalk: {
      timeTillClear: < timeTillPedestrianEnters,
    }
  }
}
```

Set II – Greedy Preventive Rules

System Id : GPR1
Decision : decelerate-to-halt
Constraint: Pedestrian
Goal: Avoid a spontaneous pedestrian when it is not on a crosswalk.

```
ALL {
  a-pedestrian {
    isObstructing: true,
    location: {
      on: 'not-crosswalk'
    }
  }
}
```

System Id : GPR2
Decision : decelerate-to-halt
Constraint: Pedestrian + Vehicle
Goal: Avoid a spontaneous pedestrian with a secure buffer for the leading vehicle when the pedestrian is not on the crosswalk.

```
ALL {
  a-vehicle {
    beliefs: {
      heading: 'not-untrusted'
    },
    isLeading: true,
    location: {
      on: 'not-off-road'
    }
  }

  a-pedestrian {
    isObstructing: true,
    location: {
      on: 'not-crosswalk'
    }
  }
}
```

System Id : GPR3
Decision : decelerate-to-halt
Constraint: Vehicle
Goal: Avoid a spontaneous vehicle.

```
ALL {
  a-vehicle {
    isObstructing: true
  }
}
```

System Id : GPR4
Decision : decelerate-to-halt
Constraint: 2 Vehicles
Goal: Avoid a spontaneous vehicle with a secure buffer for the leading vehicle.

```
ALL {
  a-vehicle {
    beliefs: {
      heading: 'not-untrusted'
    },
    isLeading: true,
    location: {
      on: 'not-off-road'
    }
  }

  a-vehicle {
    isObstructing: true
  }
}
```

System Id : GPR5
Decision : decelerate-to-halt
Constraint: Vehicle
Goal: Avoid a vehicle crossing your lane.

```
ALL {
  a-vehicle {
    beliefs: {
      SOME-OF {
        heading: 'left',
        heading: 'right',
        heading: 'untrusted'
      }
    },
    distance: {
      x: < THRESHOLD,
      y: < THRESHOLD
    },
    location: {
      ALL {
        on: 'not-off-road',
        on: 'not-on-intersection'
      }
    },
    orientation: {
      lateral: 'center',
      longitudinal: 'front'
    }
  }
}
```

System Id : GPR6
Decision : decelerate-to-halt
Constraint: 2 Vehicles
Goal: Avoid a vehicle crossing your lane with a secure buffer for the leading vehicle.

```
ALL {
  a-vehicle {
    beliefs: { heading: 'not-untrusted' },
    isLeading: true,
    location: { on: 'not-off-road' }
  }
  a-vehicle {
    beliefs: {
      SOME-OF {
        heading: 'left',
        heading: 'right',
        heading: 'untrusted'
      }
    },
    distance: {
      x: < THRESHOLD,
      y: < THRESHOLD
    },
    location: {
      ALL {
        on: 'not-off-road',
        on: 'not-on-intersection'
      }
    },
    orientation: {
      lateral: 'center',
      longitudinal: 'front'
    }
  }
}
```

System Id : GPR7
Decision : emergency-stop
Constraint: None
Goal: Avoid dangerous driver maneuvers.

```
ALL {
  a-vehicle {
     beliefs: {
        heading: 'towards-ego'
     },
     location: {
       ALL {
        on: 'not-off-road',
        on: 'not-on-intersection'
       }
     },
     orientation: {
       SOME-OF {
         lateral: 'center',
         lateral: 'right'
       },
       longitudinal: 'front'
    }
  }
}
```

System Id : GPR8
Decision : emergency-stop
Constraint: None
Goal: Panic when a collision has too high probability of colliding with ego.

```
ALL {
  ego {
    location: {
      on: 'intersection'
    }
  }

  a-vehicle {
    location: {
      on: 'intersection'
    },
    SOME-OF {
      ALL {
        beliefs: {
          heading: 'left'
        },
        relativeOrientation: {
          lateral: 'not-left',
          longitudinal: 'front'
        }
      }
      ALL {
        beliefs: {
          heading: 'right'
        },
        relativeOrientation: {
          lateral: 'not-right',
          longitudinal: 'front'
        }
      }
    }
  }
}
```

System Id : GPR9
Decision : emergency-stop
Constraint: None
Goal: Immobilize the vehicle if there is a parked vehicle on your way near an intersection.

```
ALL {
  ego {
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection'
      }
    }
  }

  a-vehicle {
    isParked: true,
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection'
      }
    },
    orientation: {
      lateral: 'center',
      longitudinal: 'front'
    }
  }}
```

System Id : GPR10
Decision : emergency-stop
Constraint: None
Goal: Immobilize the vehicle when there is a parked vehicle on an intersection.

```
ALL {
  ego {
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection',
        on: 'intersection'
      }
    }
  }

  a-vehicle {
    isParked: true,
    location: {
      on: 'intersection'
    },
    orientation: {
      longitudinal: 'not-behind'
    }
  }
}
```

Set III – Two-lane road

System Id : RTLR1
Decision : track-speed
Constraint: Speed Limit
Goal: Continue to move to the next destination.

```
ALL {
  ego {
    location: {
      approaching: '',
      at: '',
      on: 'drive-lane'
    }
  }
}
```

System Id : RTLR2
Decision : follow-leader
Constraint: Vehicle + Speed Limit
Goal: Follow the vehicle in single file.

```
ALL {
  ego {
    location: {
      approaching: '',
      at: '',
      on: 'drive-lane'
    }
  }

  a-vehicle {
    beliefs: {
      heading: 'not-untrusted'
    },
    isLeading: true,
    location: {
      on: 'not-off-road'
    }
  }
}
```

System Id : RTLR3
Decision : follow-leader
Constraint: Pedestrian + Speed Limit
Goal: Follow the indications of an authoritarian pedestrian (i.e. construction guy).

```
ALL {
  ego {
    location: {
      approaching: '',
      at: '',
      on: 'drive-lane'
    }
  }

  a-pedestrian {
    beliefs: {
      heading: 'not-untrusted'
    },
    isLeading: true,
    location: {
      on: 'not-off-road'
    }
  }
}
```

System Id : RTLR4
Decision : yield
Constraint: N Vehicles
Goal: Yield until the road is clear before starting an overtaking procedure for one or more parked vehicle.

```
ALL {
  ego {
    location: {
      approaching: '',
      at: '',
      on: 'drive-lane',
    },
    speed: this <= THRESHOLD
  }

  a-vehicle {
    beliefs: {
      heading: 'not-untrusted'
    },
    distance: {
      x: this <= THRESHOLD,
      y: this <= THRESHOLD
    },
    isLeading: true,
    isParked: true,
    location: {
      on: 'not-off-road'
    },
    orientation: {
      lateral: 'center',
      longitudinal: 'front'
    }
  }

  SOME-OF {
    a-vehicle {
      location: {
        on: 'not-off-road'
      },
      orientation: {
        lateral: 'left',
        longitudinal: 'center'
      }
    }

    a-vehicle {
      beliefs: {
        SOME-OF {
          heading: 'along-ego',
          heading: 'untrusted'
        }
      },
      isParked: false,
      location: {
        on: 'not-off-road'
      },
      orientation: {
        longitudinal: 'front'
      },
      speed: <= THRESHOLD
    }
```

```
    a-vehicle {
      beliefs: {
        SOME-OF {
          heading: 'along-ego',
          heading: 'untrusted'
        }
      },
      isParked: false,
      location: {
        on: 'not-off-road'
      },
      orientation: {
        longitudinal: 'behind'
      },
      speed: > ego
    }

    a-vehicle {
      beliefs: {
        heading: 'not-along-ego'
      },
      location: { on: 'not-off-road' },
      orientation: {
        longitudinal: 'front'
      }
      SOME-OF {
        speed: this > THRESHOLD,
        ALL {
          distance: {
            x: this <= THRESHOLD,
            y: this <= THRESHOLD
          },
          speed: this <= THRESHOLD
        }
      }
    }
  }
}
```

System Id : RTLR5
Decision : overtake
Constraint: N Vehicles + Speed Limit
Goal: Start an overtaking procedure for one or more parked vehicle.

```
ALL {
  ego {
    location: {
      approaching: '',
      at: '',
      on: 'drive-lane',
    },
    speed: this <= THRESHOLD
  }

  a-vehicle {
    beliefs: { heading: 'not-untrusted' },
    distance: {
      x: this <= THRESHOLD,
      y: this <= THRESHOLD
    },
    isLeading: true,
    isParked: true,
    location: { on: 'not-off-road' },
    orientation: {
      lateral: 'center',
      longitudinal: 'front'
    }
  }

  no-vehicle {
    location: {
      on: 'not-off-road'
    },
    orientation: {
      lateral: 'left',
      longitudinal: 'center'
    }
  }

  no-vehicle {
    beliefs: {
      SOME-OF {
        heading: 'along-ego',
        heading: 'untrusted'
      }
    },
    isParked: false,
    location: {
      on: 'not-off-road'
    },
    orientation: {
      longitudinal: 'front'
    },
    speed: <= THRESHOLD
  }

  no-vehicle {
    beliefs: {
      SOME-OF {
        heading: 'along-ego',
        heading: 'untrusted'
      }
    },
    isParked: false,
    location: {
      on: 'not-off-road'
    },
    orientation: {
      longitudinal: 'behind'
    },
    speed: > ego
  }

  no-vehicle {
    beliefs: {
      heading: 'not-along-ego'
    },
    location: {
      on: 'not-off-road'
    },
    orientation: {
      longitudinal: 'front'
    }
    SOME-OF {
      speed: this > THRESHOLD,
      ALL {
        distance: {
          x: this <= THRESHOLD,
          y: this <= THRESHOLD
        },
        speed: this <= THRESHOLD
      }
    }
  }
}
```

System Id : RTLR6
Decision : overtake
Constraint: N Vehicles + Speed Limit
Goal: Proceed in the overtaking procedure for one or more parked vehicle.

```
ALL {
  ego {
    lastManeuver: 'overtake',
    location: {
      approaching: '',
      at: '',
      on: 'drive-lane',
    }
  }

  a-vehicle {
    distance: {
      x: this <= THRESHOLD,
      y: this <= THRESHOLD
    },
    isParked: true,
    location: {
      on: 'not-off-road'
    },
    orientation: {
      SOME-OF {
        ALL {
          lateral: 'right',
          longitudinal: 'center'
        },
        ALL {
          SOME-OF {
            lateral: 'center',
            lateral: 'right'
          }
          longitudinal: 'front'
        }
      },
    }
  }

  no-vehicle {
    location: {
      on: 'not-off-road'
    },
    orientation: {
      lateral: 'left',
      longitudinal: 'center'
    }
  }

  no-vehicle {
    beliefs: {
      SOME-OF {
        heading: 'along-ego',
        heading: 'untrusted'
      }
    },
    isParked: false,
    location: {
      on: 'not-off-road'
    },
    orientation: {
      longitudinal: 'front'
    },
    speed: <= THRESHOLD
  }

  no-vehicle {
    beliefs: {
      heading: 'not-along-ego'
    },
    location: {
      on: 'not-off-road'
    },
    orientation: {
      longitudinal: 'front'
    }
    SOME-OF {
      speed: this > THRESHOLD,
      ALL {
        distance: {
          x: this <= THRESHOLD,
          y: this <= THRESHOLD
        },
        speed: this <= THRESHOLD
      }
    }
  }

  no-vehicle {
    isParked: false,
    location: {
      on: 'not-off-road'
    },
    orientation: {
      heading: 'along-ego',
      longitudinal: 'behind'
    },
    speed: > ego
  }
}
```

System Id : RTLR7
Decision : emergency-stop
Constraint: None
Goal: Cancelling the overtaking decision if a source of danger is visible.

```
ALL {
  ego {
    lastManeuver: 'overtake',
    location: {
      approaching: '',
      at: '',
      on: 'drive-lane',
    }
  }

  a-vehicle {
    distance: {
      x: this <= THRESHOLD,
      y: this <= THRESHOLD
    },
    isParked: true,
    location: {
      on: 'not-off-road'
    },
    orientation: {
      SOME-OF {
        ALL {
          lateral: 'right',
          longitudinal: 'center'
        },
        ALL {
          SOME-OF {
            lateral: 'center',
            lateral: 'right'
          },
          longitudinal: 'front'
        }
      },
    }
  }

  SOME-OF {
    a-vehicle {
      location: {
        on: 'not-off-road'
      },
      orientation: {
        lateral: 'left',
        longitudinal: 'center'
      }
    }

    a-vehicle {
      beliefs: {
        SOME-OF {
          heading: 'along-ego',
          heading: 'untrusted'
        }
      },
      isParked: false,
      location: {
        on: 'not-off-road'
      },
      orientation: {
        longitudinal: 'front'
      },
      speed: <= THRESHOLD
    }

    a-vehicle {
      beliefs: {
        SOME-OF {
          heading: 'along-ego',
          heading: 'untrusted'
        }
      },
      location: {
        on: 'not-off-road'
      },
      orientation: {
        longitudinal: 'behind'
      },
      speed: > ego
    }


    a-vehicle {
      beliefs: {
        heading: 'not-along-ego'
      },
      isParked: false,
      location: {
        on: 'not-off-road'
      },
      orientation: {
        longitudinal: 'front'
      }
      SOME-OF {
        speed: this > THRESHOLD,
        ALL {
          distance: {
            x: this <= THRESHOLD,
            y: this <= THRESHOLD
          },
          speed: this <= THRESHOLD
        }
      }
    }
  }
}
```

Set IV – Crosswalk

System Id : RCW1
Decision : track-speed
Constraint: Reduced Speed
Goal: Reduce ego's speed when nearby a crosswalk.

```
ALL {
  ego {
    location: {
      SOME-OF {
        approaching: 'crosswalk',
        at: 'crosswalk',
        on: 'crosswalk'
      }
    }
  }
}
```

System Id : RCW2
Decision : decelerate-to-halt
Constraint: Crosswalk
Goal: Yield to pedestrians on the crosswalk.

```
ALL {
  ego {
    location: {
      approaching: 'crosswalk'
    }
  },
  travel: {
    crosswalk: {
      timeTillPedestrianEnters: <= THRESHOLD,
      timeTillClear: >= THRESHOLD
    }
  }
}
```

System Id : RCW3
Decision : yield
Constraint: Crosswalk
Goal: Remain stationary until the crosswalk is freed

```
ALL {
  ego {
    location: {
      at: 'crosswalk'
    },
    speed: <= THRESHOLD
  },
  travel: {
    crosswalk: {
      timeTillPedestrianEnters: <= THRESHOLD,
      timeTillClear: >= THRESHOLD
    }
  }
}
```

System Id : RCW4
Decision : emergency-stop
Constraint: None
Goal: Warn the safety driver about a potential unavoidable collision at a crosswalk.

```
ALL {
  ego {
    location: {
      at: 'crosswalk'
    },
    speed: > THRESHOLD
  },
  travel: {
    crosswalk: {
      timeTillPedestrianEnters: <= THRESHOLD,
      timeTillClear: >= THRESHOLD
    }
  }
}
```

System Id : RCW5
Decision : emergency-stop
Constraint: None
Goal: Warn the safety driver about a potential unavoidable collision on a crosswalk.

```
ALL {
  ego {
    location: {
      on: 'crosswalk'
    }
  },
  travel: {
    crosswalk: {
      timeTillPedestrianEnters: <= THRESHOLD,
      timeTillClear: >= THRESHOLD
    }
  }
}
```

System Id : RCW6
Decision : follow-leader
Constraint: Vehicle + Reduced Speed
Goal: Keep following the leading vehicle with a reduced speed when nearby a crosswalk.

```
ALL {
  ego {
    location: {
      SOME-OF {
        approaching: 'crosswalk',
        at: 'crosswalk',
        on: 'crosswalk'
      }
    }
  },
  a-vehicle: {
    isLeading: true,
    location: {
      on: 'not-off-road'
    }
  }
}
```

System Id : RCW7
Decision : decelerate-to-halt
Constraint: Vehicle + Crosswalk
Goal: Yield to pedestrians on the crosswalk with a gap for the leading vehicle.

```
ALL {
  ego {
    location: {
      approaching: 'crosswalk'
    }
  },
  a-vehicle: {
    isLeading: true,
    location: {
      SOME-OF {
        approaching: 'crosswalk',
        at: 'crosswalk'
      }
      on: 'not-off-road'
    }
  }
  travel: {
    crosswalk: {
      timeTillPedestrianEnters: <= THRESHOLD,
      timeTillClear: >= THRESHOLD
    }
  }
}
```

Set V – Miscellaneous obstacles, Speedbump

System Id : MOS1
Decision : track-speed
Constraint: Reduced Speed + Speedbump
Goal: Reduce the speed when ego's is approaching a speedbump.

```
ALL {
  ego {
    location: {
      approaching: 'speedbump'
    }
  }
}
```

System Id : MOS2
Decision : track-speed
Constraint: Reduced Speed
Goal: Make a slow collision with the speedbump.

```
ALL {
  ego {
    location: {
      SOME-OF {
        at: 'speedbump',
        on: 'speedbump'
      }
    }
  }
}
```

System Id : MOS3
Decision : emergency-stop
Constraint: None
Goal: Warn the safety driver about an unintended high velocity during the collision with the speedbump.

```
ALL {
  ego {
    location: {
      SOME-OF {
        at: 'speedbump',
        on: 'speedbump'
      }
    },
    speed: > THRESHOLD
  }
}
```

System Id : MOS4
Decision : follow-leader
Constraint: Vehicle + Reduced Speed + Speedbump
Goal: Reduce the speed when ego's is approaching a speedbump and there exists a leading vehicle.

```
ALL {
  ego {
    location: {
      approaching: 'speedbump'
    }
  },
  a-vehicle: {
    isLeading: true,
    location: {
      on: 'not-off-road'
    }
  }
}
```

System Id : MOS5
Decision : follow-leader
Constraint: Vehicle + Reduced Speed
Goal: Make a slow collision with the speedbump in line with a leading vehicle.

```
ALL {
  ego {
    location: {
      SOME-OF {
        at: 'speedbump',
        on: 'speedbump'
      }
    }
  },
  a-vehicle: {
    isLeading: true,
    location: {
      on: 'not-off-road'
    }
  }
}
```

Set VI – Intersection with stop sign for ego

System Id : RIS1
Decision : track-speed
Constraint: Speed Limit
Goal: Cross the intersection if no danger zone is visible.
We should have a distance/speed gap for the vehicle approaching intersection

```
ALL {
  ego {
    location: {
      on: 'intersection'
    }
  }

  no-vehicle {
    beliefs: {
      heading: 'not-along-ego
    },
    isParked: false,
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection'
      },
      on: 'not-off-road'
    },
    regulation: 'no-stop'
  }

  no-vehicle {
    location: {
      on: 'intersection'
    },
    SOME-OF {
      relativeOrientation: {
        heading: 'towards-ego'
      },
      ALL {
        beliefs: {
          heading: 'untrusted'
        },
        relativeOrientation: {
          longitudinal: 'front'
        }
      }
      ALL {
        relativeOrientation: {
          heading: 'left',
          lateral: 'not-left',
          longitudinal: 'front'
        }
      }
      ALL {
        relativeOrientation: {
          heading: 'right',
          lateral: 'not-right',
          longitudinal: 'front'
        }
      }
    }
  }
}
```

System Id : RIS2
Decision : follow-leader
Constraint: Vehicle + Speed Limit
Goal: Cross the intersection in line if no danger zone is visible.
We should have a distance/speed gap for the vehicle approaching intersection

```
ALL {
  ego {
    location: {
      on: 'intersection'
    }
  }

  a-vehicle: {
    beliefs: {
      heading: 'not-untrusted'
    },
    isLeading: true,
    location: { on: 'not-off-road' }
  }

  no-vehicle {
    beliefs: { heading: 'not-along-ego },
    isParked: false,
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection'
      },
      on: 'not-off-road'
    },
    regulation: 'no-stop'
  }

  no-vehicle {
    location: { on: 'intersection' },
    SOME-OF {
      relativeOrientation: {
        heading: 'towards-ego'
      },
      ALL {
        beliefs: { heading: 'untrusted' },
        relativeOrientation: {
          longitudinal: 'front'
        }
      }
      ALL {
        relativeOrientation: {
          heading: 'left',
          lateral: 'not-left',
          longitudinal: 'front'
        }
      }
      ALL {
        relativeOrientation: {
          heading: 'right',
          lateral: 'not-right',
          longitudinal: 'front'
        }
      }
    }
  }
}
```

System Id : RIS3
Decision : decelerate-to-halt
Constraint: Stop Line
Goal: Slow down to stop at the stop line of the intersection.

```
ALL {
  ego {
    location: {
      approaching: 'intersection'
    }
  }

  travel: {
    regulation: 'stop'
  }
}
```

System Id : RIS4
Decision : decelerate-to-halt
Constraint: Stop Line + Vehicle
Goal: Slow down to stop in line at the stop sign.

```
ALL {
  ego {
    location: {
      approaching: 'intersection'
    }
  }

  travel: {
    regulation: 'stop'
  }

  a-vehicle: {
    isLeading: true,
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection'
      },
      on: 'not-off-road'
    }
  }
}
```

System Id : RIS5
Decision : decelerate-to-halt
Constraint: Stop Line
Goal: Smoothly end the deceleration to stop at the stop line of the intersection.

```
ALL {
  ego {
    location: {
      at: 'intersection'
    },
    speed: > THRESHOLD,
    SOME-OF {
      stopBeginAt: 0,
      stopElapsedTime: this / 1000 <=
STOP_THRESHOLD
    }
  }

  travel: {
    regulation: 'stop'
  }
}
```

System Id : RIS6
Decision : stop
Constraint: None
Goal: Make a complete stop according to the Canadian Highway Code.

```
ALL {
  ego {
    location: {
      at: 'intersection'
    },
    speed: < THRESHOLD,
    SOME-OF {
      stopBeginAt: 0,
      stopElapsedTime: this / 1000 <=
STOP_THRESHOLD
    }
  }

  travel: {
    regulation: 'stop'
  }
}
```

System Id : RIS7
Decision : track-speed
Constraint: Speed Limit
Goal: Take the right of way when all vehicles have a mandatory stop and we have the precedence.

```
ALL {
  ego {
    location: {
      at: 'intersection'
      timeOfArrival: T1
    }
  }

  travel: {
    regulation: 'stop'
  }

  no-vehicle: {
    isParked: false,
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection'
      },
      on: 'not-off-road'
    },
    regulation: 'no-stop'
  }

  no-vehicle: {
    isParked: false,
    location: {
      at: 'intersection',
      on: 'not-off-road',
      timeOfArrival: <= T1
    },
    regulation: 'stop',
  }

  no-vehicle {
    location: { on: 'intersection' },
    SOME-OF {
      relativeOrientation: {
        heading: 'towards-ego'
      },
      ALL {
        beliefs: { heading: 'untrusted' },
        relativeOrientation: {
          longitudinal: 'front'
        }
      }
      ALL {
        relativeOrientation: {
          heading: 'left',
          lateral: 'not-left',
          longitudinal: 'front'
        }
      }
      ALL {
        relativeOrientation: {
          heading: 'right',
          lateral: 'not-right',
          longitudinal: 'front'
        }
      }
    }
  }
}
```

System Id : RIS8
Decision : follow-leader
Constraint: Vehicle + Speed Limit
Goal: Take the right of way when all vehicles have a mandatory stop and we have the precedence even if there is a leading vehicle.

```
ALL {
  ego {
    location: {
      at: 'intersection'
      timeOfArrival: T1
    }
  }
  travel: { regulation: 'stop' }
  a-vehicle: {
    beliefs: { heading: 'not-untrusted' },
    isLeading: true,
    location: { on: 'not-off-road' }
  }
  no-vehicle: {
    isParked: false,
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection'
      },
      on: 'not-off-road'
    },
    regulation: 'no-stop'
  }
  no-vehicle: {
    isParked: false,
    location: {
      at: 'intersection',
      on: 'not-off-road',
      timeOfArrival: <= T1
    },
    regulation: 'stop'
  }
  no-vehicle {
    location: { on: 'intersection' },
    SOME-OF {
      relativeOrientation: {
        heading: 'towards-ego'
      },
      ALL {
        beliefs: { heading: 'untrusted' },
        relativeOrientation: {
          longitudinal: 'front'
        }
      }
      ALL {
        relativeOrientation: {
          heading: 'left',
          lateral: 'not-left',
          longitudinal: 'front'
        }
      }
      ALL {
        relativeOrientation: {
          heading: 'right',
          lateral: 'not-right',
          longitudinal: 'front'
        }
      }
    }
  }
}
```

System Id : RIS9
Decision : yield
Constraint: N Vehicles
Goal: Yield the right of way when incoming traffic does not have a mandatory stop or we don't have the precedence.

```
ALL {
  ego {
    location: {
      at: 'intersection'
      timeOfArrival: T1
    }
  }
  travel: {
    regulation: 'stop'
  }
  SOME-OF {
    a-vehicle: {
      isParked: false,
      location: {
        SOME-OF {
          approaching: 'intersection',
          at: 'intersection'
        },
        on: 'not-off-road'
      },
      regulation: 'no-stop'
    }
    a-vehicle: {
      isParked: false,
      location: {
        at: 'intersection',
        on: 'not-off-road',
        timeOfArrival: <= T1
      },
      regulation: 'stop'
    }
    a-vehicle {
      location: {
        on: 'intersection'
      },
      SOME-OF {
        relativeOrientation: {
          heading: 'towards-ego'
        },
        ALL {
          beliefs: { heading: 'untrusted' },
          relativeOrientation: {
            longitudinal: 'front'
          }
        }
        ALL {
          relativeOrientation: {
            heading: 'left',
            lateral: 'not-left',
            longitudinal: 'front'
          }
        }
        ALL {
          relativeOrientation: {
            heading: 'right',
            lateral: 'not-right',
            longitudinal: 'front'
          }
        }
      }
    }
  }
}
```

System Id : RIS10
Decision : yield
Constraint: N Pedestrians
Goal: Go straight to the intersection will only wait if an unparallelly pedestrian cross at the same time.

```
ALL {
  ego {
    location: {
      at: 'intersection'
    },
    navigation: 'straight'
  },

  travel: {
    regulation: 'stop'
  },

  a-pedestrian: {
    distance: {
      x: potentiallyCollide(ego),
      y: potentiallyCollide(ego)
    },
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection',
        on: 'intersection'
      }
    }
  }
}
```

System Id : RIS11
Decision : emergency-stop
Constraint: None
Goal: Go straight to the intersection with unpredicted pedestrian will only stop if an unparallelly pedestrian cross at the same time.

```
ALL {
  ego {
    location: {
      on: 'intersection'
    },
    navigation: 'straight'
  }

  travel {
    wasRegulatedByIntersectionStop: true
  }

  a-pedestrian: {
    distance: {
      x: potentiallyCollide(ego),
      y: potentiallyCollide(ego)
    },
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection',
        on: 'intersection'
      }
    }
  }
}
```

System Id : RIS12
Decision : yield
Constraint: N Pedestrians
Goal: Go left to the intersection will only wait if a pedestrian cross at angle the same time.

```
ALL {
  ego {
    location: {
      at: 'intersection'
    },
    navigation: 'left'
  }

  travel: {
    regulation: 'stop'
  },

  a-pedestrian: {
    distance: {
      x: potentiallyCollide(ego),
      y: potentiallyCollide(ego)
    },
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection',
        on: 'intersection'
      }
    }
  }
}
```

System Id : RIS13
Decision : emergency-stop
Constraint: None
Goal: Go left to the intersection with unpredicted pedestrian will only stop if a pedestrian cross at angle the same time.

```
ALL {
  ego {
    location: {
      on: 'intersection'
    },
    navigation: 'left'
  }

  travel {
    wasRegulatedByIntersectionStop: true
  }

  a-pedestrian: {
    distance: {
      x: potentiallyCollide(ego),
      y: potentiallyCollide(ego)
    },
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection',
        on: 'intersection'
      }
    }
  }
}
```

System Id : RIS14
Decision : yield
Constraint: N Pedestrians
Goal: Go right to the intersection will only wait if a pedestrian cross at angle the same time.

```
ALL {
  ego {
    location: {
      at: 'intersection'
    },
    navigation: 'right'
  }

  travel: {
    regulation: 'stop'
  },

  a-pedestrian: {
    distance: {
      x: potentiallyCollide(ego),
      y: potentiallyCollide(ego)
    },
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection',
        on: 'intersection'
      }
    }
  }
}
```

System Id : RIS15
Decision : emergency-stop
Constraint: None
Goal: Go right to the intersection with unpredicted pedestrian will only stop if a pedestrian cross at angle the same time.

```
ALL {
  ego {
    location: {
      on: 'intersection'
    },
    navigation: 'right'
  }

  travel {
    wasRegulatedByIntersectionStop: true
  }

  a-pedestrian: {
    distance: {
      x: potentiallyCollide(ego),
      y: potentiallyCollide(ego)
    },
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection',
        on: 'intersection'
      }
    }
  }
}
```

Set VII – Intersection without stop sign for ego

System Id : RIWS1
Decision : track-speed
Constraint: Speed Limit
Goal: Cross an intersection if no source of danger is visible.

```
ALL {
  ego {
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection'
      }
    }
  }
  travel: {
    regulation: 'no-stop'
  }
  no-vehicle {
    SOME-OF {
      beliefs: { heading: 'untrusted' },
      relativeOrientation: {
        heading: 'left',
        heading: 'right'
      }
    }
    isParked: false,
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection'
      },
      on: 'not-off-road'
    },
    regulation: 'no-stop',
    relativeOrientation: {
      longitudinal: 'front'
    }
  }
  no-vehicle {
    location: { on: 'intersection' },
    SOME-OF {
      ALL {
        beliefs: { heading: 'untrusted' },
        relativeOrientation: {
          longitudinal: 'front'
        }
      }
      ALL {
        relativeOrientation: {
          heading: 'left',
          lateral: 'not-left',
          longitudinal: 'front'
        }
      }
      ALL {
        relativeOrientation: {
          heading: 'right',
          lateral: 'not-right',
          longitudinal: 'front'
        }
      }
    }
  }
}
```

System Id : RIWS2
Decision : track-speed
Constraint: Speed Limit
Goal: Cross an intersection if no source of danger is visible.

```
ALL {
  ego {
    location: {
      on: 'intersection'
    }
  }

  travel: {
    wasRegulatedByIntersectionStop: false
  }

  no-vehicle {
    SOME-OF {
      beliefs: { heading: 'untrusted' },
      relativeOrientation: {
        heading: 'left',
        heading: 'right'
      }
    }
    isParked: false,
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection'
      },
      on: 'not-off-road'
    },
    regulation: 'no-stop',
    relativeOrientation: {
      longitudinal: 'front'
    }
  }

  no-vehicle {
    location: { on: 'intersection' },
    SOME-OF {
      ALL {
        beliefs: { heading: 'untrusted' },
        relativeOrientation: {
          longitudinal: 'front'
        }
      }
      ALL {
        relativeOrientation: {
          heading: 'left',
          lateral: 'not-left',
          longitudinal: 'front'
        }
      }
      ALL {
        relativeOrientation: {
          heading: 'right',
          lateral: 'not-right',
          longitudinal: 'front'
        }
      }
    }
  }
}
```

System Id : RIWS3
Decision : decelerate-to-halt
Constraint: End of Lane
Goal: Slow down before entering the intersection if some source of danger is visible.

```
ALL {
  ego {
    location: {
      approaching: 'intersection',
    }
  }

  travel: {
    regulation: 'no-stop'
  }

  SOME-OF {
    a-vehicle {
      SOME-OF {
        beliefs: { heading: 'untrusted' },
        relativeOrientation: {
          heading: 'left',
          heading: 'right'
        }
      }
      isParked: false,
      location: {
        SOME-OF {
          approaching: 'intersection',
          at: 'intersection'
        },
        on: 'not-off-road'
      },
      regulation: 'no-stop',
      relativeOrientation: {
        longitudinal: 'front'
      }
    }

    a-vehicle {
      location: {
        on: 'intersection'
      },
      SOME-OF {
        ALL {
          beliefs: { heading: 'untrusted' },
          relativeOrientation: {
            longitudinal: 'front'
          }
        }
        ALL {
          relativeOrientation: {
            heading: 'left',
            lateral: 'not-left',
            longitudinal: 'front'
          }
        }
        ALL {
          relativeOrientation: {
            heading: 'right',
            lateral: 'not-right',
            longitudinal: 'front'
          }
        }
      }
    }
  }
}
```

System Id : RIWS4
Decision : decelerate-to-halt
Constraint: End of Lane
Goal: Decelerate safely to wait until the intersection is safe if some source of danger is visible.

```
ALL {
  ego {
    location: {
      at: 'intersection',
    },
    speed: <= THRESHOLD
  }
  travel: {
    regulation: 'no-stop'
  }

  SOME-OF {
    a-vehicle {
      SOME-OF {
        beliefs: { heading: 'untrusted' },
        relativeOrientation: {
          heading: 'left',
          heading: 'right'
        }
      }
      isParked: false,
      location: {
        SOME-OF {
          approaching: 'intersection',
          at: 'intersection'
        },
        on: 'not-off-road'
      },
      regulation: 'no-stop',
      relativeOrientation: {
        longitudinal: 'front'
      }
    }

    a-vehicle {
      location: { on: 'intersection' },
      SOME-OF {
        ALL {
          beliefs: { heading: 'untrusted' },
          relativeOrientation: {
            longitudinal: 'front'
          }
        }
        ALL {
          relativeOrientation: {
            heading: 'left',
            lateral: 'not-left',
            longitudinal: 'front'
          }
        }
        ALL {
          relativeOrientation: {
            heading: 'right',
            lateral: 'not-right',
            longitudinal: 'front'
          }
        }
      }
    }
  }
}
```

System Id : RIWS5
Decision : emergency-stop
Constraint: None
Goal: Panic when an intersection has an ambiguous shape that may lead to a collision with ego.

```
ALL {
  ego {
    location: {
      at: 'intersection'
    },
    speed: > THRESHOLD
  }

  travel: {
    regulation: 'no-stop'
  }

  SOME-OF {
    a-vehicle {
      SOME-OF {
        beliefs: { heading: 'untrusted' },
        relativeOrientation: {
          heading: 'left',
          heading: 'right'
        }
      }
      isParked: false,
      location: {
        SOME-OF {
          approaching: 'intersection',
          at: 'intersection'
        },
        on: 'not-off-road'
      },
      regulation: 'no-stop',
      relativeOrientation: {
        longitudinal: 'front'
      }
    }

    a-vehicle {
      location: { on: 'intersection' },
      SOME-OF {
        ALL {
          beliefs: { heading: 'untrusted' },
          relativeOrientation: {
            longitudinal: 'front'
          }
        }
        ALL {
          relativeOrientation: {
            heading: 'left',
            lateral: 'not-left',
            longitudinal: 'front'
          }
        }
        ALL {
          relativeOrientation: {
            heading: 'right',
            lateral: 'not-right',
            longitudinal: 'front'
          }
        }
      }
    }
  }
}
```

System Id : RIWS6
Decision : follow-leader
Constraint: Vehicle + Speed Limit
Goal: Cross in line at an intersection if no source of danger is visible.

```
ALL {
  ego {
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection'
      }
    }
  }
  travel: { regulation: 'no-stop' }
  a-vehicle: {
    beliefs: { heading: 'not-untrusted' },
    isLeading: true,
    location: { on: 'not-off-road' }
  }
  no-vehicle {
    SOME-OF {
      beliefs: { heading: 'untrusted' },
      relativeOrientation: {
        heading: 'left',
        heading: 'right'
      }
    }
    isParked: false,
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection'
      },
      on: 'not-off-road'
    },
    regulation: 'no-stop',
    relativeOrientation: {
      longitudinal: 'front'
    }
  }
  no-vehicle {
    location: { on: 'intersection' },
    SOME-OF {
      ALL {
        beliefs: { heading: 'untrusted' },
        relativeOrientation: {
          longitudinal: 'front'
        }
      }
      ALL {
        relativeOrientation: {
          heading: 'left',
          lateral: 'not-left',
          longitudinal: 'front'
        }
      }
      ALL {
        relativeOrientation: {
          heading: 'right',
          lateral: 'not-right',
          longitudinal: 'front'
        }
      }
    }
  }
}
```

System Id : RIWS7
Decision : follow-leader
Constraint: Vehicle + Speed Limit
Goal: Cross in line at an intersection if no source of danger is visible.

```
ALL {
  ego {
    location: {
      on: 'intersection'
    }
  }
  travel: {
    wasRegulatedByIntersectionStop: false
  }
  a-vehicle: {
    beliefs: { heading: 'not-untrusted' },
    isLeading: true,
    location: { on: 'not-off-road' }
  }

  no-vehicle {
    SOME-OF {
      beliefs: { heading: 'untrusted' },
      relativeOrientation: {
        heading: 'left',
        heading: 'right'
      }
    }
    isParked: false,
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection'
      },
      on: 'not-off-road'
    },
    regulation: 'no-stop',
    relativeOrientation: {
      longitudinal: 'front'
    }
  }

  no-vehicle {
    location: { on: 'intersection' },
    SOME-OF {
      ALL {
        beliefs: { heading: 'untrusted' },
        relativeOrientation: {
          longitudinal: 'front'
        }
      }
      ALL {
        relativeOrientation: {
          heading: 'left',
          lateral: 'not-left',
          longitudinal: 'front'
        }
      }
      ALL {
        relativeOrientation: {
          heading: 'right',
          lateral: 'not-right',
          longitudinal: 'front'
        }
      }
    }
  }
}
```

System Id : RIWS8
Decision : decelerate-to-halt
Constraint: Vehicle + End of Lane
Goal: Slow down in line before entering the intersection if some source of danger is visible.

```
ALL {
  ego {
    location: {
      approaching: 'intersection',
    }
  }
  travel: {
    regulation: 'no-stop'
  }
  a-vehicle: {
    beliefs: { heading: 'not-untrusted' },
    isLeading: true,
    location: { on: 'not-off-road' }
  }
  SOME-OF {
    a-vehicle {
      SOME-OF {
        beliefs: { heading: 'untrusted' },
        relativeOrientation: {
          heading: 'left',
          heading: 'right'
        }
      }
      isParked: false,
      location: {
        SOME-OF {
          approaching: 'intersection',
          at: 'intersection'
        },
        on: 'not-off-road'
      },
      regulation: 'no-stop',
      relativeOrientation: {
        longitudinal: 'front'
      }
    }
    a-vehicle {
      location: { on: 'intersection' },
      SOME-OF {
        ALL {
          beliefs: { heading: 'untrusted' },
          relativeOrientation: {
            longitudinal: 'front'
          }
        }
        ALL {
          relativeOrientation: {
            heading: 'left',
            lateral: 'not-left',
            longitudinal: 'front'
          }
        }
        ALL {
          relativeOrientation: {
            heading: 'right',
            lateral: 'not-right',
            longitudinal: 'front'
          }
        }
      }
    }
  }
}
```

System Id : RIWS9
Decision : decelerate-to-halt
Constraint: End of Lane + Vehicle
Goal: Wait in line until the intersection is safe if some source of danger is visible.

```
ALL {
  ego {
    location: {
      at: 'intersection'
    },
    speed: <= THRESHOLD
  }
  travel: { regulation: 'no-stop' }
  a-vehicle: {
    beliefs: { heading: 'not-untrusted' },
    isLeading: true,
    location: { on: 'not-off-road' }
  }
  SOME-OF {
    a-vehicle {
      SOME-OF {
        beliefs: { heading: 'untrusted' },
        relativeOrientation: {
          heading: 'left',
          heading: 'right'
        }
      }
      isParked: false,
      location: {
        SOME-OF {
          approaching: 'intersection',
          at: 'intersection'
        },
        on: 'not-off-road'
      },
      regulation: 'no-stop',
      relativeOrientation: {
        longitudinal: 'front'
      }
    }
    a-vehicle {
      location: { on: 'intersection' },
      SOME-OF {
        ALL {
          beliefs: { heading: 'untrusted' },
          relativeOrientation: {
            longitudinal: 'front'
          }
        }
        ALL {
          relativeOrientation: {
            heading: 'left',
            lateral: 'not-left',
            longitudinal: 'front'
          }
        }
        ALL {
          relativeOrientation: {
            heading: 'right',
            lateral: 'not-right',
            longitudinal: 'front'
          }
        }
      }
    }
  }
}
```

System Id : RIWS10
Decision : decelerate-to-halt
Constraint: LTAP
Goal: Slow down to the incoming traffic before starting a left turn.

```
ALL {
  ego {
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection'
      }
    },
    navigation: 'left'
  }

  travel: {
    regulation: 'no-stop'
  }

  SOME-OF {
    a-vehicle {
      isParked: false,
      location: {
        SOME-OF {
          approaching: 'intersection',
          at: 'intersection'
        },
        on: 'not-off-road'
      },
      regulation: 'no-stop',
      relativeOrientation: {
        heading: 'towards-ego',
        longitudinal: 'front'
      }
    }

    a-vehicle {
      location: {
        on: 'intersection'
      },
      relativeOrientation: {
        heading: 'towards-ego',
        longitudinal: 'front'
      }
    }
  }
}
```

System Id : RIWS11
Decision : decelerate-to-halt
Constraint: LTAP
Goal: Slow down to the incoming traffic nearby the intersection center before starting a left turn.

```
ALL {
  ego {
    location: { on: 'intersection' },
    navigation: 'left'
  }
  travel: {
    wasRegulatedByIntersectionStop: false
  }
  SOME-OF {
    a-vehicle {
      SOME-OF {
        beliefs: { heading: 'untrusted' },
        relativeOrientation: {
          heading: 'towards-ego'
        }
      }
      distance: {
        x: <= THRESHOLD,
        y: <= THRESHOLD
      },
      isParked: false,
      location: {
        approaching: 'intersection',
        on: 'not-off-road'
      },
      regulation: 'no-stop',
      relativeOrientation: {
        longitudinal: 'front'
      }
    }
    a-vehicle {
      SOME-OF {
        beliefs: { heading: 'untrusted' },
        relativeOrientation: {
          heading: 'towards-ego'
        }
      }
      isParked: false,
      location: {
        at: 'intersection',
        on: 'not-off-road'
      },
      regulation: 'no-stop',
      relativeOrientation: {
        longitudinal: 'front'
      }
    }

    a-vehicle {
      SOME-OF {
        beliefs: { heading: 'untrusted' },
        relativeOrientation: {
          heading: 'towards-ego'
        }
      }
      location: { on: 'intersection' },
      relativeOrientation: {
        longitudinal: 'front'
      }
    }
  }
}
```

System Id : RIWS12
Decision : decelerate-to-halt
Constraint: LTAP + Vehicle
Goal: Slow down in line to the incoming traffic before starting a left turn.

```
ALL {
  ego {
    location: {
      SOME-OF {
        approaching: 'intersection',
        at: 'intersection'
      }
    },
    navigation: 'left'
  }

  travel: {
    regulation: 'no-stop'
  }

  a-vehicle: {
    beliefs: {
      heading: 'not-untrusted'
    },
    isLeading: true,
    location: {
      on: 'not-off-road'
    }
  }

  SOME-OF {
    a-vehicle {
      isParked: false,
      location: {
        SOME-OF {
          approaching: 'intersection',
          at: 'intersection'
        },
        on: 'not-off-road'
      },
      regulation: 'no-stop',
      relativeOrientation: {
        heading: 'towards-ego',
        longitudinal: 'front'
      }
    }

    a-vehicle {
      location: {
        on: 'intersection'
      },
      relativeOrientation: {
        heading: 'towards-ego',
        longitudinal: 'front'
      }
    }
  }
}
```

System Id : RIWS13
Decision : decelerate-to-halt
Constraint: LTAP + Vehicle
Goal: Slow down in line to the incoming traffic nearby the intersection center before starting a left turn.

```
ALL {
  ego {
    location: {
      on: 'intersection'
    },
    navigation: 'left'
  }

  travel: {
    wasRegulatedByIntersectionStop: false
  }

  a-vehicle: {
    beliefs: {
      heading: 'not-untrusted'
    },
    isLeading: true,
    location: {
      on: 'not-off-road'
    }
  }

  SOME-OF {
    a-vehicle {
      SOME-OF {
        beliefs: {
          heading: 'untrusted'
        },
        relativeOrientation: {
          heading: 'towards-ego'
        }
      }
      distance: {
        x: <= THRESHOLD,
        y: <= THRESHOLD
      },
      isParked: false,
      location: {
        approaching: 'intersection',
        on: 'not-off-road'
      },
      regulation: 'no-stop',
      relativeOrientation: {
        longitudinal: 'front'
      }
    }

    a-vehicle {
      SOME-OF {
        beliefs: {
          heading: 'untrusted'
        },
        relativeOrientation: {
          heading: 'towards-ego'
        }
      }
      isParked: false,
      location: {
        at: 'intersection',
        on: 'not-off-road'
      },
      regulation: 'no-stop',
      relativeOrientation: {
        longitudinal: 'front'
      }
    }

    a-vehicle {
      SOME-OF {
        beliefs: {
          heading: 'untrusted'
        },
        relativeOrientation: {
          heading: 'towards-ego'
        }
      }
      location: {
        on: 'intersection'
      },
      relativeOrientation: {
        longitudinal: 'front'
      }
    }
  }
}
```