Formal Modeling and Automatic Generation of Test Cases for the Autonomous Vehicle

Wei Chen

université
PARIS-SACLAY

Formal Modeling and Automatic Generation of Test Cases for the Autonomous Vehicle

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580 : sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: Informatique
Unité de recherche: Université Paris-Saclay, UVSQ, Données et Algorithmes pour une ville intelligente et durable, 78035, Versailles, France.
Référent: Université de Versailles-Saint-Quentin-en-Yvelines

Thèse présentée et soutenue en visioconférence totale le 24/09/2020, par

Wei CHEN

Composition du jury:

Hanna KLAUDEL Professeur, Université d'Evry-Val d'Essonne	Présidente
Stephen GILMORE Professeur, The University of Edinburgh	Rapporteur
Eric LEVRAT Professeur, Université de Lorraine	Rapporteur
Stéphane GéRONIMI Ingénieur, Groupe PSA, Vélizy	Examinateur
Christian LAUGIER Directeur de recherche, INRIA Grenoble Rhône-Alpes	Examinateur
Mohamed TLIG Ingénieur de recherche, IRT SystemX	Examinateur
Leïla KLOUL Maître de conférences HDR, Université de Versailles-Saint-Quentin-en-Yvelines	Directrice

Thèse de doctorat

NNT: 2020UPASG002

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Modélisation formelle et génération automatique de cas de test pour le véhicule autonome

Mots clés: Véhicule autonome, Cas de test, Sûreté de fonctionnement, Modèles formels, Ontologie, PEPA

Résumé: Les véhicules autonomes reposent principalement sur un pilote de système intelligent pour réaliser les fonctions de la conduite autonome. Ils combinent une variété de capteurs (caméras, radars, lidars,..) pour percevoir leurs environnements. Les algorithmes de perception des ADSs (Automated Driving Systems) fournissent des observations sur les éléments environnementaux à partir des données fournies par les capteurs, tandis que les algorithmes de décision génèrent les actions à mettre en oeuvre par les véhicules. Les ADSs sont donc des systèmes critiques dont les pannes peuvent avoir des conséquences catastrophiques. Pour assurer la sûreté de fonctionnement de tels systèmes, il est nécessaire de spécifier, valider et sécuriser la fiabilité de l'architecture et de la logique comportementale de ces systèmes pour toutes les situations qui seront rencontrées par le véhicule. Ces situations sont décrites et générées comme différents cas de test.

L'objectif de cette thèse est de développer une approche complète permettant la conceptualisation et la caractérisation de contextes d'exécution pour le véhicule autonome, et la modélisation formelle des cas de test dans le contexte de l'autoroute. Enfin, cette approche doit permettre une génération automatique des cas de test qui ont un impact sur les performances et la fiabilité du véhicule.

Dans cette thèse, nous proposons une méthodologie de génération de cas de test composée de trois niveaux. Le premier niveau comprend tous les concepts statiques et mobiles de trois ontologies que nous définissons afin de conceptualiser et de caractériser l'environnement d'execution du véhicule autonome: une ontologie de l'autoroute et une ontologie de la météo pour spécifier l'environnement dans lequel évolue le véhicule autonome, et une ontologie du véhicule qui se compose des feux du véhicule et les actions de contrôle. Chaque concept de ces ontologies est défini en termes d'entité, de sous-entités et de propriétés.

Le second niveau comprend les interactions entre les entités des ontologies définies. Nous utilisons les équations de la logique du premier ordre pour représenter les relations entre ces entités.

Le troisième et dernier niveau est dédié à la génération de cas de test qui est basée sur l'algèbre des processus PEPA (Performance Evaluation Process Algebra). Celle-ci est utilisée pour modéliser les situations décrites par les cas de test.

Notre approche permet de générer automatiquement les cas de test et d'identifier les cas critiques. Nous pouvons générer des cas de test à partir de n'importe quelle situation initiale et avec n'importe quel nombre de scènes. Enfin, nous proposons une méthode pour calculer la criticité de chaque cas de test. Nous pouvons évaluer globalement l'importance d'un cas de test par sa criticité et sa probabilité d'occurrence.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Title: Formal Modeling and Automatic Generation of Test Cases for the Autonomous Vehicle

Keywords: Autonomous vehicle, Test cases, Safety analysis, Formal models, Ontology, PEPA

Abstract: Autonomous vehicles mainly rely on an intelligent system pilot to achieve the purpose of self-driving. They combine a variety of sensors (cameras, radars, lidars,..) to perceive their surroundings. The perception algorithms of the Automated Driving Systems (ADSs) provide observations on the environmental elements based on the data provided by the sensors, while decision algorithms generate the actions to be implemented by the vehicles. Therefore, ADSs are safety-critical systems whose failures can have catastrophic consequences. To ensure the safety of such systems, it is necessary to specify, validate and secure the dependability of the architecture and the behavioural logic of ADSs running on vehicle for all the situations that will be met by the vehicle. These situations are described and generated as different test cases.

The objective of this thesis is to develop a complete approach allowing the conceptualization and the characterization of execution contexts of autonomous vehicle, and the formal modelling of the test cases in the context of the highway. Finally, this approach has to allow an automatic generation of the test cases that have an impact on the performances and the dependability of the vehicle.

In this thesis, we propose a three-layer test case generation methodology. The first layer includes all static and mobile concepts of three ontologies we define in order to conceptualize and characterize the driving environment for the construction of test cases: a highway ontology and a weather ontology to specify the environment in which evolves the autonomous vehicle, and a vehicle ontology which consists of the vehicle lights and the control actions. Each concept of these ontologies is defined in terms of entity, sub-entities and properties.

The second layer includes the interactions between the entities of the defined ontologies. We use first-order logic equations to represent the relationships between these entities.

The third and last layer is dedicated to the test case generation which is based on the process algebra PEPA (Performance Evaluation Process Algebra), which is used to model the situations described by the test cases.

Our approach allows us to generate automatically the test cases and to identify the critical ones. We can generate test cases from any initial situation and with any number of scenes. Finally we propose a method to calculate the criticality of each test case. We can comprehensively evaluate the importance of a test case by its criticality and its probability of occurrence.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

Table des matières

Table des figures

Liste des tableaux

Abstract

Autonomous vehicles mainly rely on an intelligent system pilot to achieve the purpose of self-driving. They combine a variety of sensors to perceive their surroundings, such as cameras, radars and lidars. The perception algorithms of the Automated Driving Systems (ADSs) provide observations on the environmental elements based on the data provided by the sensors, while decision algorithms generate the actions to be implemented by the vehicles. Therefore, ADSs are safety-critical systems whose failures can have catastrophic consequences. To ensure the safety of such systems, it is necessary to specify, validate and secure the dependability of the architecture and the behavioural logic of ADSs running on vehicle for all the situations that will be met by the vehicle. These situations are described and generated as different test cases.

The objective of this thesis is to develop a complete approach allowing the conceptualization and the characterization of execution contexts of autonomous vehicle, and the formal modelling of the test cases. Finally, this approach has to allow an automatic generation of the test cases that have an impact on the performance and the dependability of the vehicle.

Firstly, we defined a test case as a scenario describing a specific situation with the properties of each component in the scenario. In order to conceptualize and the characterize the driving environment for the construction of test cases, we define three ontologies: a highway ontology and a weather ontology to specify the environment in which evolves the autonomous vehicle, and a vehicle ontology which consists of the vehicle lights and the control actions. Each concept of these ontologies is defined in terms of entity, sub-entities and properties. Then we use first-order logic equations to represent the relationships between the defined entities. Finally, we propose a three-layer test case generation methodology.

The first layer of the proposed generation methodology includes all static and mobile concepts of our ontologies, while the second layer includes the interactions between the entities of these ontologies. The third and last layer is the test case generation layer which is based on the process algebra PEPA (Performance Evaluation Process Algebra). The components of a PEPA model are the entities of the ontologies and the activities are the actions performed by these entities with their occurrence rates. We developed an algorithm to generate automatically the PEPA models, which are then executed using Eclipse PEPA to obtain their steady-state probability distributions.

Our approach allows us to generate automatically the test cases and to identify the critical ones. We can generate test cases with any initial situation and any number of scenes. Finally we propose a method to calculate the criticality of each test case. We can comprehensively evaluate the importance of a test case by its criticality and its probability of occurrence.

Key Word: Autonomous Vehicle, Test Case, Ontology, Formal Method, PEPA

Chapitre 1

Introduction

Autonomous cars mainly rely on intelligent systems to achieve the purpose of self-driving. They combine a variety of sensors to perceive their surroundings, such as cameras, radars and lidars. They must evolve in an unpredictable environment and a wide context of dynamic execution, with strong interactions. The perception algorithms of the Automated Driving Systems (ADSs) provide observations on the environmental elements based on the data provided by the sensors, while decision algorithms generate the actions to be implemented by these vehicles.

ADSs are being developed to perform the primary aspects of the dynamic driving task [SAE, 2014]. Since the 1970s, the research on the autonomous vehicle became a tentancy in the industry. Lately and after years of exploration, a certain progress has been made. In early 2018, Audi expanded Traffic Light Information Vehicle-to-Infrastructure (V2I) system to Washington [Krok, 2018]. Nissan plans to continue the collaboration with NASA to adapt the NASA technology for use in their Seamless Autonomous Mobility platform [Bartosiak, 2018]. Not only is the traditional auto industry dedicated to this research domain, but other companies, such as Google and Intel, have also participated to the development of the autonomous vehicles. Waymo, which started as Google's self-driving car project, canceled the design of the steering wheel and pedals [Gain, 2017], which completely overturns the design of traditional cars.

The developed technologies are expected to prevent accidents, reduce emissions, transport the mobility-impaired and reduce driving related stress. For example, the U.S. Department of Transportation (USDOT) believes that automated vehicles can reduce crash-related deaths and injuries, improve access to transportation and reduce traffic congestion and vehicle emissions [NHTSA, 2016]. But the autonomous vehicles also raise new safety issues which are due to the emerging nature of the technology.

ADSs are safety-critical systems whose failures can have catastrophic consequences. Systematic technical errors of ADSs, for example bugs and flaws in the sensors or the data short-

comings systems, could become significant hazards akin to human errors. Fatalities caused by immature ADSs have been reported and are considered to be on the rise [BBC, 2018] [Everington, 2020]. Safety and reliability certification is a task yet to be solved.

As autonomous vehicles are integrated into our lives, and ADSs are given control of even more complex driving tasks, the need for dependable, secure systems has become acute. Like any other system that can generate potentially risky events, the autonomous vehicle must be designed to ensure the safety of its occupants and other road users. The dependability of the architecture and the behavioural logic of ADSs should be tested, verified, and validated before the autonomous vehicles equipped with these systems are on the road. This emphasizes the need of enhanced approaches and tools to assess the safety of the movements of the autonomous vehicles in dynamic and uncertain environments.

In order to guarantee the functionality and the safety of the autonomous driving system, it is necessary to validate the decisions of the algorithms for all the situations that will be met by the vehicle. The complexity of demonstrating the safety of an autonomous vehicle are related to the large number of these situations, their uncertainty, and to the on-board technologies. This makes validation by tests in real use extremely costly, even impossible in certain cases. In order to gain confidence that the safety requirements have been achieved, validating the ADSs of the autonomous vehicle through digital simulation is necessary.

SVA (Simulation of Autonomous Vehicule Safety) Project [SVA, 2016] aims to respond by digital simulation to the challenge posed by the demonstration of safety and harmlessness of the functions on board autonomous vehicles. Launched in 2015 for a period of four years in the framework of IRT SystemX, Paris-Saclay, France, the SVA project aims to address the issue of autonomous vehicle validation through digital simulation, by developing methods and tools to assist in the design and validation. The models of vehicle components and their environment should be specified, adapted or developed in order to simulate the behaviour of the vehicle in the event of a failure of one of its components and the impact on its operation due to external disturbances. SVA project applies the developed methods to the TJC (Traffic Jam Chauffeur) autonomy function, which can control the vehicle in a traffic jam situation, at a maximum speed of 70 km/h and on a separate carriageway.

The objective of this these, which is supervised in the context of the SVA project, is to develop a complete approach allowing, on the one hand the conceptualization and the characterization of the execution contexts of the autonomous vehicle, and on the other hand, the test cases modelling and generation. These test cases are generated to describe the driving situations. We are interested in an automatic generation method which allows generating test cases that have an impact on the performance and the dependability of the vehicle. Generating all possible test cases is a challenge. We focus on the test cases in the context of the highway which is of separate carriageway type. Moreover, compared to other types of roads, there are uniform specifications for highways.

1.1 Test case generation issues

Safety is a generic concept that measures the quality of service provided by a system, so that the user has justified confidence in it. This justified trust is obtained through a qualitative and quantitative analysis of the different properties of the service delivered by the system.

Since the ADSs are tested, verified and validated through digital simulation, the driving environment of the autonomous vehicle must be modeled in order to be able to generate all the possible situations the vehicle can meet in a dynamic execution context. These situations are generated as different test cases which are applied to verify the functions and the information needed to perform these functions, and the decisions of ADSs.

Manufacturers need a complete generation strategy to ensure the completeness of situations that the vehicle will meet [Kone et al., 2019]. However, as the autonomous vehicle relies on the cooperation of artificial intelligence, sensors such as radar, camera and lidar, and GPS to improve the road safety and the traffic efficiency, with the development of the technologies, these sensors provide more and more driving environment elements to ADSs. These infrastructure elements combined to the weather conditions with their own properties may lead to the combinatorial explosion of the number of the situations met by the vehicle, and consequently the scenes constituting the test cases. Therefore, generating all possible test cases becomes close to impossible.

In this thesis, we propose therefore a test case generation approach that focuses on the most representative test cases for testing and validating ADSs. This model-based approach allows us to formally model these test cases and to identify the most critical ones.

1.2 The Contribution of the thesis

Before generating the test cases, we need to deal with another challenge – the conceptual and terminological confusion in the SVA project. Different terminologies are used by project partners with different backgrounds and different needs. Moreover, some words used in the same terminology are ambiguous, some are redundant and thus have the same meaning, while a same word may have different meanings. This makes the communication between partners lack a common understanding which leads to difficulties for cooperation between them in the project, and limits the potential for re-using and sharing their works. Thus, we need a common vocabulary for all stakeholders who need to share information in the autonomous vehicle field.

To deal with this first challenge, we need to identify the key concepts and possible relationships between the elements involved in the different execution contexts, to give clear definitions of these elements. First of all, we must clarify the definition of a test case to identify without ambiguity these concepts and relationships. We define a test case as a scenario describing a

specific situation with the properties of each element in the scenario. In order to conceptualize and characterize the driving environment for the construction of test cases, we define three ontologies: a highway ontology and a weather ontology to specify the environment in which evolves the autonomous vehicle, and a vehicle ontology which consists of the vehicle lights and the control actions. Each concept of these ontologies is defined in terms of entity, sub-entities and properties. Then we use first-order logic equations to represent the relationships between the defined entities.

Based on these ontologies, we conduct a formal modelling of the test cases. We first propose to develop a methodology to model the environment of the autonomous vehicle, and generate the test cases for testing the vehicle. This test cases generation methodology consists of three-layer. The first layer involves all static and mobile concepts of our ontologies, while the second layer includes the interactions between the entities of these ontologies. The third and last layer is the test cases generation layer which is based on the process algebra PEPA (Performance Evaluation Process Algebra) [Hillston, 1994].

Finally, as we are interested in the automatic generation of test cases, we propose a method that allows the classification of these test cases in terms of their impact on the performance and the dependability of the vehicle. We propose an approach which is integrated to the third layer of our methodology, that allows generating automatically the test cases with any initial situation and any number of scenes. This method allows us to identify the critical test cases. We also propose a method to calculate the criticality of each test case to evaluate comprehensively its importance.

1.3 The thesis outline

This thesis is structured in 9 chapters:

- In Chapter 2, we present the intelligent autonomous vehicle systems with different levels of automation, the Automated Driving Systems, and the safety problems we face;
- Chapter 3 is dedicated to the state of the art in the ontologies domain and the formal methods applied in the test case generation domain;
- In Chapter 4, we propose a definition of the test case and a methodology to generate automatically test cases for the autonomous vehicle in the context of the highway;
- In Chapter 5, we define the three ontologies (highway, weather and vehicle) we use for the conceptualization and characterization of test case;
- In Chapter 6, we define the relationships between the concepts of the ontologies using the first-order logic;
- In Chapter 7, we use the Performance Evaluation Process Algebra (PEPA) for modelling the driving scenes and the transitions between them;
- In Chapter 8, we consider two case studies on which we apply our methodology;

- Chapter 9 is dedicated to the conclusions of these works and the possible future works.

[Chen and Kloul, 2020] [Chen and Kloul, 2019] [Chen and Kloul, 2018a]

Chapitre 2

Autonomous Vehicles and Safety

2.1 Introduction

Today, vehicles can be thought of as complex systems with various intelligent functions. And the relationship between these vehicles and their drivers is expected to change significantly over the next ten to twenty years. Indeed, automotive technology continues to advance, and research into automotive innovation has the potential to change our lives.

An autonomous vehicle is a vehicle that is constantly in interaction with its environment. It interprets the environment using sensors and decision algorithms, then acts according to these interpretations. Automated Driving Systems (ADSs) of the different vehicle automation levels are implemented to perform several driving tasks.

Although the autonomous vehicles are supposed to improve safety and to reduce the daily loss of life due to road traffic accidents, they are safety-critical systems whose failures can have catastrophic consequences because they bring new safety risks. It is thus necessary to ensure the safety of the autonomous vehicle to give all the road users the justified confidence in it.

This chapter is dedicated to the autonomous vehicle system and the related safety problems. We first present the intelligent autonomous vehicle systems with their different levels of automation and the ADSs in Section 2.2. Then, we discuss the safety problems we face and present the safety standards in Section 2.3. The conclusion of this chapter is given in Section 2.4.

2.2 Autonomous Vehicle

An autonomous vehicle is a vehicle capable of detecting its environment and navigating without requiring guidance or control by teleoperator [Cox and Wilfong, 1990] [Gehrig and Stein, 1999]. It relies on the cooperation of artificial intelligence, sensors such as radar, camera and lidar, and GPS to improve road safety and traffic efficiency by reducing the number of road accidents. New technologies in vehicle control systems also offer new employment opportunities in different industries to develop, manufacture and maintain them.

Two regulators covering the United States have defined levels of vehicle automation. The first agency to define these levels is the National Highway Traffic Safety Administration (NHTSA). The NHTSA published a policy on automated vehicles in May 2013 defining the automation levels from vehicles that do not have any of their control systems automated (level 0) to fully automated vehicles (level 4) [USDOT, 2013]. However, its definition of the most automated driving level was deemed too broad. Thus the Society of Automotive Engineers (SAE) International defined new levels which are based on the NHTSA's previous work. The vehicle automation levels of SAE, which replaced the NHTSA levels in October 2016, are six (6). Levels 0 to 3 of SAE are very similar to those defined by NHTSA. However, Level 4, fully automated in the definition of NHTSA, is divided into two levels, 4 and 5 in the definition of SAE. Currently, the SAE International levels are considered as standards and are defined as follows [SAE International, 2014]:

— Level 0 - No Automation: the human driver does everything. Systems at this level do not provide any automation of the dynamic driving task (DDT) but can provide warnings [Taxonomy, 2012];
— Level 1 - Driver Assistance: an automated system on the vehicle can sometimes assist the driver to perform parts of the driving task (steering, acceleration / braking);
— Level 2 - Partial Automation: an automated system on the vehicle can effectively perform parts of the driving task (steering, acceleration / braking, OEDR: Object and Event Detection and Response), while humans continue to monitor the environment driving and perform the rest of the driving task;
— Level 3 - Conditional Automation: an automated system can both perform parts of the driving task (steering, acceleration / braking, OEDR) and monitor the driving environment in some cases, but the human driver must be ready to resume control when the autonomous system requests it;
— Level 4 - High Automation: an automated system can perform the driving task and monitor the driving environment, and the human driver does not need to regain control. However, this system can only work in certain environments and under certain conditions (Ex: during a traffic jam on a highway);
— Level 5 - Full Automation: the automated system can perform all driving tasks, in all conditions and on a road where a human can legally drive a vehicle. It is no longer limited by an Operational Domain Design (ODD). The human pilot is only necessary

for the activation, deactivation and determination of waypoints and destinations.

Figure 2-1 summarizes the six levels of vehicle automation as defined in [Committee et al., 2018].

Level	Name	Narrative definition	DDT		DDT fallback	ODD
			Sustained lateral and longitudinal vehicle motion control	OEDR		
Driver* performs part or all of the *DDT						
0	**No Driving Automation**	The performance by the *driver* of the entire *DDT*, even when enhanced by *active safety systems*.	*Driver*	*Driver*	*Driver*	n/a
1	**Driver Assistance**	The *sustained* and *ODD*-specific execution by a *driving automation system* of either the *lateral* or the *longitudinal vehicle motion control* subtask of the DDT (but not both simultaneously) with the expectation that the *driver* performs the remainder of the *DDT*.	*Driver* and *System*	*Driver*	*Driver*	Limited
2	**Partial Driving Automation**	The *sustained* and *ODD*-specific execution by a *driving automation system* of both the *lateral* and *longitudinal vehicle motion control* subtasks of the *DDT* with the expectation that the *driver* completes the *OEDR* subtask and *supervises* the *driving automation system*.	***System***	*Driver*	*Driver*	Limited
***ADS* ("*System*") performs the entire *DDT* (while engaged)**			*System*	***System***	*Fallback-ready user (becomes the driver during fallback)*	Limited
3	**Conditional Driving Automation**	The *sustained* and *ODD*-specific performance by an *ADS* of the entire DDT with the expectation that the *DDT fallback-ready user* is *receptive* to *ADS*-issued *requests to intervene*, as well as to *DDT performance-relevant system failures* in other *vehicle* systems, and will respond appropriately.				
4	**High Driving Automation**	The *sustained* and *ODD*-specific performance by an *ADS* of the entire *DDT* and *DDT fallback* without any expectation that a *user* will respond to a *request to intervene*.	*System*	*System*	***System***	Limited
5	**Full Driving Automation**	The *sustained* and unconditional (i.e., not *ODD*-specific) performance by an *ADS* of the entire *DDT* and *DDT fallback* without any expectation that a *user* will respond to a *request to intervene*.	*System*	*System*	*System*	**Unlimited**

FIGURE 2-1 – Levels of driving automation

The idea of assisting drivers has led to the development of Advanced Driver Assistance Systems (ADAS) since the early 1990s [Wilson-Jones et al., 1998]. These vehicle control systems use environmental sensors (radar, laser, vision) to improve driving comfort and road safety by helping the driver to recognize and react to potentially dangerous traffic situations [Gietelink et al., 2006]. With the continuous improvement of vehicle automation, now we mainly focus on the development of Automated Driving Systems (ADSs).

ADSs contribute to the road safety by performing several tasks such as: keeping the vehicle in its lane (lateral control), intelligent regulation of the speed of movement according to the environment in front of the vehicle (longitudinal control), collision alert, etc. [Chauvel, 2008].

Some systems of the different vehicle automation levels are implemented. In the following, we present typical examples of systems for each level of vehicle automation.

Typical examples of level 0 automation system include the Forward Collision Warning (FCW) [Chen and Parikh, 2000] [Cabrera et al., 2012], the Blind Spot Active Warning System (BSW) Warning) [Miller and Tascillo, 2005], [Strumolo et al., 2007] and the Lane Departure Warning (LDW) system [Batavia, 1999] [Barickman et al., 2007].

Level 1 of vehicle automation (individual automatic functions) is basically universal. A vehicle equipped with a cruise control system (ACC: Autonomous Cruise Control) [Woll and Olds, 1996] [Ioannou and Chien, 1993] is considered to be a level 1 autonomous car. Typical examples also include an active parking assistance system with power steering, the active Lane Keeping Assistance (LKA) system Type II [Kawazoe et al., 2002] [Kawazoe et al., 2001] and a combination of ACC with the LKA Type II. LKA Type I systems refer to LKA systems which apply corrective lane steering if the vehicle has to leave the lane, and LKA Type II systems refer to LKA systems which apply corrective lane steering if the vehicle must leave the center of the lane. LKA Type III systems center the vehicle in the middle of the lane without the driver at all times of driving.

Level 2 (automatic multifunction) is more widespread. The General Motor Vehicle Safety Regulation of the European Parliament and of the Council of the European Union forces the compulsory installation of the Advanced Emergency Braking System (AEBS) [Jang et al., 2013] since 1 November 2013 for heavy vehicles (categories M2, M3, N2, N3), and since 2014 for all new vehicles [Regulation, 2009]. Typical examples also include the Traffic Jam Assist (TJA) system [Bartels et al., 2015] [Siemens and Automotive, 2005] and the Key Parking system. The TJA system helps the driver to drive monotonously in traffic jams on motorways or similar roads with speeds up to 60 km/h. The system tracks the target vehicle ahead at a safe distance and keeps the vehicle in the center of the lane. The driver can only activate the system if slow vehicles are detected at the front. The driver must monitor the system at all times and must intervene if necessary. In principle, the driver can remove his hands from the steering wheel and does not need to use the pedals. Volvo Cars' City Safety system [Distner et al., 2009], Honda's Collision Mitigation Brake System (CMBS) [Sugimoto and Sauer, 2005] and Tesla's Autopilot system [Ingle and Phute, 2016] on Model S all belong to this level.

Level 3 (restricted automation) is under development. The DISTRONIC PLUS system with Mercedes-Benz Steering Assist and *Stop&Go* Pilot can automatically track vehicles in the event of a traffic jam [Balasbramani et al., 2019]. The typical example is the TJC (Traffic Jam Chauffeur) system [Bartels et al., 2015] [Tlig et al., 2018]. The TJC system and the TJA (Traffic Jam Assist) system are similar. Once the TJC system is activated, the autonomous vehicle takes full longitudinal and lateral control when it detects a traffic jam on a road with separate carriageways. It allows the driver to divert his attention from his driving task in the specific scenario of a traffic jam on a highway, although the driver must provide fallback performance, if necessary. The driver must be able to regain control in a longer period of time if a system

recovery request occurs. So only secondary tasks with an appropriate reaction time are allowed. This system applied to Audi A8 is considered to be the first production vehicle to use level 3 autonomous technology [Stanchev and Geske, 2016].

For level 4 systems, secondary tasks with long reaction times (reading a newspaper) are allowed. Similarly, unmanned applications such as the DVP (Driverless Valet Parking) system [Paul et al., 2017] [Abisheik and Mohan, 2017] and Audi's Highway Pilot System [newsroom, 2019] which includes the TJP (Traffic Jam Pilot) system [Bartels et al., 2015] are possible. Like TJC, the TJP system allows the driver to divert his attention from his driving task in the specific scenario of a traffic jam on a highway. But istead of the driver, the system provids fallback performance if necessary. The pilot is not required to be able to regain control if a system recovery request occurs, and all secondary tasks are allowed without limitation. The Google Waymo vehicle [Wells and Weinstock, 2019] focuses on systems of levels 4 and 5, which offer greater vehicle autonomy without requiring interaction with the driver.

In this thesis, we adopt the SAE International Levels of Automation as the SVA Project, in which this thesis work lies. One of the major challenges of the SVA project is to be able to qualify the safety of autonomous vehicle decision algorithms.

2.3 Safety of Autonomous Vehicle

Like any other system that can generate potentially risky events, the autonomous vehicle must be designed to ensure the safety of its occupants and other road users.

Safety is a generic concept that measures the quality of service provided by a system, so that the user has justified confidence in it. This justified trust is obtained through a qualitative and quantitative analysis of the different properties of the service delivered by the system. Safety is often seen as one of a group of related concepts: reliability, availability, maintainability and safety (RAMS), which are used in engineering to characterize a product or system. Reliability is the ability of a system to remain constantly operational for a given period of time, while availability is the ability of a system to be operational when it is requested. Maintainability is the ability of a system to quickly return to an operational state. Thus systems whose components are very easily removable can benefit from better maintainability than the others. Finally, safety is the property of a system that does not harm people, the environment, or any asset during a whole life cycle. It is concerned about failures affecting life or a single property of the system.

The supply and use of automated vehicles are expected to improve road safety, travel times, highway and intersection capacity, fuel efficiency, emissions per kilometre, travel choices, mobility, accessibility and opportunities for sharing [Milakis et al., 2017]. The most significant anticipated benefit of increasing vehicle automation is improved safety to reduce the loss of life each day in roadway crashes. But this also raises new safety risks which are due to the emerging

nature of the technology. Following advances in robotics, autonomous vehicles are entering increasingly complex environments. These new environments pose particular problems for the concept of safety. The autonomous and secure operation of vehicles is essential for increasingly complex applications in environments with human presence. Systematic technical errors of ADSs could become significant hazards akin to human error. Appropriate solutions to assess the safety of the movements of the autonomous vehicles in dynamic and uncertain environments are essential.

The U.S. Department of Transportation (USDOT) believes that autonomous vehicles can reduce crash-related deaths and injuries, improve access to transportation and reduce traffic congestion and vehicle emissions [NHTSA, 2016]. Safety remains the number one priority for the USDOT and is the specific focus of the U.S. National Highway Traffic Safety Administration (NHTSA). NHTSA presents a Voluntary Guidance [NHTSA, 2016] which offers a non-regulatory approach to autonomous vehicle technology safety. This guidance means to support the automotive industry, the States, and other key stakeholders as they consider and design best practices relative to the testing and deployment of autonomous vehicle technologies. This guidance points out that in order to design ADS without unreasonable safety risks, the overall process should adopt and follow industry standards. autonomous vehicle safety regulation must be resolved before the autonomous vehicles equipped with Level 3 and above ADSs are on the road.

ISO 26262 [ISO, 2011] is adapted from the International Electrotechnical Commission 61508 standard [Brown, 2000]. This standard is the first comprehensive and voluntary automotive safety standard that addresses the functional safety of electrical and/or electronic (E/E) and software-intensive features in today's road vehicles [Van Eikema Hommes, 2016]. It focus on possible hazards caused by malfunctioning behaviour of E/E safety-related systems. Safety is defined as the absence of unreasonable risk in ISO 26262. Safety analyses are applied to examine the influence of faults and failures on items or elements regarding their architecture, functions and behaviour. The results of the safety analyses provide information on conditions and causes that could lead to violation of a safety goal or safety requirement [Czerny et al., 2002].

However, many safety issues are not necessarily caused by system failures. For some systems that rely on sensing external or internal environments, potentially hazardous behaviors may be caused due to the intended functionality or performance limitations of the system. The absence of unreasonable risk due to potentially hazardous behaviours related to such limitations is defined as the safety of the intended functionality (SOTIF). ISO/PAS 21448 SOTIF are defined to solve the problems of performance limitations and misuse. Performance limitations refer to the insufficiencies of the function itself and misuse is defined as the usage of the system by a human in a way not intended by the manufacturer of the system, and for which the system has insufficient performance, or is inadequate.

ISO/PAS 21448 SOTIF [ISO, 2019] is complementary to ISO 26262. Both are distinct and are concerned with complementary aspects of safety. The functional safety addressed by the

ISO 26262 series is the focus of this work.

ISO 26262 employs safety analysis, including Failure Mode and Effect Analysis (FMEA) [Stamatis, 2003], Fault Tree Analysis (FTA) [Ericson, 1999], Event Tree Analysis (ETA) [Andrews and Dunnett, 2000], hazard and operability study (HAZOP) [Kletz, 2018]. These are valid and commonly used methods in the automotive industry to evaluate the reliability, availability, maintainability and safety of ADSs [Kim, 2014] [Mehmed et al., 2014] [Becker et al., 2017].

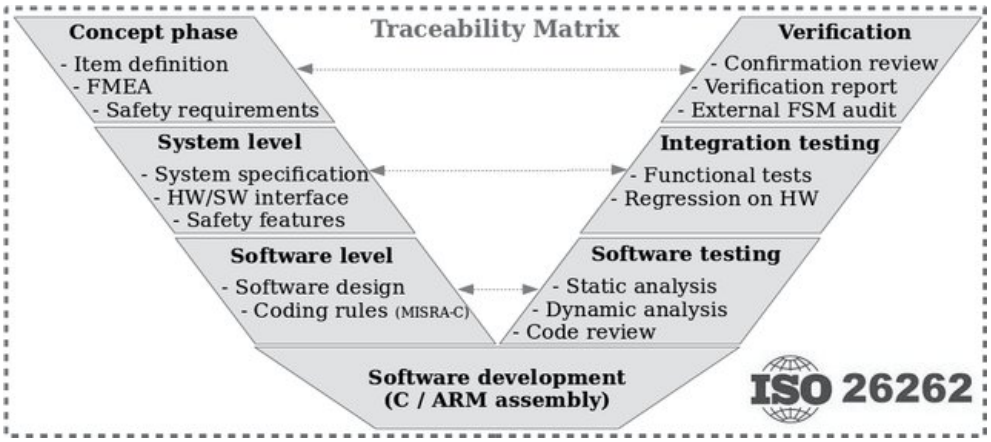

FIGURE 2-2 – V-Cycle ISO 26262 Road vehicles Functional Safety

ISO 26262 uses a classic V-Cycle framework (Figure 2-2) to organise its requirements [Lucas et al., 2018]. In the left side of the cycle, targets are models or representations of the system before the implementation. The right side of the cycle corresponds to an implementation of the system components, their integration and their validation.

The aim of the SVA project, in which this thesis work lies, is to respond by digital simulation to the challenge posed by the complexity of demonstrating the safety of an autonomous vehicle. Indeed, this complexity, which is linked both to the large number of situations that the driver encounters on the road, their uncertainty, and to the on-board technologies, makes validation by tests in real use extremely costly, even impossible in certain cases. In order to gain confidence that the safety requirements have been achieved, a model-based approach is necessary to validate the ADASs and ADSs of the autonomous vehicle.

In the SVA project, we also apply V-Cycle (Figure 2-3) to the design of the TJC system for simulation. FMECA and FTA are the tools to identify the risk areas that exist to identify the dependability requirements. The real code and the real hardware can be tested in a software-in-the-loop (SIL) simulations and a real-time hardware-in-the-loop (HIL) simulation respectively. During the development of TJC system, the initial design and specification of its controller is supported by off-line model-in-the-loop (MIL) simulations, where the controller logic is simulated in closed-loop with models of vehicle dynamics, sensors, actuators, and the traffic environment. Model-based testing approaches are used at the MIL level to automate the testing process while laying the groundwork for test coverage analysis.

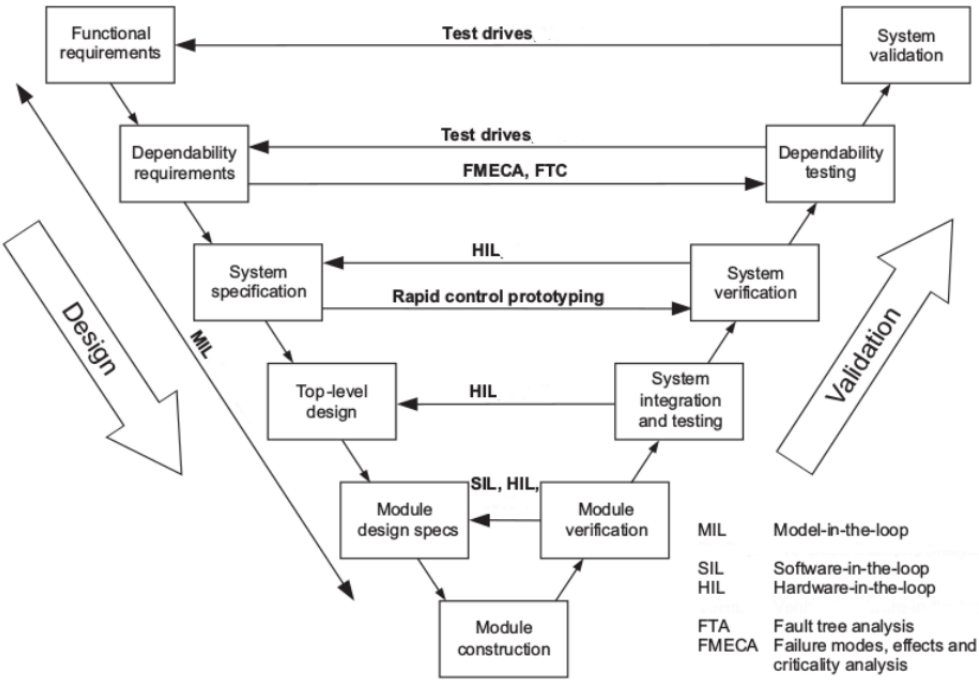

FIGURE 2-3 – V-Cycle of Autonomous vehicle development

In order to automate reasoning on the issues of the autonomous vehicle testing process, in this these we propose a formal model-based automatic generation approach of test cases.

2.4 Conclusion

In this chapter, we discussed the autonomous vehicles and the different automation levels of the corresponding ADSs. We also discussed the safety notion and its importance for the autonomous vehicle industry. Besides, we discussed the safety standard ISO 26262 which is considered in the SVA project.

All the decisions of the system should be tested, verified and validated before the autonomous vehicles equipped with ADSs are on the road. Because of the complex environments and the numerous situations that will be met by these vehicles, it appears clearly that we need methods and tools to model their environments and to generate the test cases (situations) for the testing and the validation of ADSs.

In the next chapter, we discuss the state of the art of the test cases generation methods and some related works.

Chapitre 3

State of art

3.1 INTRODUCTION

In this chapter, we discuss the state of the art of the methods and techniques used for test cases generation for the autonomous vehicle. In our knowledge, test cases generation has rarely been the subject of research works in the literature.

In [Lesemann et al., 2011], testing scenarios have been derived based on accident statistics to represent the majority of accidents in which active safety functions could possibly mitigate the outcome. The authors of [Tuncali et al., 2016] also focus on collisions. Their major objective is to find the conditions on the boundaries between safe scenarios and collision scenarios. The specific test case generation for autnomoous vehicles was developed based on the S-TaLiRo tool [Annpureddy et al., 2011], that is a MATLAB [Version, 2019] toolbox for systematic testing of hybrid systems with a focus on collisions. This method does not take into account the impact of the infrastructure and the weather conditions on the autonomous vehicle behaviours.

Many autonomous driving systems have machine learning components [Moujahid et al., 2018] [Navarro et al., 2017], which are difficult to test and verify. In [Tuncali et al., 2018], the authors present a framework for Simulation-based Adversarial Testing of Autonomous Vehicles (Sim-ATAV) to check closed-loop properties of autonomous driving systems that include machine learning components. The covering array is used to minimize the number of test cases. In [Vishnukumar et al., 2017], the authors propose a methodology using machine learning and deep neural network for testing and validating ADAS (Advanced driver assistance system) and autonomous vehicles. All these works focus on testing and validation methods, but do not give detailed test scenarios generation methodology.

In [Thorn et al., 2018], the U.S. National Highway Traffic Safety Administration (NHTSA)

proposes a framework for developing test cases and test scenarios for ADS. The core aspects of a common ADS test scenario are making up with the tactical manoeuvring behaviours, operational design domain (ODD) elements, Object and Event Detection and Responses (OEDR) capabilities and failure mode behaviours. The test scenarios are not formal since they are given by the check-lists including one or more elements of each of these core components.

In [Schuldt et al., 2018], a systematic method based on a unified 4-level model is proposed for the test case generation for advanced driver assistance systems in virtual environments. This model can be divided into four steps: system analysis, systematic test case generation, test case execution, and test case evaluation. In order to generate non-redundant, representative, unified, and reproducible test cases, the boundary value analysis is used to reduce the number of test cases. Unfortunately, test case generation is not their key research object, thus they did not give a formal method for generating these test cases.

In the above works, some contain incomplete driving environment elements [Lesemann et al., 2011] [Tuncali et al., 2016], some do not propose method for generating the scenarios [Moujahid et al., 2018] [Navarro et al., 2017] [Tuncali et al., 2018] [Vishnukumar et al., 2017], and some use generation methods which are not formal [Thorn et al., 2018] [Schuldt et al., 2018]. Unlike all these works, we propose a solution to address these problems and this solution is formal. We consider building ontologies for the context of the highway as a common vocabulary to eliminate, or at least reduce the ambiguity of used terms. Moreover, our approach allows identifying all the interactions between the different elements of our system and these interactions are expressed using the first order logic. Finally, we use the formal modelling technique PEPA (Performance Evaluation Process Algebra) which allows modelling formally the situations that can be met by the autonomous vehicle.

As we are interested in a test cases generation methodology based on ontologies and formal methods, in the following, we focus on describing the state of the art of ontologies and formal methods. In Section 3.2, we describe the state of the art of the ontologies as we use ontologies for the conceptualization and characterization of the test cases. In Section 3.3, we discuss the state of the art of the formal methods as we use a formal approach to formalize the concepts and the relationships defined in our ontologies. The conclusion of this chapter is given in Section 3.4.

3.2 ONTOLOGY

A conceptualization is an abstract and simplified view of the world that we wish to represent [Gruber, 1993a]. It is the process of developing and clarifying concepts with words and examples to arrive at precise verbal definitions. A formally represented knowledge base includes objects, concepts, and other entities that are considered to exist in the area of interest and the relationships that hold them [Genesereth and Nilsson, 2012].

An ontology is an explicit specification of a conceptualization. It is a structural framework for the representation of knowledge about the world or a part of it. It mainly consists of concepts and the relationships between them and denotes a common understanding of an area of interest. This understanding can reduce or eliminate conceptual and terminological confusion to help solve problems that impede communication between people, organizations, and/or software systems. Thus an ontology can function as a unifying framework for different points of view and serve as a basis for communication between people with different needs. It also allows interoperability between systems obtained by translating different modelling methods, paradigms of languages and software tools, and the advantages for the engineering system. The benefits for the engineering system are re-usability, reliability and specification [Uschold and Gruninger, 1996b].

The first time that the word ontology was mentioned in a discipline related to computer science is in a work on the foundations of data modelling [Mealy, 1967]. And then ontologies have been applied in a multitude of areas of IT. In the field of data and information modelling, ontology theories are used to solve database integration problems and provide a solid basis for the selection of modelling concepts [Milton and Kazmierczak, 2004] [Shanks et al., 2003] [Opdahl et al., 2001] [Fettke and Loos, 2003]. In engineering, ontology approaches are used to reduce disproportionate costs in software maintenance and enhance software reuse [Falbo et al., 2002a] [Falbo et al., 2002b]. In the field of artificial intelligence, traditionally, systems knowledge was defined in a strictly functional manner, in order to integrate the steps that experts in the field generally use to solve a given problem. Authors of [Clancey, 1993] proposed that the main concern of knowledge engineering is the modelling of systems in the world, not replicating how people think. This is important for building the foundation for the ontology of artificial intelligence. The ontologies of this domain have been built for engineering and technical applications [Alberts, 1994] [Borst, 1999] [Varejão et al., 2000], business modelling [Grüninger et al., 2000], chemistry [López et al., 1999], biology [Consortium, 2004], materials ceramics [Pisanelli et al., 2003] and the legal [Gangemi et al., 2003] [Sagri et al., 2004].

In autonomous vehicle domain, some researchers have used ontologies for the conceptualization and characterization of driving environment. But there is little work in the context of the autonomous vehicle itself.

In [Gregoriades, 2007] Monte Carlo sampling technique is used to sample the most likely events that can occur from the ontology of accident scenarios. Because of the differences between autonomous driving and traditional driving cars, the types of accidents can be quite different. We must not only analyse existing accidents, but also prevent accidents that have never occurred or were unexpected.

In [Hülsen et al., 2011], the authors use a description logic to describe the scenes. The first work provides a generic description of road intersections using the concepts *Car*, *Crossing*, *RoadConnection* and *SignAtCrossing*. They use description logic to reason about the relations between cars and describe how a traffic intersection situation is set up in the ontology and define

its semantics. The results are presented for an intersection with 5 roads, 11 lanes and 6 cars driving towards the intersection. This model is limited to intersections, and both infrastructure and vehicle numbers are static. Generated test cases are not enough to test a system like TJC.

In [Armand et al., 2014], an ontology of recognition for the driving assistance systems is presented. The authors define an ontology composed of concepts and their instances. This ontology includes contextual concepts and context parameters. It is able to process human-like reasoning on global road contexts. Another ontology is proposed by Pollard et al. [Pollard et al., 2013] for situation assessment for automated ground vehicles. It includes the sensors/actuators state, environmental conditions and driver's state. However, as the concepts of both ontologies have not been sufficiently subdivided, they are not enough to describe test cases allowing to simulate and validate ADSs.

Hummel et al. [Hummel et al., 2008a] propose an ontology to understand road infrastructure at intersections. The approach focuses on geometric details related to topological information at several levels. All the concepts of this ontology are introduced and organized in a hierarchical structure called taxonomy. This approach presents scene understanding frameworks based on description logic, which can identify unreasonable sensor data by checking consistency. However, road infrastructure at just the intersections is not enough to build test cases for testing the functions of ADSs.

To build a knowledge base for smart vehicles and implement different types of driving assistance systems, Zhao et al. [Zhao et al., 2015] propose three ontologies: map ontology, control ontology and car ontology. They focus on algorithms for rapid decision making for autonomous vehicle systems. They provide an ontology-based knowledge base and decision-making system that can make safe decisions about uncontrolled intersections and narrow roads. However, the authors did not consider the equipment of the road infrastructure in their map ontology, for example the traffic signs which are an important part for test cases construction.

Morignot et al. [Morignot and Nashashibi, 2012] propose an ontology to relax traffic regulation in unusual but practical situations, in order to assist drivers. An example of unusual but practical situations considered is: "*a truck stopping and unloading before you and your car's lane is delimited by a continuous line and a side-walk. After having waited for some amount of time, you might decide to cross the continuous line*". Their ontology represents the vehicles, the infrastructure and the traffic regulation for the general road. It is based on the experience of the members of the lab with driving license, not based on a texts corpus. That may be useful for modelling the concepts involved in traffic regulation relaxation, but we need more rigorous ontologies for modelling the concepts involved in general situations.

Finaly, in [Bagschik et al., 2017], the authors propose, using ontology, to create scenarios for development of automated driving functions. They propose a process for an ontology based scene creation and a model for knowledge representation with 5 layers: road, traffic infrastructure, temporary manipulation of road level and traffic infrastructure level, objects and environment. A scene is created from first layer to fifth layer. This ontology has modelled Ger-

man motorways with 284 classes, 762 logical axioms and 75 semantic web rules. A number of scenes could be automatically generated in natural language. However, the natural language is not a machine-understandable knowledge and the transformation of natural language based scenes to simulation data formats with such a huge ontology is a tremendous work.

Summarizing the existing research above, most of the ontologies proposed cover incomplete elements [Gregoriades, 2007] [Hülsen et al., 2011] [Armand et al., 2014] [Pollard et al., 2013] [Zhao et al., 2015] or only focus on particular situations [Hummel et al., 2008a] [Morignot and Nashashibi, 2012], and some use a natural language instead of formal language to construct the ontologies [Bagschik et al., 2017]. In our methodology, we build ontologies for the highway infrastructure elements, the weather conditions and traffics. Moreover, we use the first order logic to describe the relationships between the different elements of our system.

3.3 FORMAL METHODS

Formal Methods are a particular kind of mathematically rigorous techniques and tools for the specification, design and verification of software and hardware systems. The specifications used in formal methods are well-formed statements in a mathematical logic. Each step follows from a rule of inference and hence can be checked by a mechanical process [Alagar and Periyasamy, 2011]. These methods were originally developed for specifying and verifying the correct behaviour of software and hardware systems and have been applied in many system development fields, and many achievements have been made [Almeida et al., 2011].

There is a variety of formal methods for system modelling. These systems can be divided into two major categories: serial and concurrent. In this section we discuss the state of art of formal methods for the autonomous vehicles system modelling according to these two types of systems.

3.3.1 Serial system modelling techniques

In serial system, the elements are processed one at a time, each being completed before another begins. A serial system is usually considered as a function from the initial state to the termination state. It can be described by the relationship between its input and output. These systems can be modelled using techniques such as Z language [Meyer and Baudoin, 1978], The Vienna Development Method (VDM) [Bjørner and Jones, 1978] and B method [Abrial, 1988].

The **Z language** [Meyer and Baudoin, 1978] is a specification language based on predicates and Zermelo Freaenkel set theory. It is used for describing and modelling computing systems. There are two languages in Z (the mathematical language and the schema language). Mathematical language is used to describe the various characteristics of the system: objects

and their relationships. A schema language is a semi-graphical language used to construct, organize, describe, and encapsulate blocks of formal descriptions so that they can be reused. The program written in the Z language is used as an abstract design of computer software or hardware systems [Spivey, 1992].

The Vienna Development Method (VDM) [Bjørner and Jones, 1978] is a functional constructive specification technique that describes the function of each operation or function through first-order predicate logic and established abstract data types. The basic idea of VDM technology is to use abstract data types, mathematical concepts and symbols to specify the function of an operation or function, and the process of this specification is structured. This method is used for software development since it enables to briefly and clearly indicate the software system before the system is implemented [Jones, 1990].

The **B method** [Abrial, 1988] uses AMN (Abstract Machine Notation) to describe the requirements model, explain the interface, and carry out intermediate design and implementation. A complete development is a step-by-step implementation of the specification process. Step-by-step development can reduce the complexity of large-scale software development. A hierarchical approach can represent high-level implementations as low-level specifications. It makes it possible to formalize the system and its environment in an abstract way, then by successive refinements, to add the details to the model of the system. A formal proof activity makes it possible to verify the consistency of the abstract model and the conformity of each refinement with the superior model, thus proving the conformity of all concrete implementations with the abstract model.

B method has been used successfully for several industrial applications such as the development of embedded software for line 14 of the Paris metro (METEOR) which has been modelled, proven and generated from formal specifications B [Boulanger and Gallardo, 2000].

Another formal method called **Event-B** [Abrial et al., 2010] has been developed for the use with an incremental style of modelling. Event-B is considered an evolution of B. The new feature of Event-B are the introduced events which correspond to the transition labels between the abstract invariants. It has a simpler notation, which is easier to learn and use. It comes with tool support in the form of the Rodin tool [Butler and Hallerstede, 2007].

[Jarrar and Balouki, 2018] uses Event-B as a formal modelling and verification method to guarantee bugs absence and ensure the consistency of the system by means of invariant preservation and deadlock freedom for the air traffic control system.

In our knowledge, all these methods are mainly used for formal specification and software development. They have not been used in autonomous vehicles area.

3.3.2 Concurrent system modelling techniques

Concurrent systems are much more complicated than serial systems. Autonomous vehicle belongs to this category of systems. A concurrent system allows all the tasks to make progress to support more than one task [GALVIN and GAGNE, 2005]. It consists of a collection of processes communicating through shared data structures or objects [Παλαιοδήμος, 2018]. There exist several types of modeling technique for concurrent systems, such as Markov processes, Petri Nets [Petri, 1962] and Process Algebra [Bergstra et al., 2001].

In our knowledge, these techniques have very few applications in autonomous vehicles domain. In the following, we discus these applications when they exist.

3.3.2.1 Markov processes

A Markov process is a stochastic process with the Markov property. The conditional probability distribution of future states of the process depends only on the present state, and not on any past states [Kemeny and Snell, 1976]. That means the future and past states are independent for the present state of the system. However, Markov processes lack the notion of hierarchical system decomposition which can conquer the complexity of systems in the domain of functional system properties [Brinksma and Hermanns, 2000].

In [Althoff and Mergel, 2011] the authors compare the Monte Carlo simulation with the Markov chains according to their performance in the probabilistic prediction of road traffic scenarios. The results show that Markov chains are preferred for the probabilistic occupancy of traffic participants which helps to plan the manoeuvre of an autonomous vehicle.

The **Markov decision process (MDP)** is an optimal decision process for stochastic dynamic systems. It is a model based on Markov process theory for sequential decision making when outcomes are uncertain [Puterman, 2014]. A **partially observable Markov decision process (POMDP)** is a generalization of MDP. It is an agent decision process which permits uncertainty regarding the state of a Markov process and allows state information acquisition [Monahan, 1982].

The POMDP framework is used to model a variety of autonomous vehicles sequential decision processes [Brechtel et al., 2014] [Liu et al., 2015] [Widyotriatmo and Hong, 2008]. The authors in [Brechtel et al., 2014] present a generic approach for tactical decision-making under uncertainty in the context of driving by formulating the task of driving as a continuous POMDP that can be automatically optimized for different scenarios. POMDP is used in [Liu et al., 2015] to model the situation-aware decision making problem for autonomous driving on urban road. In [Widyotriatmo and Hong, 2008], the authors propose a decision making framework for autonomous vehicle to perform obstacle avoidance and operational task, which are achieved with respect to the nonholonomic constraint and considering the uncertainties of the autonomous

vehicle. POMDP is adopted in this decision making framework to manage the safety and task related assignment. All these POMDP based studies are aimed at optimizing environmental perception and decision frameworks, rather than ensuring the safety by testing these frameworks.

In [Raffaelli et al., 2016], test cases are generated using MaTeLo [MaTeLo, 2019], a Model Based Testing (MBT) tool, and an ad hoc random scan Gibbs sampler (RSGS) is used to cope with the combinatorial explosion of the number of scenes. It is an algorithm used to obtain a series of observation samples which are approximately equal to the multidimensional probability distribution. The following state is reached by sequentially sampling all the variables from their distribution when conditioned by the current values of all other variables and the data. However, Gibbs sampler cannot work on the mutually influential parameters of driving environment.

3.3.2.2 Petri Nets

Before the process algebra, the only part of concurrency theory that existed is the theory of **Petri nets**, conceived by Petri starting from his thesis in 1962 [Petri, 1962]. A Petri Net is a mathematical representation of a system. The basic idea is to describe state changes in a system with transitions. They are suitable for describing asynchronous, concurrent systems. The structure and dynamic behaviour of the modelled system can be analysed and used to evaluate the system finality to improve or change the system [Petri and Reisig, 2008].

A Petri Net provides a graphic representation with formal semantics of system behaviour, that is, a directed, bipartite graph where the two classes of vertices are called places and transitions. Places may contain tokens that may move to other places by firing actions. A token on a place means that the corresponding condition is fulfilled or that a resource is available [Dill et al., 2005].

In [Furda and Vlacic, 2009] and [Furda and Vlacic, 2011], the authors deal with the high-level vehicle control tasks and address the topic of real-time decision making for autonomous vehicles. There is a large number of factors to be considered in the decision making unit for the selection of feasible driving manoeuvres. Petri nets are used to model this decision stage.

[Lee et al., 2009] introduced a complete parking mechanism for autonomous car-like vehicles, such as micro electric vehicles, quadricycle, E-trick, new energy vehicles, golf car, to solve the parallel parking problem. The Petri net is used to recognize suitable parking regions and plan alternative parking routes especially in global space. This method provides an effective parking path and strategy. It also extends the case of single parking space to the case of multiple parking space.

In order to compress the repetitive structure in the classical Petri nets and improve their modeling ability, advanced Petri nets have been developped. Among them, **Coloured Petri**

nets (CPN) [Billington, 1989] are a backward compatible extension of the concept of Petri nets, which allow tokens to have a data value attached to them. This value is called token color which usually means having an object that can be identified. Thereby avoiding repeated modelling of the same structure net.

The time was introduced to Petri Nets and this led to two basic timed versions: **Timed Petri Net (TdPN)** [Ramamoorthy and Ho, 1980] and **Time Petri Net (TPN)** [Merlin and Farber, 1976]. TdPN are derived form classical Petri nets by associating a delay with each transition of the net. TPNs are more general than TdPN as each transition is associated with a time interval instead of a delay like in TdPN. Thus a TdPN can be simulated by a TPN, but the inverse is not true. They are commonly used to evaluate the performance and reliability of complex systems.

Stochastic Petri Nets (SPNs) are a modelling formalism for the description of Discrete Event Dynamic Systems (DEDS) to evaluate their performance and reliability. The dynamic behaviour of complex models could be represented by means of continuous-time homogeneous Markov chains [Kemeny and Snell, 1961]. In SPNs, changes are not immediate, but they occur only after a random delay which is a random variable exponentially distributed. Therefore, the state space of the SPN will grow exponentially as the size of the system increases.

An extension of SPN named **Generalized Stochastic Petri Nets (GSPNs**) was presented in [Marsan et al., 1984] to reduce the state space explosion. GSPNs include two classes of transitions: exponentially distributed timed transitions and immediate transitions. The former are used to model the random delays associated with the execution of activities and the latter are devoted to the representation of logical actions that do not consume time [Balbo, 2007].

In our knowledge, CPN, TdPN, TPN, SPNs and GSPNs are extension methods of Petri Net that have not been applied in the autonomous vehicle context.

3.3.2.3 Process Algebras

All process algebras (PAs) have the same fundamental basis. They are based on actions that can construct processes. Moreover, special operators may be used to compose processes to create more complex behaviours. Operators follow the algebraic laws that can be used for formal reasoning. We consider process algebra as either classical or stochastic.

a) Classical process algebra

In the late seventies, Robin Milner and Tony Hoare, respectively, proposed the **Calculus of Communicating Systems (CCS)** [Milner, 1980] and **Communicating Sequential Processes (CSP)** [Hoare, 1985]. They created a precedent for studying communication concurrency systems using algebraic methods. Since then, this research direction has flourished. After some preliminary works by others, three main classical process algebra theories were developed

[Rooda et al., 2007]. These are CCS, CSP that we mentioned and the **Algebra of Communicating Processes (ACP)** proposed by the authors in [Bergstra and Klop, 1984].

Robin Milner begins his works on the process theory CCS in 1973. After years of continuous development, the first complete process algebra with a set of equations and a semantic model was published in the book [Milner, 1980]. The expressions of the language are interpreted as a labelled transition system. Given a set of action names, the set of CCS processes is defined by BNF (Backus Normal Form) grammar.

CSS can describe parallel composition, choice between actions and scope restriction for the evaluation of the qualitative correctness of properties of a system such as deadlock or livelock [Milner, 1980]. This language is much used for the formalization of Web Services [Salaün et al., 2004] [Cámara et al., 2006].

CSP was first described in a 1978 paper by Tony Hoare [Hoare, 1978], and developed by himself in 1984 [Brookes et al., 1984] and 1985 [Hoare, 1985]. The theory of CSP itself is also still the subject of active research. CSP is a formal language for setting up and reasoning about processes that interact with their environments using this model of communication in concurrent systems [Roscoe, 1997]. The most fundamental object in CSP is therefore a communication event. These events are assumed to be drawn from a set which contains all possible communications for processes in the universe under consideration.

Both CCS and CSP are based on the notion of process. The main difference between them is that CSP has two forms of choice (internal/external or non-deterministic/deterministic). The distinction between internal and external choices allows CSP to have a semantics in terms of traces. CSP is well-suited to modelling and analysing systems that incorporate complex message exchanges. The programming language Occam [May and Taylor, 1984] arises from the concepts in CSP, which also influenced the design of programming languages such as Limbo [Dorward et al., 1997] and Go [Meyerson, 2014].

Inspired by the contributions of Milner with the basic concepts of communication and parallelism as algebraic in nature, Bergstra and Klop present the basic process algebra ACP which is an axiomatic-algebraic framework for studying processes [Bergstra and Klop, 1984]. This is the first time that the term *process algebra* is used. In ACP a process algebra is any mathematical structure, consisting of a set of atomic processes and a set of operators. ACP in many respects is similar to CCS. It emphasizes the algebraic aspect with an equational theory with a range of semantical models, and it is more easily amenable to formal analysis and mathematical proof verification.

Since the objective of classical process algebra is qualitative analysis rather than quantitative one, activities have no connection with time, thus only the functional characteristics of the concurrent system can be described, and they can only qualitatively analyse the system, not quantitatively. Therefore, quantitative analysis parameters, such as time and probability, are added to PAs, which have evolved into timed process algebras and probabilistic process algebras,

respectively.

Timed process algebras like **TCCS** [Moller and Tofts, 1990] and **Timed CSP** [Reed and Roscoe, 1988], allow each activity to be associated with an execution time to analyse the model of the real-time system. In [Bergstra and Middelburg, 2005], a process algebra with continuous relative timing was proposed to deal with the behaviour of hybrid systems (the systems exhibit both discrete and continuous behaviours), such as the water-level monitor, the thermostat, the nuclear reactor, etc.

A probabilistic process algebra allows each activity to be associated with an implementation probability, eliminating the non-determinism of the selection operation in the process algebra. A process algebra with the probabilistic transition system was presented in [Adão and Mateus, 2007] for specifying and reasoning about quantum security protocols. Probabilistic extensions of process algebras, such as **Probabilistic Calculus of Communicating Systems (PCCS)** [Giacalone et al., 1990], which are based on Milner's **Synchronous Calculus of Communicating Systems (SCCS)** [Milner, 1983], use a probabilistic choice operator to allow uncertainty to be quantified.

In our knowledge, these approaches have not been applied to the domain of autonomous vehicles. A process algebra based on basic operators of classical process algebras (CCS, CSP, ACP) is used in [Varricchio et al., 2014] as a formal specification language to express complex tasks for autonomous electric vehicles in a mobility-on-demand scenario. The authors proposed an algorithm whose solution converges to the optimal continuous-time trajectory that satisfies the task specification.

b) Stochastic Process Algebra (SPA)

Timed process algebra and Probabilistic process algebra are the basis for proposing stochastic process algebra. SPA was first proposed as a tool for analysing the performance and reliability of parallel and distributed systems in 1990 [Herzog, 1990]. Based on PA, SPAs add the continuous-time random variables to represent time instants as well as durations of activities. The action and a random delay are considered as a single entity in SPA like **Timed Processes and Performability evaluation (TIPP)** [Gotz et al., 1992], **Performance Evaluation Process Algebra (PEPA)** [Hillston, 1994] and **Extended Markovian Process Algebra (EMPA)** [Bernardo et al., 1995], while they are considered as separate entities in the timed process algebras. The syntaxes of TIPP, PEPA and EMPA are similar. The only difference is the representation of the duration of every action. TIPP uses general distribution, while PEPA uses only exponentially distribution. EMPA is inspired from PEPA and TIPP as it includes exponentially timed actions and immediate actions. SPAs are mainly used for modelling the performance of parallel and distributed systems, analysing quantitatively and qualitatively the dynamic behaviour of resource-sharing systems [Clark et al., 2007].

Performance Evaluation Process Algebra (PEPA) is a stochastic process algebra designed for modelling computer and communication systems introduced by Jane Hillston in the

1990s [Hillston, 1994]. PEPA is a simple language with a small set of operators. It is easy to reason about the language and provides a great deal of flexibility to the modeller [Hillston, 1994].

A PEPA model is constructed by identifying components performing activities which are abstracted into a continuous-time Markov process. The generation of this underlying Markov process is based on the derivation graph of the model. The derivation graph is a directed multi-graph whose set of nodes consists of the reachable states of the model and whose arcs represent the possible transitions between them. The edges of the corresponding Markov Chain are labelled only by the rates of the activities which become the corresponding entries in the infinitesimal generator matrix.

With the exception of PEPA, the other SPAs have not been applied to the domain of autonomouos vehicles. Indeed, in [Cerone and Zhao, 2013] the authors use the Markovian process algebra PEPA to describe quantitative aspects of driver behaviour to understand the relation between driver behaviour and transport systems. A three-way junction consisting of a two-way main road with a diverging one-way road is used as an example to illustrate their approach. They are interested in the probability of possible collisions, the average waiting time in a queue from arrival at the junction to finally passing the junction and the average number of cars waiting in a queue. They have modelled the effects of driver's experience in terms of state transitions associated with a finite number of pre-defined probability factors. The results show a trade-off between junction performance (reflected in number of cars in a queue and waiting time) and safety (reflected in probability of possible collision) under certain conditions on driver behaviour.

3.4 Conclusion

In this chapter, we discussed the state of art of the autonomous vehicles test cases generation methods. We focused on describing the state of the art of the ontologies and the formal methods used in this domain. As we have seen, there are few works dedicated to formal model-based approaches that allow generating test cases.

Some formal methods are suitable for modeling serial systems [Meyer and Baudoin, 1978] [Bjørner and Jones, 1978] [Abrial, 1988] [Abrial et al., 2010]. Since the autonomous vehicle is a concurrent system, these methods are not applicable. Markov processes, Petri Nets [Petri, 1962] and Process Algebra [Bergstra et al., 2001] and other formal methods suitable for concurrent systems have very few applications in autonomous vehicles domain. Compared with Markov processes and Petri Nets, Process Algebra can model more complex systems in a simpler way. PEPA can identify all components performing activities in our system with a small set of operators. These components and their interactions are abstracted into a continuous-time Markov process. We can use existing tools (Eclipse PEPA [Hillston and Gilmore, 2014]) to generate this underlying Markov process.

In the next chapter, we present our definition of the test case for the autonomous vehicle, and the test case generation methodology we propose.

Chapitre 4

Test Cases Generation Methodology

4.1 Introduction

In recent decades, activities in the field of autonomous vehicle have produced various development tools and methodologies to manage increasing complexity and testing requirements [Tian et al., 2018] [Bhat et al., 2018] [Schätz et al., 2015] [Wongpiromsarn et al., 2011] [Keviczky et al., 2006]. Each of them has success criteria for analysis and evaluation. In order to guarantee the functionality and safety of the Automated Driving Systems (ADSs), it is necessary to validate the decisions of the algorithms for all the situations that will be met by the vehicle.

However, the large number of the highway infrastructure elements combined to the weather conditions with their own properties may lead to the combinatorial explosion of the number of the situations met by the vehicle, and consequently the scenes constituting the test cases. Therefore, generating all possible test cases becomes close to impossible. Therefore, we focuses on the most representative situations for testing and validating ADSs.

These situations are described and generated as different test cases, which are applied to identify the functions, and the information needed to perform these functions and the decisions of ADSs. Test cases are critical to assess the safety of ADS in a variety of operating environments and road conditions. They should be firstly specified to simulate and test the autonomous vehicle or its modules.

In this chapter, we propose a test case definition in Section 4.2. In Section 4.3, we present a running example "*Insertion of vehicle by the right entrance lane of a highway*". In order to generate automatically the test cases for the autonomous vehicle, we propose a genaration

methodology with a three layers hierarchy in Section 4.4. Both the test case definition and the methodology are illustrated through the running example, which will also be used in the following chapters. Finally, we conclude this chapter in Section 4.5.

4.2 Test Case

In [Ulbrich et al., 2015], the authors present a definition of interfaces for the design and test of functional modules of an automated vehicle. They define a use case in terms of scenarios and scenes (Figure 4-1). A scene describes a snapshot of the environment including the scenery and the dynamic elements, as well as all actors' and observers' self-representations, and the relationships between those entities. A scenario describes the temporal development between several scenes in a sequence of scenes. It contains scenes, *actions* & *events* and *goals* & *values*. A use-case entails a description of the functional range and the desired behaviour, the specification of system boundaries, and the definition of one or several usage scenarios.

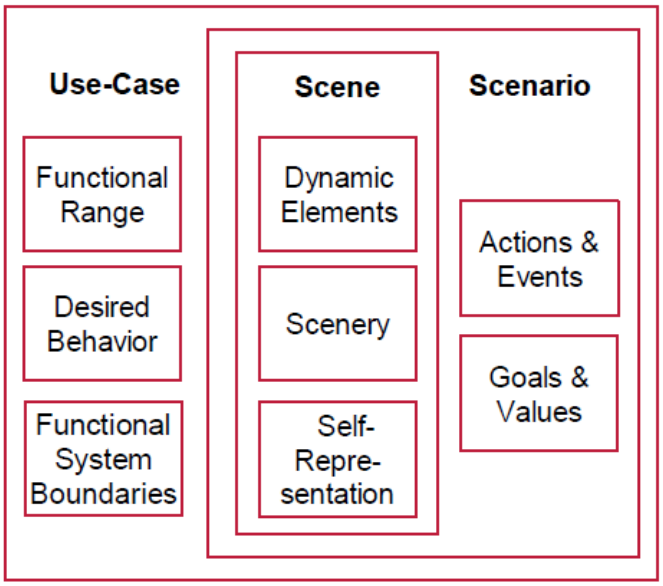

FIGURE 4-1 – Use-Case according to [Ulbrich et al., 2015]

We also define the test case in terms of scenario and scene [Chen and Kloul, 2018b] (Figure 4-2). In our definition, however, a scene describes a snapshot of the environment including static entities and mobile entities, as well as the relationships between these entities. This definition allows us to avoid ambiguity of the actors' and observers' self-representations. In [Ulbrich et al., 2015], the authors do not explain and do not provide what are the functional range, the desired behaviour and the specification of the system boundaries. Therefore, we define a test case as follows:

A **test case** describes a specific driving environment for the autonomous vehicle. It consists

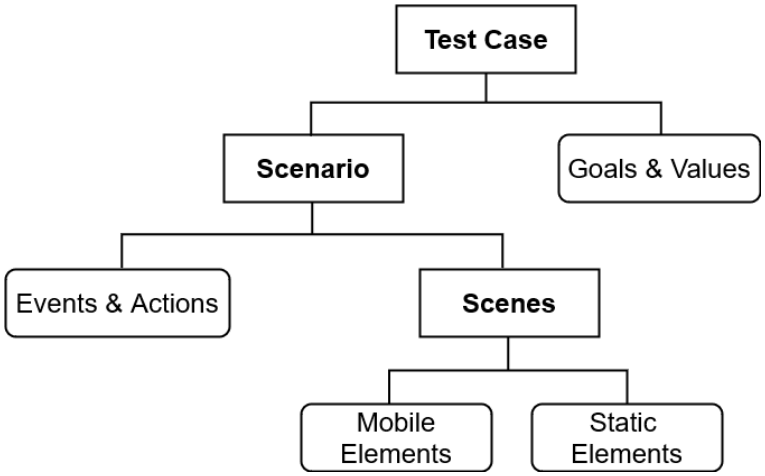

FIGURE 4-2 – The Test Case structure.

of a scenario describing a specific situation for which values are assigned to the properties of each element in the scenario. The choice of these values depends on the goal of each test case.

A **scenario** describes the temporal development between several scenes in a sequence of scenes. It is associated with the actions of all the elements in the sequence of scenes. We distinguish between two types of actions, those made by the autonomous vehicle and those made by the environment elements, which are considered as events from the point of view of the autonomous vehicle. The changes in infrastructure (ex: appearance of an entrance lane) and in the whether (ex: the start of the rain) are also considered to be events.

A **scene** is a snapshot of the vehicle environment including the static and the mobile elements, and the relationships between those elements. Static elements refer to all geospatially fixed elements which include the infrastructure of the highway and the weather conditions. Mobile elements are elements that move or have the ability to move. They include the autonomous vehicle and the other traffics. Some of the static elements, such as the lighting and the weather, can change state but not their position. We call them dynamic elements, in order to distinguish them from the mobile ones.

Figure 4-3 provides an illustration of a scenario. In this graph, the vertices are the scenes and the edges represent the events or the actions, which occurrence leads to the transition from one scene to another. Each scenario begins with an initial scene and covers a certain period of time.

An example of scenario is illustrated in Figure 4-4. This scenario consists of four scenes. There are three lanes in the initial scene *Scene* 1. It has a red car on the left lane, and a blue car on the center lane. The red car makes the action *goRightLane* which leads to *Scene* 2, where the red car is in front of the blue one. Then it does the action *decelerate* which leads to *Scene* 3. Finally, the red car makes again the action *goRightLane* which leads to *Scene* 4 which is the last scene of this scenario. The red car is on the right lane.

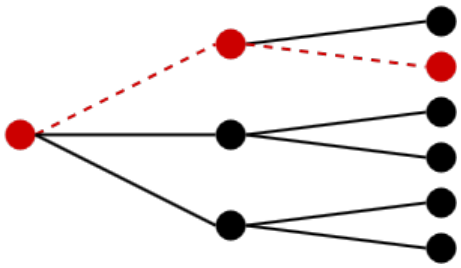

FIGURE 4-3 – A scenario (red dashed line) made by actions/events (edges) and scenes (nodes)

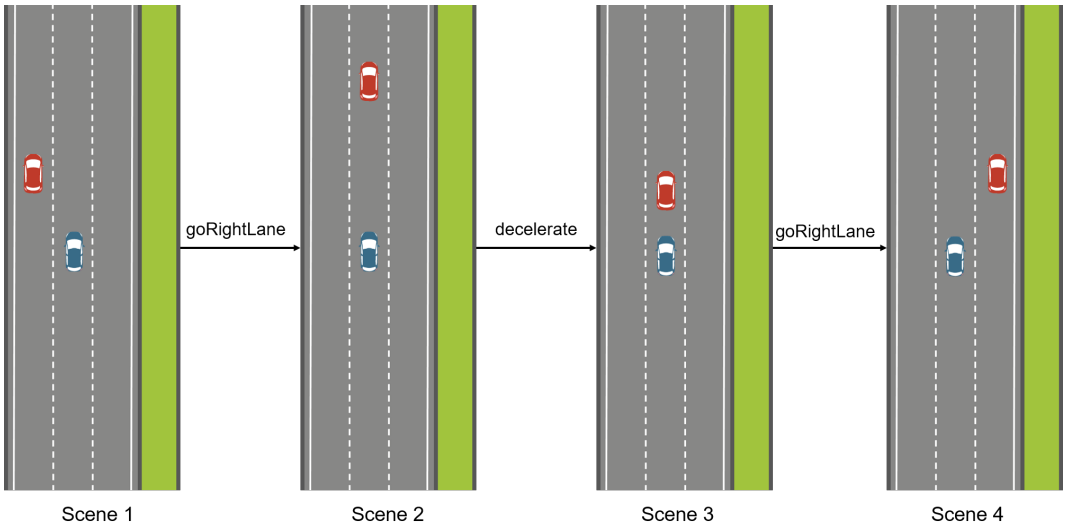

FIGURE 4-4 – A scenario representation

4.3 A Running Example

Before we explain our test cases generation methodology, we first present a running example of the situation "*Insertion of vehicle by the right entrance lane of a highway*" which will be used in the following.

It is daylight and the temperature is c °C. The humidity is h % and the pressure is p mPa. The wind speed is v_w km/h and its direction is d_w ° (from 0 to 360 °, 180 ° refers to a southerly wind).

The highway is separated into two carriageways by a median. In the scenography of this running example (Figure 4-5), a portion of one carriageway is selected. The left hard shoulder is located on the immediate outside of the median. The edge of the left hard shoulder is marked by two single solid white lines. This carriageway has three through lanes and an entrance lane. There is a chevrons marking placed between the outside lane and the entrance lane. The entrance lane is composed of an acceleration section and a taper. The right soft shoulder is located on the immediate outside of the right hard shoulder. In the beginning of the acceleration

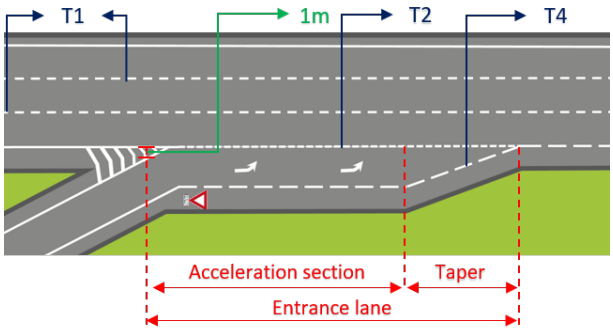

FIGURE 4-5 – Scenography of the running example.

section, a give way sign is placed on the right soft shoulder. There are two deflection arrows markings on the acceleration section. The types of dashed lines ($T1$, $T2$, $T4$) are provided on Figure 4-5. Their definitions are those provided in the official French document for road symbols [Ministère de l'écologie, 1988].

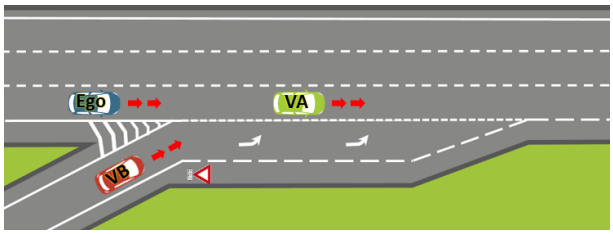

FIGURE 4-6 – Initial scene of the running example.

In the initial scene (Figure 8-2) of this example, the autonomous vehicle namely Ego (blue) rolls on the right lane of a separated lane road. The speed of Ego is given by v_e km/h on the portion which speed is limited to 130 km/h. The Traffic Jam Chauffeur (TJC) System is active and regulates the speed of Ego with respect to a target vehicle VA (green) that is located d_1 m in front of Ego. A third vehicle VB (red) arrives on the entrance lane and wants to enter the highway. VA and VB roll at a speed equal to v_1 km/h and v_2 km/h, respectively.

Here we use the natural language to describe the situation of the running example as it is the traditional way to describe test cases. In order to have a more formal description of it, we propose in the following a model-based test case generation methodology.

4.4 Test Case Generation Methodology

Few approaches have been developped for scenarios/test cases generation. Some of these approaches only focus on collision [Lesemann et al., 2011] [Tuncali et al., 2016] [Gregoriades, 2007]. Some others are limited to special situations or do not take into account several enviroment

elements, which are not enough to test ADSs [Hummel et al., 2008b] [Hülsen et al., 2011] [Armand et al., 2014] [Pollard et al., 2013], while some are not formal [Thorn et al., 2018] [Schuldt et al., 2018] [Bagschik et al., 2017].

In order to generate representative test cases with high coverage of driving environment, we should consider all the elements in the driving environment and the interactions between these elements. Moreover, in order to have reliable test cases, a formal method to generate these test cases has to be used. Therefore, in the following, we define a methodology (Figure 4-7) which consists of three layers: basic layer, interaction layer and generation layer.

The basic layer of the methodology includes the static and the mobile elements in the driving environment. This allows covering important infrastructure elements and vehicles. The actions are those made by the autonomous vehicle and the events are those made by the environment elements, which are considered as events from the point of view of the autonomous vehicle. Our method not only models the movement of vehicles, but also the appearance and disappearance of infrastructure while autonomous vehicle moves forward. The values of these elements are the values of the properties of their corresponding ontology concepts, which determine their intrinsic characteristics. The interaction layer describes the interaction relationships, between, on the one hand the static entities, and on the other hand the mobile ones. Moreover this layer describes the relationships between the static and the mobile entities. These interactions are expressed using the first order logic, which allows us to express the relationships between the different elements in our system in a simple way. Based on the concepts used in the basic layer and the formal relationships defined in the interaction layer, the generation layer build first the initial scenes, then the scenarios and the test cases from the initial. PEPA is powerful enough to model all the behaviors of the system.

More details are given in the following subsections and illustrated using the running example.

4.4.1 Basic Layer

We consider all static and mobile elements constituting the highway infrastructure, the weather and the vehicle. All these elements are represented using three ontologies (*Highway*, *Weather*, *Vehicle*) as a structural framework, which consists of a set of concepts with their definitions and relationships.

Thus, the basic layer of the methodology includes the static concepts and the mobile ones. The static concepts are those defined for the highway infrastructure and the weather while the mobile concepts are those defined for the autonomous vehicle and the other traffics. Some of the static concepts, such as the lights, can change state but not their position. They are the dynamic concepts.

In the running example, the static elements are: *daylight, temperature, humidity, pressure,*

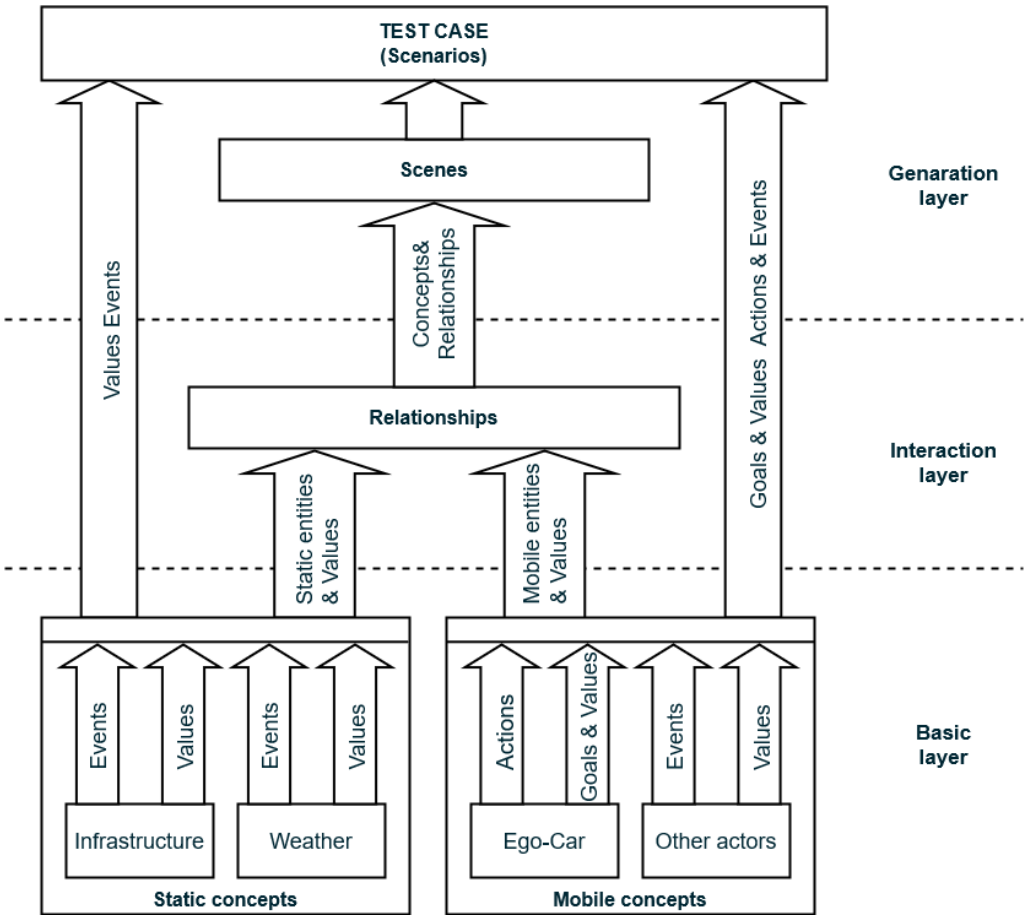

FIGURE 4-7 – Test cases generation methodology.

wind, highway, carriageway, median, soft shoulder, hard shoulder, through lane, entrance lane, acceleration section, taper, sign, marking, solid line, dashed line, chevrons marking, arrows marking. Note that the underlined static elements are the dynamic ones.

The mobile elements are autonomous vehicle *Ego*, target vehicle *VA* and vehicle *VB*.

Each static or mobile element has a corresponding concept in one of the ontologies (see Chapter 5). Each concept is defined in terms of entity, properties and sub-entities. In the basic layer (Figure 4-7), the values of Infrastructure, Weather, Ego-Car and Other actors are the values of the properties of these concepts. For example, *SolidLine* is the concept corresponding to static element *solid line* and it has the property *Color*. The value of this property in the running example is *white*.

The choice of the values depends on the goal of each test case. For example, if the goal of a test case is to test the impact of temporary road markings on the behaviour of the autonomous vehicle, the value of the property *Color* of concept *SolidLine* should be *yellow*. The values of the concepts properties allow us to determine the relationships between the corresponding

entities in the upper layer of the methodology, that is the interaction layer.

In Figure 4-7, the actions of Ego-Car are run, accelerate and decelerate. Examples of the events of Other actors are: run, accelerate, decelerate, go to left lane and go to right lane. The dynamic elements can also produce events when they change their state. In the running example, the occurrence of the entrance lane is regarded as an event.

In addition to the elements mentioned in the running example, we also consider other elements that may appear on the highway. The highway infrastructure consists of the physical components of the highway system providing facilities essential to allow the vehicle driving on the highway. We have built a highway ontology with twenty-five (25) concepts based on the French official documents [Ministère de l'écologie, 1988] [Ministère de l'équipement, 2000].

The weather describes the state of the atmosphere at a particular place and time. Some phenomena influence the visibility of the captors on the autonomous vehicle. The visibility of this one is reflected by the distance at which an infrastructure element or a vehicle can be clearly discerned. We have built a weather ontology which consists of twelve (12) concepts.

Vehicle ontology describes the performance of a vehicle with two main sub-entities: *Light* and *Action*. *Light* refers to the lights on the vehicle to illuminate the road when driving at night, or to signal other road users while *Action* refers to the control actions that could be made by the pilot. We have built a vehicle ontology which consist of fifteen (15) concepts.

All the concepts of the three ontologies are considered in the basic layer, and are presented in Chapter 5.

4.4.2 Interaction layer

The interaction layer describes the interaction relationships, between, on the one hand the static entities, and on the other hand the mobile ones. Moreover this layer describes the relationships between the static and the mobile entities.

The static concepts and the mobile concepts, which are used in the basic layer, provide the properties to build the first-order logic equations representing the interaction relationships. In the running example, we have mentioned the static elements *marking* and *sign* which have been defined in the highway ontology as concepts *Marking* and *Sign*, respectively, in the basic layer. And we have also defined a concept *Symbol* which has the property *Type* from where we learn that marking and sign are two (2) choices of values of this property. Based on the entities of their corresponding concepts and their properties values, we have defined the set *SymbolSet* which includes sets *MarkingSet* and *SignSet*. The former is the set of all possible markings and the latter is the set of all possible signs on the highway (see Chapter Cha:relations).

Thus, we can sum up using the following equation:

$$\forall x \in (MarkingSet \vee SignSet), \quad isSymbol(x)$$

where *isSymbol* is a defined relationship which states that any element x in set *MarkingSet* or *SignSet* is of type *Symbol*.

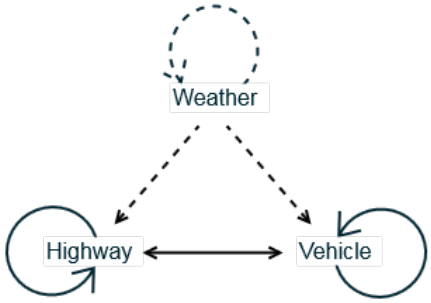

FIGURE 4-8 – Relationships (solid lines) and effects (dashed lines).

We consider the traffic regulation as rules to define the features and significance of highway infrastructure, and regulate the behaviour of the vehicles. Moreover, as the weather phenomena can have an effect on the highway, the vehicle and on itself (Figure 4-8), these effects are also written as rules. All the definitions of the relationships and the rules, which are considered in the interaction layer, are provided in Chapter 6 using first order logic.

4.4.3 Generation layer

The task of the generation layer is to build test cases. The generated scenarios are based on the concepts used in the basic layer and the formal relationships defined in the interaction layer.

Each test case consists of a scenario and the properties values of the concept corresponding to each element in the scenario. We have defined the scenario as a sequence of scenes, assailed with goals, values and actions of the autonomous vehicle, values and events from the other actors, and values of the static elements. The change of states of the dynamic elements also make events to autonomous vehicle with the change of the values of their properties. These actions and events make a scene develop to another scene. With the same initial scene, it is evident that different actions or events lead to different scenes, and make different scenarios.

Let's consider again the running example. We consider two of several possibilities: VB inserts before or after Ego.

In the first case (Figure 4-9), Ego decelerates and VB turns on the left direction lights and begins to insert before Ego. It follows that the radar of Ego detects this vehicle which becomes the new target vehicle. Ego follows VB.

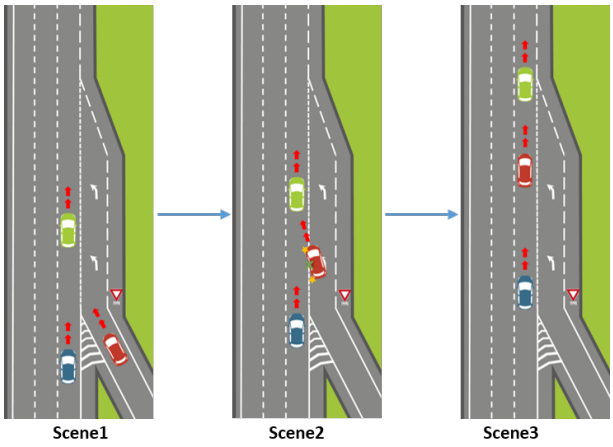

FIGURE 4-9 – Vehicle insertion before *Ego*.

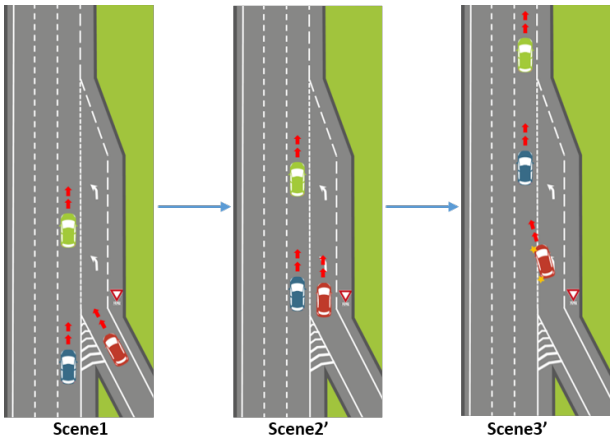

FIGURE 4-10 – Vehicle insertion after *Ego*.

In the second case, if *Ego* makes the decision to accelerate, obviously this action will lead to another scene and influence the whole scenario as showed in Figure 4-10. *VB* turns on the left direction lights and begins to insert after *Ego*. *VA* remains the target vehicle and *Ego* still follows it.

These above scenarios can be regarded as test cases when we use them with the values of the concepts' properties to test the functions of TJC system which is active and regulates the speed of *Ego*.

An autonomous vehicle is a safety-critical system for which all behaviours must be probabilistically predictable. Therefore, the generation of test cases requires the use of a semantically explicit formal language to improve their reliability and robustness. In this layer, we use the process algebra PEPA as the formal modelling technique of the test cases.

A PEPA model is constructed by identifying components performing activities. The compo-

nents are the mobile and dynamic entities of the ontologies in the basic layer, and the activities are the actions and the events performed by these entities with their occurrence rates. The PEPA model is described in Chapter 7.

4.5 Conclusion

In this chapter, we defined a test case as a scenario describing a specific situation with the values of properties for each element in the scenario. We have also introduced our test cases generation methodology with a running example.

In the next chapter, we present the three ontologies we have built for the conceptualization and characterization of test case in the basic layer. The relationships between the concepts of the ontologies used in the interaction layer are presented in Chapter 6. Finally, for the generation layer, we use PEPA for modelling the driving scenes and scenarios in Chapter 7.

Chapitre 5

Scene Conceptualization and Characterization

5.1 Introduction

Engineers and researchers with different backgrounds and different needs use different terminologies in the autonomous vehicles industry. Moreover, some words used in the same terminology are ambiguous, some are redundant and thus have the same meaning, while a same word may have different meanings. This makes communication between project partners lack a common understanding, which leads to difficulties for cooperation between them and limits the potential for re-use and sharing their works. This problem also bothers the partners of the SVA project [Project, 2015].

Because ontologies allow explicit formal specifications of the terms in a domain, in order to address the above issue, we consider building ontologies for the context of the highway to eliminate, or at least reduce conceptual and terminological confusion in the SVA project.

Sharing the common understanding is an important reason to develop ontologies. An ontology is a common understanding of an area of interest which consists mainly of a set of concepts and relationships between them [Uschold and Gruninger, 1996a]. It defines a common vocabulary for all stakeholders who need to share information in the autonomous vehicle field. It can bring together different partners and increase productivity.

Moreover, ontologies allow the reuse of knowledge not only in autonomous vehicle industry but also in other domain. For example, the ontologies built for the highway context can be used as a basis for all other types of roads. The terms of infrastructures must hold in some contexts such as road traffic, road maintenance and urban planning. When an infrastructure ontology is developed, it can be simply reused for other contexts or domains.

An ontology is an explicit specification of a conceptualization which can be realized by one or several ontologies [Gruber, 1993b]. For the conceptualization and characterization of the test cases, we need to have a common understanding of the elements involved in a scene, which is the basic element of a test case. As a scene is a snapshot of the vehicle environment including the static and the mobile elements, and the relationships among those elements, we construct three ontologies to build a semantic knowledge base of the traffic scenarios. These ontologies represent the knowledge of the road infrastructure, the weather environment and the driving control concepts of the autonomous vehicles. We conceptualize each element as an entity which is characterized by a suitable definition, a set of properties and a set of sub-entities.

Definitions are taken from dictionaries or official documents to better describe and explain the implication of the concepts. Moreover, the properties defined for each concept allow us to determine the intrinsic characteristics.

In this chapter, we introduce the ontologies we have built for the autonomous vehicle and its driving environment. A highway ontology which includes the road infrastructure is presented in Section 5.2. A weather ontology which represents visibility-related elements is presented in Section 5.3. Finally, a vehicle ontology which includes the vehicle devices and its driving control actions is presented in Section 5.4. The conclusion of this chapter is given in Section 6.4.

5.2 Highway ontology

The highway infrastructure consists of the physical components of the highway system providing facilities essential to allow the vehicle driving on the highway. It consists of all the fixed installations that have to be developed to allow the circulation of vehicles. We have built the highway ontology based on the French official documents of road development [Ministère de l'écologie, 1988] [Ministère de l'équipement, 2000]. Figure 5-1 summarizes all the concepts defined for the highway ontology.

The highway ontology consists of three main parts: the long profile of the highway (*Roadway*), which consists of *Carriageway*, *Shoulder* and *Median*, and is used to isolate the vehicles in the opposite direction to avoid scratch, the special zones of *RoadPart* including *Toll*, *Tunnle*, *Bridge* and *RoadWork*, and *Symbol* on highway. The twenty-five (25) concepts of this ontology are defined in terms of entities, sub-entities and properties which include at least the ID. For example, the concept *EntranceLane* is refereed to by entity *entrance_lane*, and is defined as "*A lane which allows vehicles wishing to access the highway to accelerate to integrate into traffic flow*". It has the properties $ID, Alignment(Horizontal\&Vertical), Length, Width$ and *SpeedLimit*. It consists of the sub-entities *Acceleration Section* and *Taper* (Table 5.1). All the concepts of this ontology are described in the following sub-sections.

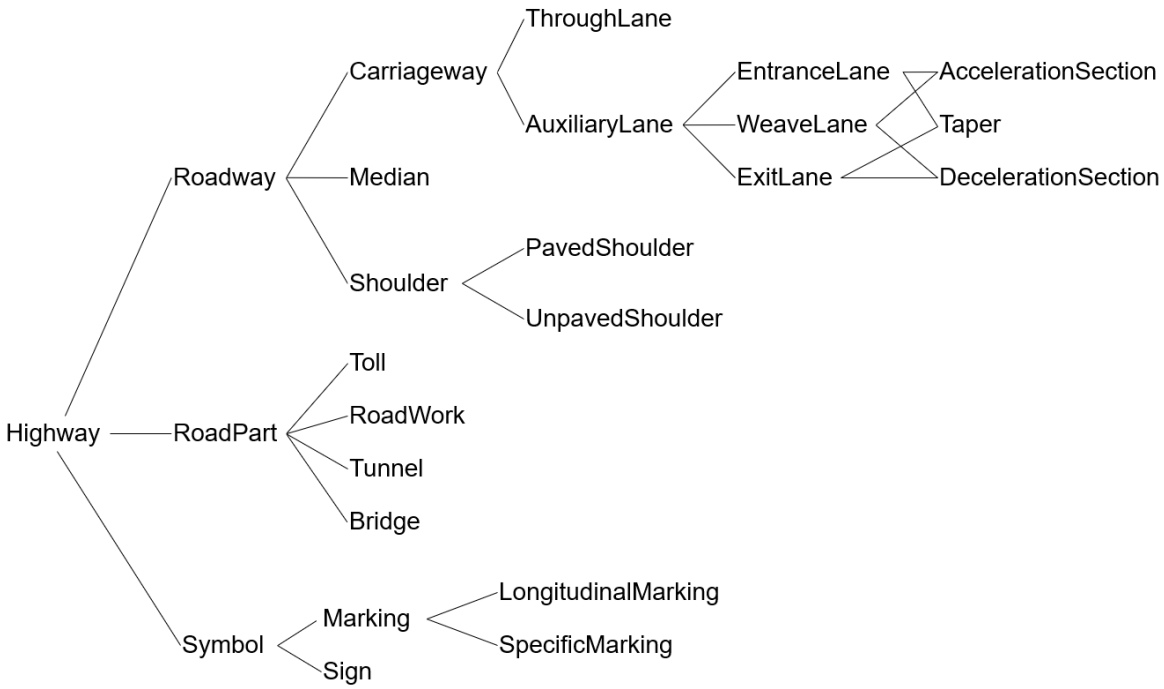

FIGURE 5-1 – Concepts of highway ontology.

TABLE 5.1 – Definition of the concept *EntranceLane*

Concept	*EntranceLane*
Entity	entrance_lane
Definition	A lane which allows vehicles wishing to access the highway to accelerate to integrate into traffic flow.
Properties	ID, Alignment (Horizontal & Vertical), Length, Width, SpeedLimit
Sub-entities	Acceleration Section, Taper

5.2.1 Highway concept

Entity: highway

Definition: highway is an entity that represents a terrestrial infrastructure with separate carriageways, each comprising at least two lanes per direction in the current section to support the circulation of wheeled vehicles.

The *Highway* is a concept characterized by the following **properties**:

- Geometry: this is the shape of the highway. We consider three types of form:
  — Line: straight highway;

  - — Spiral: helical highway;
  - — Arc: curved highway.
- Topography: The topographic slope is the tangent of the inclination between two points on the highway. We consider two types of slopes:
  - — Slope: transverse slope;
  - — Ramp: longitudinal slope.
- Length: this is the distance between the start and the end of a segment of the highway.
- Width: this is the distance between the two edges of the highway.
- Maximum speed: this is the maximum speed allowed on a highway in France. It generally depends on the location of the highway and the traffic conditions.
  - — The location of the highway
    - — Outside built-up areas, the vehicle speed is limited to 130 km/h on the highways;
    - — In built-up areas, the vehicle speed is limited to 90 km/h on urban highways.
  - — Traffic conditions: these maximum speeds are lowered to:
    - — 110 km/h on sections of highways where the normal limit is 130 km/h in the event of rain;
    - — 100 km/h on sections of highways where this limit is lower in the event of rain;
    - — 50 km/h on all highway networks in the event of visibility of less than 50 meters;
    - — 90 km/h on highways for vehicles with a total permissible laden weight greater than 3.5 tonnes or combinations of vehicles with a total permissible gross vehicle weight greater than 3.5 tonnes, with the exception of the public transport vehicles.

The highway is an entity that includes the following **sub-entity**:

- Road Part (Fr: Segment) (see Sub-section 5.2.2)

5.2.2 RoadPart concept

Entity: road_part

Definition: road_part is an entity that represents a specific part of highway, considered separately from the rest.

The *RoadPart* concept is characterized by the same **properties** as the concept *Highway* (5.2.1) and a private **property**:

- Type: this is the type of the road part.
  - — Toll (see Sub-section 5.2.3);
  - — Road Work (see Sub-section 5.2.4);
  - — Tunnel (see Sub-section 5.2.5);
  - — Bridge (see Sub-section 5.2.6).

Road_part is an entity that includes the following **sub-entity:**

- Roadway (Fr: Chaussée) (see Sub-section 5.2.7).

5.2.3 Toll concept

Entity: toll

Definition: toll is an entity that represents a segment of the highway for payment. It mainly applies to the highway network of intercity links.

The *Toll* concept is characterized by the following **properties:**

- Number: this is the number of lanes in the toll station.
- Geometry: this is the shape of the lane.
- Length: this is the distance between the start and the end of a segment of the lane.
- Width: this is the distance between the two edges of the lane.
- Maximum speed: this is the maximum speed authorized on a toll lane in France.

5.2.4 RoadWork concept

Entity: road_work

Definition: road_work is an entity that indicates construction, maintenance or improvement works on the road on one or more lanes and up to a distance of 3 m from the outside of these lanes, and this over a height of 5.5 m.

The *RoadWork* concept is characterized by the following **properties:**

- Length: this is the distance between the start and the end of a segment of the highway on which there is work.
- Width: this is the distance between the two edges of the work area on a road.

5.2.5 Tunnel concept

Entity: tunnel

Definition: tunnel is an entity that represents an underground gallery used to allow passage to a communication route.

The *Tunnel* concept is characterized by the following **properties:**

- Geometry: this is the shape of the tunnel. In general, it follows the same type of shape as the highway of which it is a part.
- Length: this is the distance between the start and the end of the tunnel.
- Width: this is the distance between the two edges of the tunnel. In general, it is identical to the highway width of which it is a part.
- Maximum speed: this is the maximum speed allowed in a tunnel.

5.2.6 Bridge concept

Entity: bridge

Definition: bridge is an entity that represents a construction making it possible to cross a depression or an obstacle by passing over this separation.

The *Bridge* concept is characterized by the following **properties:**

- Geometry: this is the shape of the bridge. In general, it follows the same type of shape as the highway of which it is a part.
- Length: this is the distance between the start and the end of the bridge.
- Width: this is the distance between the two edges of the highway.
- Maximum speed: this is the maximum speed allowed under the bridge.
- Material: this is the material from which a bridge is made. We consider three types of material:
  — Steel;
  — Brick;
  — Wood.

5.2.7 Roadway concept

Entity: roadway

Definition: roadway is an entity that represents a surface area of the highway separated by the central reservation used for vehicle traffic.

The *Roadway* concept is characterized by the same **properties** as the concept *Highway* (5.2.1)

The roadway is an entity that includes the following **sub-entities** (Figure 5-2):

- Carriageway (Fr: voies de circulation) (see Sub-section 5.2.8);
- Shoulder (Fr: Accotement) (see Sub-section 5.2.14);

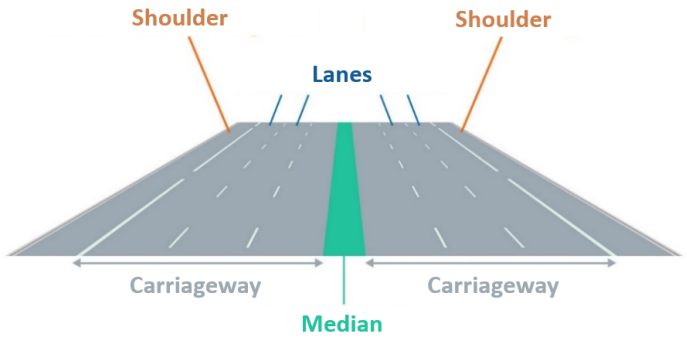

FIGURE 5-2 – Composition of Roadway.

- Median (Fr: Terre-plein central) (see Sub-section 5.2.17).

5.2.8 Carriageway concept

Entity: carriageway

Definition: carriageway is an entity that represents the paved part of it for moving traffic.

The *Carriageway* concept is characterized by the following **properties:**

- Geometry: this is the shape of the road. In general, it follows the same type of shape as the highway of which it is a part.
- Length: this is the distance between the start and the end of a segment of the carriageway. In general, it has the same length as the highway of which it is a part.
- Width: this is the distance between the two edges of the road.
- Maximum speed: this is the maximum speed allowed on a carriageway in France. It follows the same speed limit rule as on highways in France.
- Number: this is the number of lanes on a road. Each highway carriageway in France has 2 to 4 lanes.

The carriageway is an entity that includes the following **sub-entities:**

- Through Lane (Fr: Voie principale) (see Sub-section 5.2.9);
- Auxiliary Lane (Fr: Voie auxiliaire) (see Sub-section 5.2.10).

5.2.9 ThroughLane concept

Entity: through_lane

Definition: through_lane is an entity that represents a lane for the movement of the vehicles travelling from one destination to another through the traffic. It is a subdivision of the carriageway on which a single vehicle can travel in width.

FIGURE 5-3 – Types of lanes in highway.

The *ThroughLane* concept is characterized by the following **properties:**

- Geometry: this is the shape of the lane. In general, it follows the same type of shape as the carriageway of which it is a part.
- Length: this is the distance between the start and the end of a segment of the lane. In general, it has the same length as the carriageway of which it is a part.
- Width: this is the distance between the two edges of the lane. It is 3.50 m on the highway.
- Maximum speed: this is the maximum speed allowed on a lane in France. It follows the same speed limit rule as on highways in France.
- Type: this is the type of the through lane (Figure 5-3).
  - — Lane 1 (Fr: Voie 1): in relation with the direction of traffic, the rightmost lane of those assigned to the flow of traffic.
  - — Lane 2 (Fr: Voie 2): immediately to the left of Lane 1 and is reserved for traffic moving in the same direction. This lane is nearest the median if the carriageway has two lanes of traffic in the direction.
  - — Lane 3 (Fr: Voie 3): immediately to the left of Lane 2 and is reserved for traffic moving in the same direction. This lane is nearest the median if the carriageway has three lanes of traffic in the direction.
  - — Lane 4 (Fr: Voie 4): immediately to the left of Lane 3 and is reserved for traffic moving in the same direction. This lane is nearest the median if the carriageway has four lanes of traffic in the direction.

5.2.10 AuxiliaryLane concept

Entity: auxiliary_lane

Definition: auxiliary_lane is an entity that represents a lane used to separate entering, exi-

ting or turning traffic from the through traffic.

The *AuxiliaryLane* concept is characterized by the following **properties**:

- Geometry: this is the shape of the lane. In general, it follows the same type of shape as the carriageway of which it is a part.
- Length: this is the distance between the start and the end of a segment of the lane. In general, it has the same length as the carriageway of which it is a part.
- Width: this is the distance between the two edges of the lane. It is 3.50 m on the highway.
- Maximum speed: this is the maximum speed allowed on a lane in France. It follows the same speed limit rule as on highways in France.
- Type: this is the type of the auxiliary lane.
  — Entrance Lane (Fr: Voie d'insertion) (see Sub-section 5.2.11);
  — Exit Lane (Fr: Voie de sortie) (see Sub-section 5.2.12);
  — Weave Lane (Fr: Voie d'entrecroisement) (see Sub-section 5.2.13).

5.2.11 EntranceLane concept

Entity: entrance_lane

Definition: entrance_lane is an entity that represents a speed change lane whose role is to allow vehicles accessing a highway to accelerate to integrate into traffic flow.

The *EntranceLane* concept is characterized by the same **properties** as *ThroughLane* concept (see Sub-section 5.2.9).

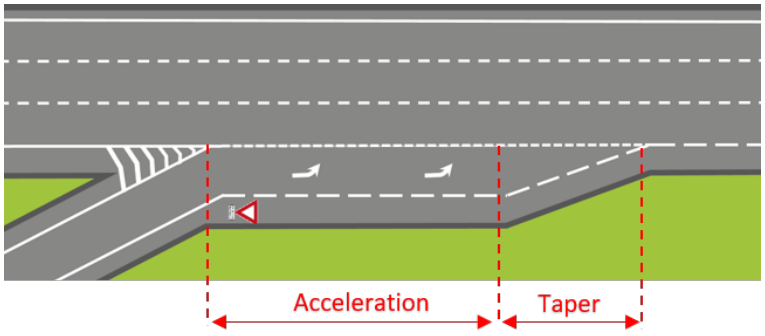

FIGURE 5-4 – Entrance Lane of highway.

The entrance lane is an entity that includes the two following sections (Figure 5-4):

— **Acceleration Section (Fr: Section d'accélération):** a section for the acceleration of the vehicle.

— **Taper Section (Fr: Biseau):** a location between *ThroughLane* and *AccelerationSection* where a driver must adjust its travel path.

5.2.12 ExitLane concept

Entity: exit_lane

Definition: exit_lane is an entity that represents a speed change lane, the role of which is to allow a vehicle wishing to leave the highway to slow down at the speed imposed by the turn encountered at the exit of the fast-flowing traffic current.

The *ExitLane* concept is characterized by the same **properties** as *ThroughLane* concept (See Sub-section 5.2.9).

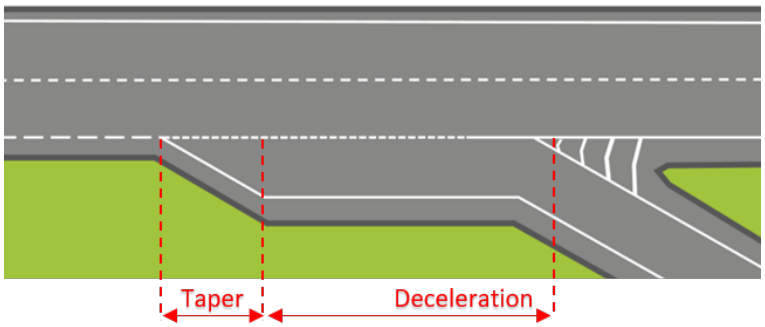

FIGURE 5-5 – Exit Lane of highway.

The exit lane is an entity that includes the two following sections (Figure 5-5):

— **Taper Section (Fr: Biseau):** a location between *ThroughLane* and *DecelerationSection* where a driver must adjust its travel path.
— **Deceleration Section (Fr: Section de décélération):** a section for the deceleration of vehicle.

5.2.13 WeaveLane concept

Entity: weave_lane

Definition: weave_lane is an entity that represents an additional lateral lane of a carriage-way, connecting a successive and close entry and exit, intended to facilitate the intersection of traffic currents which are inserted and dislocated concomitantly.

The *WeaveLane* concept is characterized by the same **properties** as *ThroughLane* concept (See Sub-section 5.2.9).

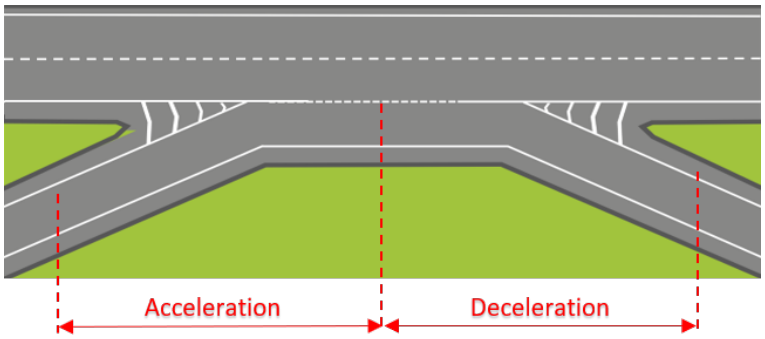

FIGURE 5-6 – Weave Lane of highway.

The weave lane is an entity that includes the two following sections (Figure 5-6):

— **Acceleration Section (Fr: Section d'accélération):** a section for the acceleration of the vehicle.

— **Deceleration Section (Fr: Section de décélération):** a section for the deceleration of the vehicle.

5.2.14 Shoulder concept

Entity: shoulder

Definition: shoulder is an entity that represents an area extending from the limit of the carriageway (in the geometric sense) to the limit of the highway.

The *Shoulder* concept is characterized by the following **properties:**

- Geometry: this is the shape of the lane. In general, it follows the same type of shape as the carriageway of which it is a part.
- Length: this is the distance between the start and the end of a segment of the lane. In general, it has the same length as the carriageway of which it is a part.
- Width: this is the distance between the two edges of the lane. It is 3.50 m on the highway.

The shoulder is an entity that includes the following **sub-entities:**

- Paved Shoulder (Fr: Bande d'arrêt d'urgence) (see Sub-section 5.2.15);
- Unpaved Shoulder (Fr: Berme) (see Sub-section 5.2.16)

.

5.2.15 PavedShoulder

Entity: paved_shoulder

Definition: paved_shoulder is an entity that represents an area located along the road and specially designed to allow, in case of absolute necessity, the stopping or parking of the vehicle. It is bounded by a broken line composed of 39 m long lines spaced by 13 m.

The *PavedShoulder* concept is characterized by the same **properties** as the *Shoulder* concept (see Sub-section 5.2.14).

5.2.16 UnpavedShoulder

Entity: unpaved_shoulder

Definition: unpaved_shoulder is an entity that represents an unpaved part of the shoulder which is not intended for use by through traffic. In addition to its normal transition function between stabilized structures and embankments or gullies, it participates in visual clearance and supports the *Sign* concept (Sub-section 5.2.22).

The Unpaved Shoulder entity is characterized by the same **properties** as the *Shoulder* concept (See Sub-section 5.2.14).

5.2.17 Median concept

Entity: median

Definition: median is an entity that represents the reserved area that separates opposing lanes of traffic on divided highways. It is used to install certain sign (Sub-section 5.2.22) and, if necessary, piles of structures and landscaping.

The Median entity is characterized by the same **properties** as the *Shoulder* concept (See Sub-section 5.2.14).

5.2.18 Symbol concept

Entity: symbol

Definition: symbol is an entity that represents a pictogram associated or not with a mention, used to identify an interchange, to pre-signal a direction prohibited to a category of vehicles, to indicate a direction recommended to a category of vehicles and to characterize a road.

The *Symbol* concept is characterized by the following **properties**:

- Color: this is the road symbol color with its meaning

  - White: this color is used for normal road markings.
  - Blue: this color is possibly used for parking limits in the blue zone.
  - Red: this color is used for the red and white checkerboard materializing the start of the distress routes.
  - Yellow: this color is used for temporary marking.
- Type: this is the type of the symbol.
  - Marking (see Sub-section 5.2.19);
  - Sign (see Sub-section 5.2.22).

5.2.19 Marking concept

Entity: marking

Definition: marking is an entity that represents a signal on the ground whose geometry, definition and use determine its legibility for the road user.

The *Marking* concept is characterized by the following **properties:**

- Color: this is the marking color with its meaning
  - White: this color is used for normal road markings.
  - Blue: this color is possibly used for parking limits in the blue zone.
  - Red: this color is used for the red and white checker-board materializing the start of the distress roads.
  - Yellow: this color is used for temporary marking.
- Width: this is the distance between the two edges of the line of the marking.
- Type: this is the type of the marking.
  - Longitudinal Marking (see Sub-section 5.2.20);
  - Specific Marking (see Sub-section 5.2.21).

5.2.20 LongitudinalMarking concept

Entity: longitudinal_marking

Definition: longitudinal_marking is an entity that designates all the lines drawn in the direction of the lanes.

The *LongitudinalMarking* entity is characterized by the following **properties:**

- Colour: this is the marking color with its meaning
  - White: this color is used for normal road markings.
  - Blue: this color is possibly used for parking limits in the blue zone.

— Red: this color is used for the red and white checker-board materializing the start of the distress roads.
— Yellow: this color is used for temporary marking.

- Width: it is defined in relation with a different unit of measurement u depending on the type of the road. On highways $u = 7.5cm$
- Type: this is the type of the longitudinal marking.
  — Solid line: a line prohibiting crossing or overlapping. It means that no driver is allowed to cross it. In addition, it is prohibited to drive on the left of a solid line, when the latter separates the two directions of traffic.
  — Dashed line: a line delimiting the lanes in order to guide traffic.

5.2.21 SpecificMarking concept

Entity: specific_marking

Definition: specific_marking is an entity that designates a special marking which gives information or a prescription.

The *SpecificMarking* concept is characterized by the following **properties:**

- Color: this is the marking color with its meaning
  — White: this color is used for normal road markings.
  — Red: this color is used for the red and white checkerboard materializing the start of the distress routes.
  — Yellow: this color is used for temporary marking.
- Type: this is the type of the specific marking.
  — Arrow: indicates the direction to follow. It may announce a continuous line or the elimination of a lane.
  — Chevrons: a set of white lines or hatching lines on which vehicles must not drive, park or stop.

5.2.22 Sign concept

Entity: sign

Definition: sign is an entity that represents a conventional signal vertically located on the road and intended to ensure the safety of road users or informing them of the dangers.

The *Sign* concept is characterized by the following **properties:**

- Shape: a panel of sign can have the shape of a triangle, square, round, rhombus or octagonal.

- Type: this is the type of the sign.
  - Type A - Danger signs;
  - Type AB - Intersection and priority signs;
  - Type B - Prescription signs subdivided into:
    - prohibition signs;
    - obligation signs;
    - end of prescription signs;
  - Type C - Signs useful for driving vehicles;
  - Type CE - Signs indicating services that may be useful to road users;
  - Types D, Dp and Dv - Tracking signs;
  - Type Dc - Local information signs;
  - Type E - Signs and identification devices for roads;
  - Type EB - Agglomeration start and end signs;
  - Type G - Position crossing signs;
  - Type H - Signs of cultural and tourist interest;
  - Type SR - Road safety information panels;
  - Types AK, K, KC and KD - Temporary traffic signs.

5.3 Weather ontology

The weather describes all the physical conditions of the atmosphere at a precise moment and at a specific point. It includes atmospheric pressure, temperature, humidity and wind. Weather phenomena such as rain and fog can occur at different time and space scales and are phenomena that can impact the autonomous vehicle behaviours. In general, the weather refers to all the parameters that influence the transport traffic.

We have defined twelve (12) concepts for the weather ontology (see Figure 5-7). Some phenomena influence the visibility of the captors on the autonomous vehicle. These concepts are: *Daylight*, *Precipitation*, *Fog* and *Haze*. As the properties of the concept *Daylight* presented in Table 5.2, the visibility of the autonomous vehicle is reflected by the distance at which an infrastructure or a vehicle can be clearly discerned. Some concepts have their properties to show the physical quantity, such as the concepts *Temperature*, *Pressure* and *Humidity*. All the concepts in this ontology are described in following sub-sections.

5.3.1 Daylight concept

Entity: daylight

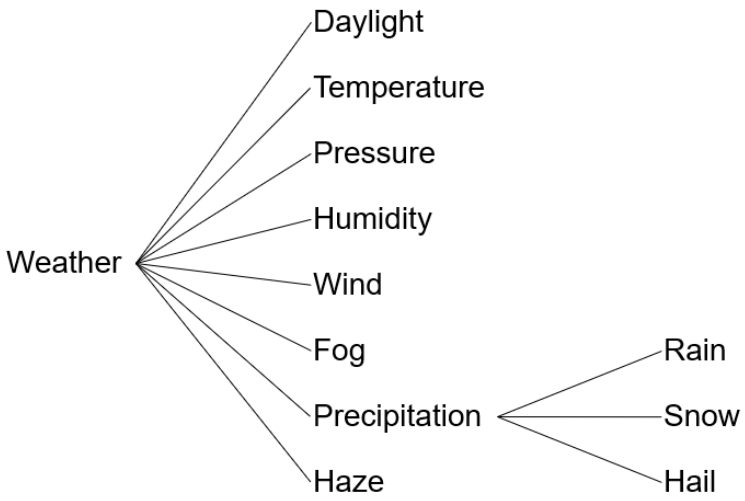

FIGURE 5-7 – Concepts of Weather ontology.

TABLE 5.2 – Definition of the concept *Daylight*

Concept	*Daylight*
Entity	daylight
Definition	The combination of all direct and indirect sunlight during the daytime.
Properties	Direction (from 0 to 360 °, 180 ° refers to south light), Brightness, Visibility

Definition: daylight is an entity that represents the combination of all direct and indirect sunlight during the daytime.

The *Daylight* concept is characterized by the following **properties:**

- Direction: from 0 to 360 °, 180 ° refers to south light
- Brightness: it is the brightness of the sun. It is represented by a real number between 0 and 1, where 0 means complete darkness and 1 means complete clarity.
- Visibility: this is the maximum visible distance for the autonomous vehicle in the daylight. It is represented in meters.

5.3.2 Temperature concept

Entity: temperature

Definition: temperature is an entity that represents the degree of freshness or heat in the atmosphere of a place.

The *Temperature* concept is characterized by the following **property:**

- Value: this is the quantitative appreciation of temperature. The unit of temperature used is the degree Celsius ($°C$).

5.3.3 Pressure concept

Entity: pressure

Definition: pressure is an entity that represents the weight that air exerts on the earth's surface.

The *Pressure* concept is characterized by the following **property:**

- Value: the unit of pressure used in meteorology is the hectopascal (hPa).

5.3.4 Humidity concept

Entity: humidity

Definition: humidity is an entity that represents the quantity of water vapor contained in the air.

The *Humidity* concept is characterized by the following **property:**

- Value: it gives the humidity in the air and is expressed in grams of water per cubic meter of air (g/m^3). It is related to the temperature as indicated in Table 5.3.

TABLE 5.3 – Maximum amount of water vapor contained in an air particle

Air temperature (en °C)	- 10	-5	0	5	10	15	20	30
Maximum amount of water vapor possible (g/m^3)	2	3	4.5	6.5	9.5	13	17	30

5.3.5 Wind concept

Entity: wind

Definition: wind is an entity that represents the horizontal movement of air.

The Wind entity is characterized by the following **properties:**

- Speed: this is the quantitative assessment of the wind. It is commonly expressed in km/h.
- Direction: this is the direction from which the wind comes. It is expressed in degrees (from 0 to 360 °). A 180 ° wind is a south wind.

5.3.6 Precipitation concept

Entity: precipitation

Definition: precipitation is an entity that represents a meteorological phenomenon associated with humidity.

The *precipitation* concept is characterized by the following **properties:**

- Visibility: this is the maximum visible distance for the autonomous vehicle during a precipitation. It is represented by a value with the meter as a unit.
- Type: this is the type of the precipitation.
  - Rain (see Sub-section 5.3.7);
  - Snow (see Sub-section 5.3.8);
  - Hail (see Sub-section 5.3.9).

5.3.7 Rain concept

Entity: rain

Definition: rain is an entity that represents a precipitation in the form of drops of water falling from the clouds to the ground.

The *Rain* concept is characterized by the following **properties:**

- Intensity: it is the qualitative and quantitative appreciation of the rain. This can be expressed in $millimeters/minute$ or $millimeters/hour$ ($1mm = 1liter/m^2$). The character of the precipitation depends on the local climatology. However, in the plain and for mainland France, the equivalences in Table 5.4 can be adopted.
- Visibility: this is the maximum visible distance for the autonomous vehicle in the rain. It is represented in meters. Each rain intensity is associated with visibility as follows:
  - Low: visibility above 1000meters;
  - Moderate: visibility between 1000 and 600 meters;
  - Strong: visibility within 600 meters.
- Speed: this is the speed at which rain falls. Its unit is the $meter/second$ (m/s).

TABLE 5.4 – Qualitative and Quantitative assessment of Rain

Qualitative assessment	Quantitative assessment
Light rain	1 to 3 mm per hour
Moderate rain	4 to 7 mm per hour
Heavy rain	8 mm per hour and more

5.3.8 Snow concept

Entity: snow

Definition: snow is an entity that represents a precipitation in the form of frozen water which falls in the form of light white flakes.

The *Snow* concept is characterized by the following **properties:**

- Intensity: it is the qualitative and quantitative appreciation of the snow. This can be expressed in $millimeters/minute$ or $millimeters/hour$ ($1mm = 1liter/m^2$). The character of the precipitation depends on the local climatology. However, in the plain and for mainland France, the equivalences in Table 5.5 can be adopted;
- Visibility: this is the maximum visible distance for the autonomous vehicle in the snow. Its unit of measurement is the meter. Each snow intensity is associated with visibility as follows:
  — Low: visibility above 1000 meters;
  — Moderate: visibility between 1000 and 600 meters;
  — Strong: visibility within 600 meters.
- Speed: it is the speed of falling snow. Its unit is the $meter/second$ (m/s).

TABLE 5.5 – Qualitative and Quantitative assessment of Snow

Qualitative assessment	**Quantitative assessment**
Light snow	Less than 1 mm per hour
Moderate snow	1 to 5 mm per hour
Heavy snow	5 mm per hour and more

5.3.9 Hail concept

Entity: hail

Definition: hail is an entity that represents a precipitation in the form of ice particles formed by the attraction and accumulation of supercooled water forming 5 to 20 layers of ice on sleet.

The *Hail* concept is characterized by the following **properties:**

- Diameter: this is the diameter of an ice particle. Its shape can be spherical, conical or irregular. It is between 5 and 150 mm.
- Visibility: this is the maximum visible distance for the autonomous vehicle under hail. Its unit is the meter. Each hail intensity is associated with visibility as follows:
  — Low: visibility above 1000 meters;
  — Moderate: visibility between 1000 and 600 meters;
  — Strong: visibility within 600 meters.

- Speed: this is the speed of falling hail. It is given in $meters/second$ (m/s). It is less than $30m/s$.

5.3.10 Fog concept

Entity: fog

Definition: fog is an entity that represents a natural phenomenon made of water vapor which forms a cloud just above the ground, hampering visibility. It is still near the surface. It is formed by droplets or ice crystals.

The *Fog* concept is characterized by the following **priority:**

- Visibility: this is the maximum visible distance for the autonomous vehicle in fog. It is measured in meters. This phenomenon reduces horizontal visibility to less than 1 $kilometer$.

5.3.11 Haze concept

Entity: haze

Definition: haze is an entity that represents a natural phenomenon made of fine droplets or fine ice crystals which have formed on microscopic hygroscopic particles (a substance which tends to absorb moisture from the air, by absorption or by adsorption). These droplets are so small that you can neither smell them nor distinguish them from each other. They usually form a continuous veil which reduces visibility on the surface.

The *Haze* Concept is characterized by the following **priority:**

- Visibility: this is the maximum visible distance for the autonomous vehicle in haze. It is measured in meters. This phenomenon reduces horizontal visibility to less than 1 $kilometer$.

We have defined twelve (12) concepts in the weather ontology to describe the weather phenomena. We only know that the weather has an impact on the driving, but we do not know what kind of impact it is, especially, when the weather conditions interact with each others to form complex effects. Generally, when the visibility is high, the vehicle travels faster and when the visibility is low, the vehicle tends to slow down. Moreover, we do not have data on the impact of the different weather conditions on the autonomous vehicle. Therefore, in this work, we use the property of visibility to model the impact of the weather on the autonomous vehicle. If the tester is an expert in this area or has more precise test requirements, other components can be added as needed. And the results of the field test can provide us with research data.

5.4 Vehicle ontology

The vehicle refers to both the classical vehicle and the autonomous vehicle. Figure 5-8 shows the fifteen (15) concepts we have defined for the vehicle ontology.

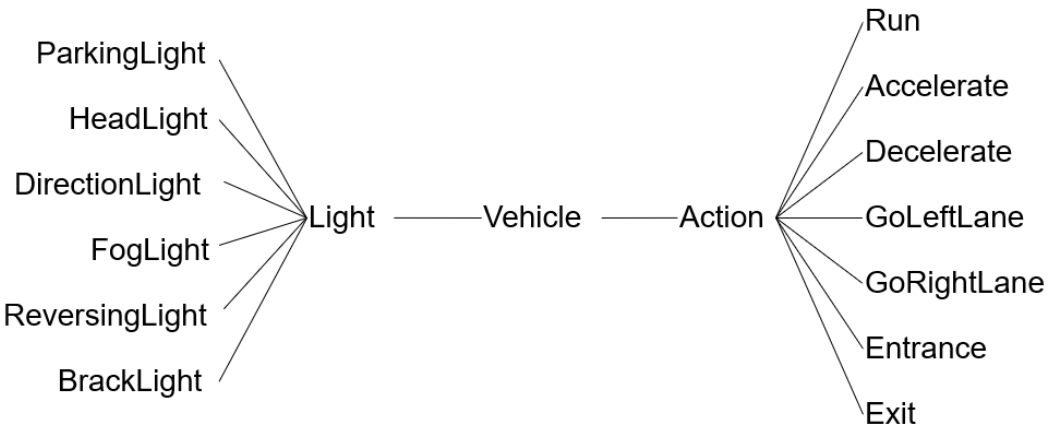

FIGURE 5-8 – Concepts of vehicle ontology.

The concept *Vehicle* consists of two main sub-entities: *Light* and *Action*. *Light* refers to the lights on a vehicle to illuminate the road when driving at night, or to signal other road users. *Action* refers to the vehicle manoeuvre actions that could be made by pilot (human drivers / intelligent system). All the concepts in this ontology are described in the following sub-sections.

5.4.1 Vehicle concept

Entity: vehicle

Definition: vehicle is an entity that represents a wheeled automobile machine. It is powered by an engine and intended for the land transport of people or goods.

The *Vehicle* concept is characterized by the following **properties:**

- Role: a vehicle can have one of the following three roles:
  — EgoCar which represents the autonomous vehicle itself;
  — TargetCar representing the vehicle targeted by the autonomous vehicle;
  — OtherCar represents one of the other vehicles in a scene.
  Table 5.6 shows the properties of the three types of vehicles according to their roles: *EgoCar*, *TargetCar* and *OtherCar*.
- Category: this is the class to which the vehicle belongs. We consider vehicles authorized on the highway. Thus, five (5) vehicle classes are defined [Ministère de l'équipement, 2000]
  — Classe1: light vehicles;

— Classe2: vehicles and truck intermediaries;
— Classe3: coaches and other heavy duty 2-axle vehicles;
— Classe4: coaches and other vehicles with 3 axles and more;
— Classe5: motorcycle, sidecar and trike.

- Height: this is the vertical distance between the roof of the vehicle or of the rolling assembly and the ground. It is given in meters (m).
- Width: this is the horizontal distance between the left end of the vehicle and the right end of the vehicle. It is given in meters (m).
- Length: this is the horizontal distance between the front of the vehicle and the rear of all vehicles (car plus trailer). It is given in meters (m). All vehicles must not exceed 18 *meters*. The maximum authorized length of a trailer is 12 *meters* (not including the coupling device).
- Weight: this is the force exerted on the ground due to the whole vehicle. It has for unit the ton (t).
- Color: it is the visual perception of the spectral distribution of visible light from the vehicle.
- Position: this is the relative position in relation to the autonomous vehicle. We consider the path on which the vehicle is located, the distance between it and the autonomous vehicle, and its location in front, next to or behind the autonomous vehicle.
- Direction: it is the direction of movement of the vehicle: the same direction of traffic, or the opposite direction of traffic.
- Speed: it is a quantity that measures the ratio of a distance to the duration of its journey. It is given in kilometers per hour (km/h).
- Acceleration: it is a quantity which represents the modification affecting the speed of a movement as a function of time. It is given in meters per second squared (m/s^2).

TABLE 5.6 – Properties of concept *Vehicle*

ID	**Ego**	**Vc$_1$**	**Vc$_2$**
Role	*EgoCar*	*TargetCar*	*OtherCar*
Category	*Class*1	*Class*1	*Class*1
Height	H_e	H_1	H_2
Width	W_e	W_1	W_2
Length	L_e	L_1	L_2
Weight	m_e	m_1	m_2
Color	*Blue*	*Green*	*Red*
Speed	v_e	v_1	v_2

The vehicle is an entity that includes the following **sub-entities:**

- Light (Fr: Feu) (see Sub-section 5.4.2);
- Action (Fr: Action) (see Sub-section 5.4.3).

5.4.2 Light concept

Entity: light

Definition: light is an entity that represents all of the automotive lighting on a vehicle to illuminate the road when driving at night, or to signal other road users.

The *Light* concept is characterized by the following **properties:**

- State: it is the lighting state of the lights (on, flashing, off).
- Position: this is where the lights are installed (front or rear, left or right).
- Number: this is a value that represents the number of lights on a vehicle (1-4).
- Color: this is the color of the lights.
- Type: we consider six (6) *Light* types:
  - Parking light: they are used to help other drivers to notice the vehicle and to judge its width. There 2 white lights at the front and 2 red lights at the rear.
  - Headlight: they are 2 large powerful white or yellow lights at the front to illuminate the road. They have two settings:
    - High-beams: the vehicle headlights are set to shine their brightest to brighten a longer distance in front of the car, than when on the low-beams setting.
    - Low-beams: that is the dimmer, shorter-range setting of vehicle's headlights to illuminate the road without dazzling other motorists.
  - Brake light: there are the red lights attached to the rear of a motor vehicle that lights up when the brakes are applied, serving as a warning to following drivers.
  - Direction light: these 4 orange lights in front and rear of the vehicle indicate the intention to slow down in order to change the direction of the vehicle, and remain activated during the maneuver.
  - Reversing Light: these 1 or 2 white light in rear of the vehicle indicate that the vehicle is in reverse. They also light up the recoil area.
  - Fog light: they improve the visibility in the event of fog or snow. There 2 white or yellow lights at the front and 2 red lights at the rear. The rear fog lights are never used in the event of rain as they are too dazzling.

The properties of the lights presented above are summarized in Table 5.7.

5.4.3 Action concept

Entity: action

Definition: action is an entity that represents a maneuver that can be performed by a vehicle.

TABLE 5.7 – Properties of the lights

Type of lights	State	Position	Number	Color
Parking light	on or off	front and rear	4 (2 at the front, 2 at the rear)	white (front) red (rear)
Headlight	high-beams, low-beams or off	front	2	white or yellow
Brake Light	on or off	rear	2 or 3	red
Reversing light	on or off	rear or rear right	1 or 2	white
Direction light	flashing or off	front and rear	4	orange
Fog light	on or off	front and rear	3 or 4 (2 at the front, 1 or 2 at the rear)	white or yellow (front) red (rear)

The *Action* concept is characterized by the following **propriety:**

- Type: this is the type of the action.
  - — Run: this action indicates that the vehicle is taking the same direction and with the same speed as in the previous scene.
  - — Accelerate: this action indicates the increase in the speed of a vehicle.
  - — Decelerate: this action is aimed to reduce the speed of a vehicle.
  - — GoLeftLane: this action indicates a change in the lateral position of a vehicle. The vehicle can move to the left lane.
  - — GoRightLane: this action indicates a change in the lateral position of a vehicle. The vehicle can move to the right lane.
  - — Enter: this action is used to enter the highway
  - — Exit: this action is used to exit the highway,

In the following, we only model the actions and not the lights because we do not need to predict how the intelligent system on the autonomous vehicle reacts to changes of the elements in the driving environment. We focus on the impact of the reactions of intelligent system when the other vehicles approach the autonomous vehicle and the scene changes caused by the actions of autonomous vehicle and the events of the other elements.

5.5 Conclusion

In this chapter, we have introduced three ontologies for the conceptualization and the characterization of the components of the test cases: a highway ontology and a weather ontology to specify the environment in which evolves the autonomous vehicle, and a vehicle ontology which consists of the vehicle devices and the control actions.

Analysing the knowledge in a domain is possible as soon as the specification of the terms of the domain is made. Based on the defined ontologies, we can build instances such as maps, roads and vehicles. In the next chapter (Chapter 6), we develop the ontologies by applying first-order logic to model the relationships between the elements in the scenes.

Chapitre 6

Logical Relations for Ontologies

6.1 Introduction

In the previous chapter, we have presented the concepts of three ontologies for the conceptualization and characterization of the components of the test cases. In order to represent the complex and intricate relationships between the entities of these concepts, we consider three kinds of relationships: the relationships between the highway entities, the relationships between the vehicle entities, and the relationships between the entities of the highway and the vehicle.

Logics are of particular importance in providing a logical formalism for ontologies. Propositional logic, description logic and first-order logic are used to represent the relationships in ontologies [Farkas and Sarbo, 2000] [Jiménez-Ruiz and Grau, 2011] [Peinado et al., 2004] [Horrocks and Sattler, 2001] [Berardi et al., 2004] [Witherell et al., 2010]. The language of first-order logic is completely formal and is more expressive than propositional logic and description logic.

In our context, we need to express the relationships between the different elements in our system. As we will see, we have simple relationships such as the inheritance relationships and the composition relationships between the highway infrastructure entities. And we also have complex relationships such as the position relationships. All these relationships need to be expressed in a simple way. Furthermore, our system requires functions that define the properties of the ontologies concepts. First order logic is thus very suitable.

Therefore, in this chapter, we use first-order logic to represent the different relationships that we define in the context of the ontologies we have built. Moreover, the traffic regulation and the interactions between the entities are written as logics expressions to simulate the environment of the autonomous vehicle. The syntax of the logical model, which includes the logical symbols, the set symbols and the function symbols, is presented in Section 6.2. Then all the

relationships between the entities of the ontologies are presented in Section 6.3. Finally the conclusion of this chapter is given in Section 6.4.

6.2 The Syntax

In this section, we introduce the syntax of first-order logic we use when representing the relationships. We first present the logical symbols, then the non-logical ones, namely the set symbols and the function symbols.

6.2.1 Logical symbols

The logical symbols include two (2) quantifier symbols, seven (7) logical symbols and a set of variables.

- Quantifier symbols
  - Universal: $\forall$, which reads as "for all".
  - Existential: $\exists$, which reads as "it exists".
- Logical connectives
  - conjunction: $\wedge$, which reads as "and".
  - disjunction: $\vee$, which reads as "or".
  - implication: $\rightarrow$, which reads as "if... then...".
  - identity: $=$, which reads as "equal to".
  - conclusion: $|$, which reads as "such that".
  - union: $\cup$, which refers to the union of a collection of sets.
  - concatenation: _ , which refers to a concatenation of a set of elements.
- A set of variables V, denoted by the alphabet elements.

6.2.2 Set symbols

The set symbols represent the sets of entities of the highway ontology, weather ontology and vehicle ontology.

As we have explained in Chapter 5, we consider that the highway is divided into several road parts.

— *RoadPartSet*: it is the set of all possible road parts of the highway including the special zones (toll, tunnel, bridge, road work zone).

— $HighwaySet$: it is a concatenation of elements from set $RoadPartSet$, that is $HighwaySet = x_1\_x_2\_...\_x_n$, where $x_i \in RoadPartSet$, $i = 1, ..., n$, and n is the total number of road parts.

— $RoadwaySet$: it is the set of the roadways of the highway.

— $CarriagewaySet$: it is the set of carriageways of the highway. A highway has a maximum of two (2) carriageways.

— $ThroughLaneSet$: it is the set of through lanes of the highway. A carriageway has a maximum of four (4) through lanes.

— $EntranceLaneSet$: it is the set of entrance lanes of the highway. A road part of highway has a maximum of two (2) entrance lanes.

— $WeaveLaneSet$: it is the set of weave lanes of the highway. A road part of highway has one (1) weave lane, at most.

— $ExitLaneSet$: it is the set of exit lanes of the highway. A road part of highway has a maximum of two (2) exit lanes.

— $TaperSet$: it is the set of tapers of the highway. A road part of highway has a maximum of two (2) tapers.

— $AccelerationSectionSet$: it is the set of acceleration sections of the highway. An entrance lane of highway has one (1) acceleration section, at most.

— $DecelerationSectionSet$: it is the set of deceleration sections of the highway. An exit lane of highway has one (1) deceleration section, at most.

— $AuxiliaryLaneSet = EntranceLaneSet \cup WeaveLaneSet \cup ExitLaneSet$. It is the set of all possible auxiliaries lanes, that is, $\forall x \in AuxiliaryLaneSet$, $x \in EntranceLaneSet$ or $x \in WeaveLaneSet$ or $x \in ExitLaneSet$.

— $LaneSet = ThroughLaneSet \cup AuxiliaryLaneSet$. It is the set of all possible lanes, that is, $\forall x \in LaneSet$, $x \in ThroughLaneSet$ or $x \in AuxiliaryLaneSet$.

— $MedianSet$: it is the set of medians of the highway. A highway has one (1) median, at most.

— $PavedShoulderSet$: it is the set of paved shoulders of the highway. A highway has one (1) paved shoulder, at most.

— $UnpavedShoulderSet$: it is the set of unpaved shoulders of the highway. A highway has one (1) unpaved shoulder, at most.

— $ShoulderSet = PavedShoulderSet \cup UnpavedShoulderSet$. It is the set of all possible shoulders, that is, $\forall x \in ShoulderSet$, $x \in PavedShoulderSet$ or $x \in UnpavedShoulderSet$.

— $TollSet$: it is the set of tolls of the highway. Thus $\forall x \in TollSet$, $x \in RoadPartSet$.

— $TunnelSet$: it is the set of tunnels of the highway. Thus $\forall x \in TunnelSet$, $x \in RoadPartSet$.

— $BridgeSet$: it is the set of bridges of the highway. Thus $\forall x \in BridgeSet$, $x \in RoadPartSet$.

— $RoadWorkSet$: it is the set of road works areas on the highway. Thus $\forall x \in RoadWorkSet$, $x \in RoadPartSet$.

— $SolidLineSet$: it is the set of solid lines on the highway.

— $DashedLineSet$: it is the set of dashed lines on the highway.

— $LongitudinalMarkingSet$: it is the set of longitudinal markings of the highway.

— $ArrowSet$: it is the set of arrows on the highway.

— $ChevronSet$: it is the set of chevrons on the highway.

— $SpecificMarkingSet = ArrowSet \cup ChevronSet$. It is the set of all possible specific markings, that is, $\forall x \in SpecificMarkingSet$, $x \in ArrowSet$ or $x \in ChevronSet$.

— $MarkingSet = LongitudinalMarkingSet \cup SpecificMarkingSet$. It is the set of all possible markings, that is, $\forall x \in MarkingSet$, $x \in LongitudinalMarkingSet$ or $x \in SpecificMarkingSet$.

— $SignSet = SignASet \cup SignBSet \cup SignABSet \cup SignCSet \cup SignDSet \cup SignESet \cup SignJSet \cup SignSRSet \cup SignKSet$. It is the set of all possible signs, that is, $\forall x \in SignSet$, $x \in SignASet$ or $x \in SignBSet$ or $x \in SignABSet$ or $x \in SignCSet$ or $x \in SignDSet$ or $x \in SignESet$ or $x \in SignJSet$ or $x \in SignSRSet$ or $x \in SignKSet$.

— $SymbolSet = MarkingSet \cup SignSet$. It is the set of all possible symbols, that is, $\forall x \in SymbolSet$, $x \in MarkingSet$ or $x \in SignSet$.

— $VehicleSet$: it is the set of vehicles on the highway.

— $ParkingLightSet$: it is the set of parking lights of the vehicle.

— $HeadLightSet$: it is the set of head lights of the vehicle.

— $DirectionLightSet$: it is the set of direction lights of the vehicle.

— $FogLightSet$: it is the set of fog lights of the vehicle.

— $ReversingLightSet$: it is the set of reversing lights of the vehicle.

— $BrakeLightSet$: it is the set of brake lights of the vehicle.

— $LightSet = ParkingLightSet \cup HeadLightSet \cup DirectionLightSet \cup FogLightSet \cup ReversingLightSet \cup BrakeLightSet$. It is the set of all possible lights, that is, $\forall x \in LightSet$, $x \in ParkingLightSet$ or $x \in HeadLightSet$ or $x \in DirectionLightSet$ or $x \in FogLightSet$ or $x \in ReversingLightSet$ or $x \in BrakeLightSet$.

— $ActionSet = \{Run, Accelerate, Decelerate, GoLeftLane, GoRightLane, Enter, Exit\}$. It is the set of all possible actions that can be made by the vehicles.

6.2.3 Function symbols

The functions provide the parameters characterizing the entities which are defined as concepts properties in the ontologies introduced in Chapter 5.

— $Length: L \rightarrow \mathbb{R}+$, where $L = LaneSet \cup TaperSet \cup AccelerationSectionSet \cup DecelerationSectionSet \cup TollSet \cup TunnelSet \cup BridgeSet \cup RoadWorkSet$. $Length(x)$ provides the length of x, $\forall x \in L$.

— $Width: L \rightarrow \mathbb{R}+$. $Width(x)$ provides the width of x, $\forall x \in L$.

— $Height$: $RoadPartSet \rightarrow \mathbb{R}+$. $Height(x)$ provides the height of x, $\forall x \in RoadPartSet$.

— $Material$: $BridgeSet \rightarrow M$, where $M = \{Steel, Brick, Wood\}$. $Material(x)$ provides the material of x, $\forall x \in BridgeSet$.

— $Color$: $VehicleSet \rightarrow C$, where $C = \{White, Silver, Black, Grey, Blue, Red, Brown, Green, Others\}$. $Color(x)$ provides the color of x, $\forall x \in VehicleSet$.

— $Type$: $SignSet \rightarrow T$, where $T = \{A, B, AB, C, D, E, J, SR, K\}$. $Type(x)$ provides the type of x, $\forall x \in SignSet$.

— $Direction$: $VehicleSet \rightarrow D$, where $D = \{Same, Reverse\}$. $Direction(x)$ provides the direction of x, $\forall x \in VehicleSet$.

— $Number$: $CarragewaySet \rightarrow N$, $N \in \{2, 3, 4\}$. $Number(x)$ provides the lane number of x, $\forall x \in CarragewaySet$.

— $Speed$: $VehicleSet \rightarrow \mathbb{R}$. $Speed(x)$ provides the speed of x, $\forall x \in VehicleSet$.

— $SpeedLimit$: $LaneSet \rightarrow \mathbb{R}$. $SpeedLimit(x)$ provides the speed limit of x, $\forall x \in LaneSet$.

— $Position$: $SymbolSet \rightarrow P$, where $P = \{Median, Lane, UnpavedShoulder\}$. $Position(x)$ provides the position of x, $\forall x \in SymbolSet$.

— $Role$: $VehicleSet \rightarrow O$, where $O = \{EgoCar, TargetCar, OtherCar\}$. $Role(x)$ provides the role of x, $\forall x \in VehicleSet$.

— $Category$: $VehicleSet \rightarrow V$, where $V = \{Class1, Class2, Class3, Class4, Class5\}$. $Category(x)$ provides the category of x, $\forall x \in VehicleSet$.

6.3 Relationships

In order to represent the complex and intricate relationships between the entities, we consider three kinds of relationships: the relationships between the highway entities, the relationships between the vehicle entities, and the relationships between the highway and vehicle entities.

6.3.1 Relationships between the highway entities

There are three types of relationships between the entities of the highway ontology:

— Inheritance relationships (unary),
— Composition relationships (binary),
— Position relationships which consist of the longitudinal position (binary), the transverse position (binary or ternary) and the vertical position (binary).

All the defined relationships are summarized in Figure 6-1.

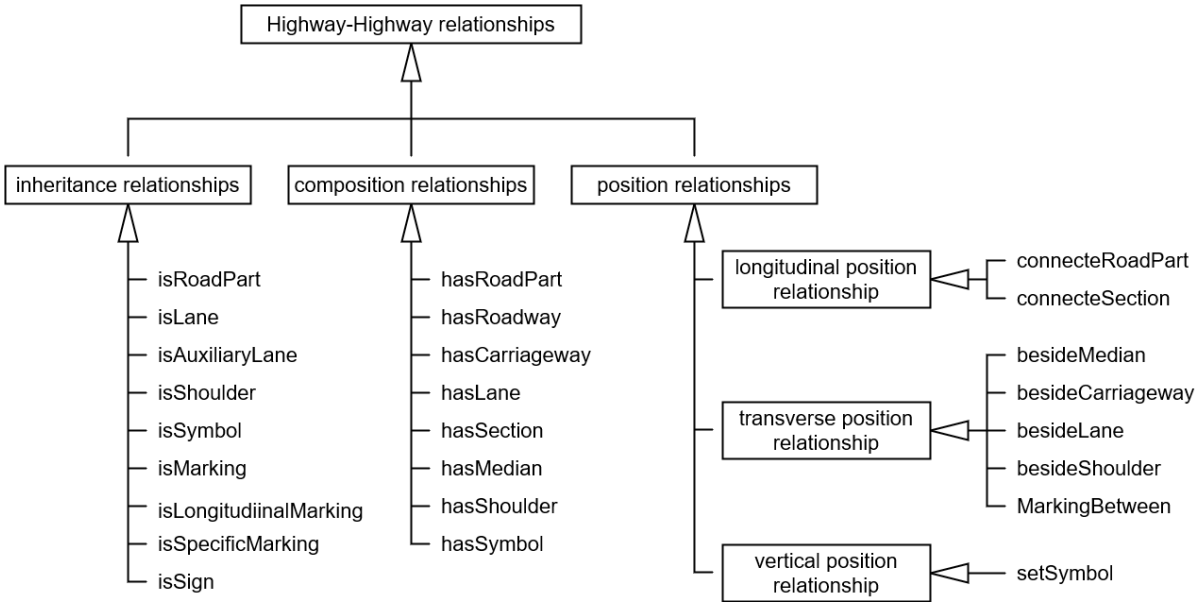

FIGURE 6-1 – Relationships between the highway entities.

6.3.1.1 Inheritance relationships

Inheritance relationship is of "IS-A" type relationship. An inheritance relationship between two entities A and B implies that "A is a B".

Before defining relationships, we define the following inheritance relation symbols:

- *isRoadPart*(x) is a relationship which specifies that an entity belongs to set *RoadPartSet*.
- *isLane* is a relationship which specifies that an entity belongs to set *LaneSet*.
- *isAuxiliaryLane* is a relationship which specifies that an entity belongs to set *AuxiliaryLaneSet*.
- *isShoulder* is a relationship which specifies that an entity belongs to set *ShoulderSet*.
- *isSymbol* is a relationship which specifies that an entity belongs to set *SymbolSet*.
- *isMarking* is a relationship which specifies that an entity belongs to set *MarkingSet*.
- *isLongitudinalMarking* is a relationship which specifies that an entity belongs to set *LongitudinalMarkingSet*.
- *isSpecificMarking* is a relationship which specifies that an entity belongs to set *SpecificMarkingSet*.
- *isSign* is a relationship which specifies that an entity belongs to set *SignSet*.

We can define 2 types of inheritance relationships: simple and combined. For example, the following combined relationship means that any element x in set *EntranceLaneSet* or

WeaveLaneSet or *ExitLaneSet* is of type *AuxiliaryLane*, which implies that x is also a lane.

$$\forall x \in (EntranceLaneSet \vee WeaveLaneSet \vee ExitLaneSet), \quad isAuxiliaryLane(x) \rightarrow isLane(x)$$

We define the following four (4) simple inheritance relationships:

— $\forall x \in (ThroughLaneSet \vee AuxiliaryLaneSet), \quad isLane(x)$;
— $\forall x \in (UnpavedShoulderSet \vee PavedShoulderSet), \quad isShoulder(x)$;
— $\forall x \in (TollSet \vee TunnelSet \vee BridgeSet \vee RoadWorkSet), \quad isRoadPart(x)$;
— $\forall x \in (MarkingSet \vee SignSet), \quad isSymbol(x)$;

Finally, we define the following five (5) combined inheritance relationships:

— $\forall x \in (EntranceLaneSet \vee WeaveLaneSet \vee ExitLaneSet), \quad isAuxiliaryLane(x) \rightarrow isLane(x)$;
— $\forall x \in (LongitudinalMarkingSet \vee SpecificMarkingSet), \quad isMarking(x) \rightarrow isSymbol(x)$;
— $\forall x \in (SolidLineSet \vee DashedLineSet), \quad isLongitudinalMarking(x) \rightarrow isMarking(x) \rightarrow isSymbol(x)$;
— $\forall x \in (ArrowSet \vee ChevronSet), \quad isSpecificMarking(x) \rightarrow isMarking(x) \rightarrow isSymbol(x)$;
— $\forall x \in (SignASet \vee SignBSet \vee SignABSet \vee SignCSet \vee SignDSet \vee SignESet \vee SignJSet \vee SignSRSet \vee SignKSet), \quad isSign(x) \rightarrow isSymbol(x)$;

6.3.1.2 Composition relationships

A composition relationship is a "PART-OF" relationship. In this binary relationship, involved entities are dependent of each other. Let us take an example of sets *Highway* and *RoadPart*. An element of *RoadPart* set is an element of *Highway* and thus both depend on each other. This relationship can be written as: $\forall x \in Highway, \exists y \in RoadPart \mid hasRoadPart(x,y)$, where *hasRoadPart* is the relationship symbol which specifies that entity x is composed of y.

Before defining all the composition relationships, we define first the following composition relationship symbols:

- *hasRoadPart* is a relationship which specifies that an entity is composed of another entity belonging to set *RoadPartSet*.
- *hasRoadway* is a relationship which specifies that an entity is composed of another entity belonging to set *RoadwaySet*.
- *hasCarriageway* is a relationship which specifies that an entity is composed of another entity belonging to set *CarriagewaySet*.
- *hasLane* is a relationship which specifies that an entity is composed of another entity

belonging to set *LaneSet*.

- *hasSection* is a relationship which specifies that an entity is composed of another entity belonging to set $(AccelerationSectionSet \cup DecelerationSectionSet \cup TaperSet)$.
- *hasMedian* is a relationship which specifies that an entity is composed of another entity belonging to set *MedianSet*.
- *hasShoulder* is a relationship which specifies that an entity is composed of another entity belonging to set *ShoulderSet*.
- *hasSymbol* is a relationship which specifies that an entity is composed of another entity belonging to set *SymbolSet*.

It follows that, we define the following composition relationships:

— $\forall x \in HighwaySet, \exists y \in RoadPartSet \mid hasRoadPart(x,y)$;
— $\forall x \in RoadPartSet, \exists y \in RoadwaySet \mid hasRoadway(x,y)$;
— $\forall x \in RoadwaySet, \exists y \in CarriagewaySet \mid hasCarriageway(x,y)$;
— $\forall x \in CarriagewaySet, \exists y \in LaneSet \mid hasLane(x,y)$;
— $\forall x \in RoadwaySet, \exists \in LaneSet \mid hasLane(x,y)$;
— $\forall x \in RoadPartSet, \exists y \in LaneSet \mid hasLane(x,y)$;
— $\forall x \in EntranceLaneSet, \exists y \in AccelerationSectionSet \mid hasSection(x,y)$;
— $\forall x \in EntranceLaneSet, \exists y \in TaperSet \mid hasSection(x,y)$;
— $\forall x \in WeaveLaneSet, \exists y \in AccelerationSectionSet \mid hasSection(x,y)$;
— $\forall x \in WeaveLaneSet, \exists y \in DecelerationSectionSet \mid hasSection(x,y)$;
— $\forall x \in WeaveLaneSet, \exists y \in TaperSet \mid hasSection(x,y)$;
— $\forall x \in ExitLaneSet, \exists y \in DecelerationSectionSet \mid hasSection(x,y)$;
— $\forall x \in RoadwaySet, \exists y \in MedianSet \mid hasMedian(x,y)$;
— $\forall x \in RoadwaySet, \exists y \in ShoulderSet \mid hasShoulder(x,y)$;
— $\forall x \in HighwaySet, \exists y \in SymbolSet \mid hasSymbolt(x,y)$;
— $\exists x \in RoadPartSet, \exists y \in SymbolSet \mid hasSymbol(x,y)$;

6.3.1.3 Position relationships

A position refers to a longitudinal position, a transverse position or a vertical position. A **longitudinal position relationship** indicates the connection order of elements in the driving direction of the vehicle. Before defining the relationships, we define the following longitudinal position relation symbols:

- *connecteRoadPart* is a relationship which specifies that an entity is connected to another entity belonging to set *RoadPartSet*.

- *connecteSection* is a relationship which specifies that an entity is connected to another entity belonging to set $(AccelerationSectionSet \cup DecelerationSectionSet \cup TaperSet)$.

For example, $\forall x, y \in RoadPartSet, connecteRoadPart(x, y)$ means that road part y is connected to road part x, and thus y is the next road part after x.

We define the following three (3) longitudinal position relationships:

— $\forall x \in RoadPartSet, \exists y \in RoadPartSet \mid connecteRoadPart(x, y)$;
— $\forall x \in AccelerationSectionSet, \exists y \in TaperSet \mid connecteSection(x, y)$;
— $\forall x \in DecelerationSectionSet, \exists y \in AccelerationSectionSet \mid connecteSection(x, y)$;

A **transverse position relationship** indicates the connection order of parallel elements. Before defining these relationships, we define the following transverse position relationship symbols:

- *besideMedian* is a relationship which specifies that an entity is adjacent to another entity belonging to set *MedianSet*.
- *besideCarriageway* is a relationship which specifies that an entity is adjacent to another entity belonging to set *CarriagewaySet*.
- *besideLane* is a relationship which specifies that an entity is adjacent to another entity belonging to set *LaneSet*.
- *besideShoulder* is a relationship which specifies that an entity is adjacent to another entity belonging to set *ShoulderSet*.
- *MarkingBetween* is a relationship which specifies that an entity belonging to set *MarkingSet* is between two other entities.

For example, relationship $\exists x \in LaneSet, \forall y \in MedianSet \mid besideMedian(x, y)$ means that for all median y, there is a lane x adjacent to y.

Another example is: $\exists x \in MarkingSet, \forall y, z \in (MedianSet \cup LaneSet \cup ShoulderSet) \wedge (besideMedian(y, z) \vee besideShoulder(y, z) \vee besideLane(y, z)) \mid MarkingBetween(x, y, z)$, which means that, for all y and z belonging to set *MedianSet* or *LaneSet* or *ShoulderSet*, there is a marking x between y and z.

We define the following five (5) transverse position relationships:

— $\exists x \in LaneSet, \forall y \in MedianSet \mid besideMedian(x, y)$;
— $\exists x \in (ThroughLaneSet \cup AuxiliaryLaneSet \cup PavedShoulderSet), \forall y \in ThroughLaneSet \mid besideLane(x, y)$;
— $\exists x \in (AuxiliaryLaneSet \cup PavedShoulderSet), \forall y \in AuxiliaryLaneSet \mid besideLane(x, y)$;
— $\exists x \in (MedianSet \cup PavedShoulderSet), \forall y \in CarriagewaySet \mid besideCarriageway(x, y)$;

— $\exists x \in MarkingSet, \forall y,z \in (MedianSet \cup LaneSet \cup ShoulderSet) \wedge (besideMedian(y,z) \vee besideShoulder(y,z) \vee besideLane(y,z)) \mid MarkingBetween(x,y,z)$.

Finaly a **vertical position relationship** indicates the vertical connection to the ground. Before defining the vertical position relationships, we define one (1) relation symbol:

- *setSymbol* is a relationship which specifies that an entity belonging to set *Equipment* is located on another entity.

For example, $\forall x \in SignCSet, \exists y \in CarriagewaySet \mid setSymbol(x,y)$ means that for each indication sign x in set *SignCSet*, there is a carriageway y on which x is located.

Thus we define the following three (3) vertical position relationships:

— $\exists y \in (UnpavedShoulderSet \cup MedienSet), \forall x \in SignSet \mid setSymbol(x,y)$;
— $\exists y \in CarriagewaySet, \forall x \in SignCSet \mid setSymbol(x,y)$;
— $\exists y \in ArrowSet, \exists x \in LaneSet \mid setSymbol(x,y)$;

6.3.2 Relationships between the vehicle entities

As explained before, a vehicle can have one of the three following roles: *EgoCar*, *TargetCar* and *OtherCar*. *EgoCar* represents the autonomous vehicle (AV) itself, *TargetCar* represents the vehicle targeted by the AV and *OtherCar* represents one of the other vehicles in a scene. There are eight (8) binary relationships between *EgoCar* and the other cars (*TargetCar* and *OtherCar*). We consider that *EgoCar* position is the origin point as shown in Figure 6-2.

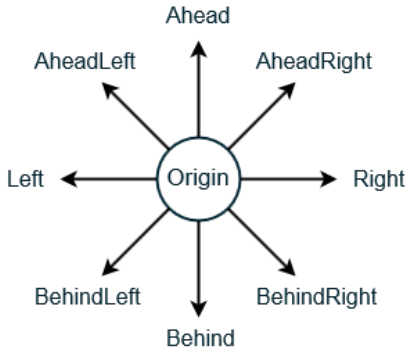

FIGURE 6-2 – Vehicles around *EgoCar*.

Before defining the relationships between the vehicle entities, we define the following relationship symbols:

- *hasAheadVehicle* is a relationship which specifies that an entity belonging to set *VehicleSet* is located ahead of $EgoCar$.
- *hasAheadLeftVehicle* is a relationship which specifies that an entity belonging to set *VehicleSet* is located at the left front of $EgoCar$.
- *hasLeftVehicle* is a relationship which specifies that an entity belonging to set *VehicleSet* is located at the left of $EgoCar$.
- *hasBehindLeftVehicle* is a relationship which specifies that an entity belonging to set *VehicleSet* is located at the left rear of $EgoCar$.
- *hasBehindVehicle* is a relationship which specifies that an entity belonging to set *VehicleSet* is located behind of $EgoCar$.
- *hasBehindRightVehicle* is a relationship which specifies that an entity belonging to set *VehicleSet* is located at the right rear of $EgoCar$.
- *hasRightVehicle* is a relationship which specifies that an entity belonging to set *VehicleSet* is located at the right of $EgoCar$.
- *hasAheadRightVehicle* is a relationship which specifies that an entity belonging to set *VehicleSet* is located at the right front of $EgoCar$.

$EgoCar$ can have a *TargetCar* in front, which is conceptualised using the following relationship:

$$\exists x, y \in VehicleSet \mid (Role(x) = `EgoCar') \wedge (Role(y) = `TargetCar') \wedge hasAheadVehicle(x,y).$$

And each other car around $EgoCar$ is considered using the following seven (7) relationships:

— $\exists x, y \in VehicleSet \mid (Role(x) = `EgoCar') \wedge (Role(y) = `OtherCar') \wedge hasAheadLeftVehicle(x,y)$;
— $\exists x, y \in VehicleSet \mid (Role(x) = `EgoCar') \wedge (Role(y) = `OtherCar') \wedge hasAheadRightVehicle(x,y)$;
— $\exists x, y \in VehicleSet \mid (Role(x) = `EgoCar') \wedge (Role(y) = `OtherCar') \wedge hasLeftVehicle(x,y)$;
— $\exists x, y \in VehicleSet \mid (Role(x) = `EgoCar') \wedge (Role(y) = `OtherCar') \wedge hasRightVehicle(x,y$;
— $\exists x, y \in VehicleSet \mid (Role(x) = `EgoCar') \wedge (Role(y) = `OtherCar') \wedge hasBehindLeftVehicle(x,y)$;
— $\exists x, y \in VehicleSet \mid (Role(x) = `EgoCar') \wedge (Role(y) = `OtherCar') \wedge hasBehindRightVehicle(x,y)$;
— $\exists x, y \in VehicleSet \mid (Role(x) = `EgoCar') \wedge (Role(y) = `OtherCar') \wedge hasBehindVehicle(x,y)$;

6.3.3 Relationships between the entities of the highway and the vehicle

In order to define the relationships between the highway and vehicle entities, we consider that all vehicles obey the traffic rules. Therefore, we define the following relationship symbols:

- *enters* is a relationship which specifies that an entity belonging to *VehicleSet* is entering an entity belonging to set $W = (RoadwaySet \cup TollSet \cup TunnelSet \cup BridgeSet \cup RoadWorkSet)$.
- *leaves* is a relationship which specifies that an entity belonging to set *Vehicle* is leaving an entity belonging to set W.
- *on* is a relationship which specifies that an entity belonging to set *Vehicle* is located on an entity belonging to set W.

Thus, we define the following four (4) binary relationships between the vehicle and highway entities:

— $\exists x \in VehicleSet, \exists y \in (LaneSet \cup PavedShoulderSet) \mid on(x,y)$

— $\exists x \in VehicleSet, \exists y \in RoadPartSet \mid in(x,y)$

— $\exists x \in VehicleSet, \exists y \in (LaneSet \cup PavedShoulderSet \cup RoadPartSet) \mid enters(x,y)$

— $\exists x \in VehicleSet, \exists y \in (LaneSet \cup PavedShoulderSet \cup RoadPartSet) \mid leaves(x,y)$

6.4 Conclusion

In this chapter, we have defined three kinds of relationships between the entities of the ontologies built in Chapter 5. We have also showed how to express these relationships using the first order logic.

In the next chapter, we present our test cases formalization model based on the ontological concepts defined in Chapter 5 and the logical relationships defined in Chapter 6.

Chapitre 7

Formal modelling using PEPA

7.1 Introduction

We have built three ontologies for the conceptualization and the characterization of the elements of the test cases in Chapter 5. Relationships and rules, such as traffic regulation, are expressed using a first-order logic in Chapter 6. The scenes of the test cases can be defined using the concepts and the relationships we have defined. But these scenes are static like snapshots lacking of dynamic behaviours. Therefore, they are not enough for the simulation and the validation of the functions of ADSs. This emphasizes the necessity of a model which allows us to capture the dynamicity of the scenarios.

In this chapter, we focus on the modelling of the dynamic transitions between the driving scenes to generate scenarios using a formal modelling technique that can be used to describe and analyze concurrent systems like ADSs. We are interested in the Performance Evaluation Process Algebra (PEPA) [Hillston, 1994] modelling technique to generate the scenarios according to the state-change elements of the system. PEPA can model system elements which behave and evolve individually or in cooperation with each others.

Indeed, we need a formalism that allows us to model concurrent agents, like the vehicles. PEPA allows modelling such a concurrent system which can behave in an individual or competitive way. This formalism is powerful enough to model all the behaviors of the components in our system. In our knowledge, until now no work has been devoted to the formal modelling of test cases (see Chapter 3).

In this chapter, the syntax of the PEPA language is presented in Section 7.2. We introduce our general PEPA model for the highway in Section 7.3. In Section 7.4, we use the running example introduced in Chapter 4 to describe more specifically the construction of the corresponding PEPA model, and the generation of the scenarios. Moreover, we show how to identify

the critical ones. Finally, Section 7.6 concludes this chapter.

7.2 Syntax of PEPA

Performance Evaluation Process Algebra (PEPA) is a stochastic process algebra designed for modelling computer and communication systems [Hillston, 1994]. This language has been developed to investigate how the compositional features of a process algebra might impact upon the practice of performance modelling.

A PEPA model is constructed by identifying the components performing activities. The main attribute which was missing from a process algebra is the quantification of time and uncertainty, which is necessary for performance evaluation. This is achieved in PEPA by associating an exponentially distributed random variable with each activity, representing its duration. The duration of the activity may be represented by a single real number parameter, which can be any positive real number or the distinguished symbol $\top$, which refers to the unspecified cooperation activity rate. If more than one activity can be simultaneously enabled by a component, each unspecified activity rate must be assigned a weight to determine the relative probabilities of possible outcomes of the activity [Hillston, 1994].

The stochastic process algebra PEPA is a simple language with a small set of operators. The operators and their syntax are defined as follows:

$$S \stackrel{\text{def}}{=} (\alpha, r).P \mid P + Q \mid P \underset{L}{\bowtie} Q \mid P/L \mid A$$

Prefix:	$S \stackrel{\text{def}}{=} (\alpha, r).P,$	component S carries out activity (α, r) which has action type α and a duration which is exponentially distributed with parameter r before behaving as P.
Choice:	$S \stackrel{\text{def}}{=} P + Q,$	S may behave either as component P or as component Q.
Cooperation:	$S \stackrel{\text{def}}{=} P \underset{L}{\bowtie} Q,$	S is the result of the cooperation or synchronisation between components P and Q. Shared activities in the cooperation set L determine the interactions between components P and Q, replacing the individual activities of the individual components P and Q with a rate reflecting the rate of the slower participant.
Hiding:	$S \stackrel{\text{def}}{=} P/L,$	the system behaves as component P except that any activity of a type within the set L is hidden. Its type is not witnessed upon completion. It appears as the unknown type τ and can be regarded as an internal delay by the component.
Constant:	$S \stackrel{\text{def}}{=} A$	it assigns S the behaviour of component A. In general, it assigns names to components.

PEPA abstracts activities performed by components into a continuous-time Markov pro-

cess. The generation of this underlying Markov process is based on the derivation graph of the model. From a model definition M we can apply the semantic rules exhaustively to find the complete set of reachable states, the derivative set of M, $ds(M)$. The derivation graph is a directed multi-graph whose nodes are the reachable states of the model $ds(M)$ and whose arcs represent the possible transitions between them. In order to derivea Markov process from a PEPA model, we associate a state with each node of the derivation gragh. The edges are labelled only by the activity rates since the action type information is discarded. While multiple edges between a pair of nodes are combined by summing the corresponding tates. The rate on an edge in this modified gragh becomes the corresponding entry in the infinitesimal generator matrix [Kloul, 2006].

Like all state-based modelling techniques, there is a risk of combinatorial explosion of the state space of the underlying Markov chain. PEPA has an aggregation technique that may allow reducing the size of the model [Hillston, 1994]. It is also supported by a resolution method based on the differential equations [Gilmore et al., 2003].

7.3 General PEPA model for highway

In this section, we introduce our general PEPA model for the highway. We consider the portion of a highway carriageway around the autonomous vehicle, Ego (bleu vehicle in Figure 7-1). Depending on the speed of Ego and the speed of the possible vehicle following and just before Ego, we can define a ***critical zone*** (yellow zone in Figure 7-1) in the center lane. This zone is delimited considering the minimum safe distance that must separate Ego from the other vehicles: the one just before and the one just after.

The minimum safe distances are the following distances of two seconds, which are calculated from the speed of the vehicles. The formulas for front minimum safe distances is: $AMSD = 2v - e$, where v_e is the speed of the autonomous vehicle. And the formulas for behind minimum safe distances are: $BMSD = 2v - t$, where v_t is the speed of the vehicle behind the autonomous vehicle. Here, we do not consider the influence of weather, pavement materials, road slope and vehicle performance, but we can improve the division of critical zone as needed. The formulas for minimum safe distances can be changed to: $FMSD = 2v - e + \alpha$ and $FMSD = 2v - t + \alpha$, where α is the distance added caused by the influence of the other factors. Vehicles, including autonomous vehicle, determines their speeds according to different conditions. In the PEPA model, they can achieve acceleration or deceleration.

Moreover, we separate the portion of carriageway into six (6) zones as shown in Figure 7-2. We number these zones from one to six. Zone 1 indicates the left lane. Zone 2 and Zone 5 indicate the uncritical zones in front and behind Ego. Zone 3 indicates the critical zone in front of Ego while Zone 4 indicates the critical one behind it. Zone 6 indicates the right lane. Both Zone 1 and Zone 6 are uncritical zones for Ego.

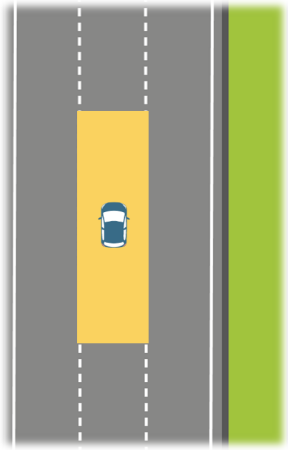

FIGURE 7-1 – Critical zone around Ego.

Another vehicle can be in any area around Ego. We model the movements of the other vehicles between the zones with the graph in Figure 7-3. This graph shows all possibles transitions between these zones. For example, a vehicle can move between Zone 1 and Zone k, and between Zone k and Zone 6, $k = 2,3,4,5$. It can also move between Zone 2 and Zone 3, and between Zone 4 and Zone 5. All vehicles in the uncritical zones can enter the critical zones and vice versa.

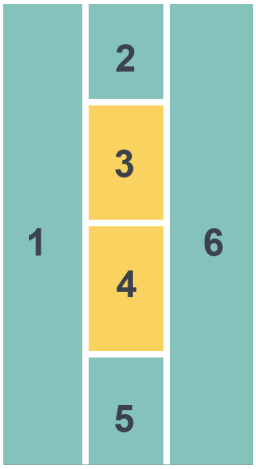

FIGURE 7-2 – Zones in the scene.

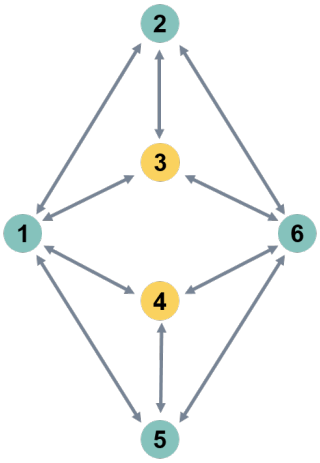

FIGURE 7-3 – Movements between zones.

Considering the concepts we defined in our ontologies, we build a PEPA model which consists of nineteen (19) components. Table 7.1 shows the PEPA component/action in the model corresponding to each concept in the ontologies. Moreover, the PEPA components are listed in Table 7.2 with, for each, its possible actions.

There are actions which have no corresponding concept in Table 7.1. They are used to indicate the end of an action whose concept is given in this table. For example, action *enterV* has the corresponding concept *Enter*, while action *noEnterV* does not.

TABLE 7.1 – Ontology concepts and their corresponding components or actions

Ontology Concept	**Components**
Vehicle	*VehicleEGO*, *VehicleA*
EntranceLane	$EnLOff$
ExitLane	$ExLOff$
Toll	$TollOff$
RoadWork	$RoadWorkOff$
Tunnel	$TunnelOff$
Bridge	$BridgeOff$
Sign	$SignAOff$, $SignBOff$, $SignABOff$, $SignCOff$, $SignDOff$, $SignEOff$, $SignJOff$, $SignSROff$, $SignKOff$, *SpeedLimit*
Weather	*Visibility*
	Actions
Run	*runVehicleEGO*, *runVehicleA*
Accelerate	*accelerateVehicleEGO*, *accelerateVehicleA*
Decelerate	*decelerateVehicleEGO*, *decelerateVehicleA*
GoLeftLane	*goLeftLaneVehicleA*
GoRightLane	*goRightLaneVehicleA*
Enter	*enterV*
Exit	*exitV*

TABLE 7.2 – PEPA Components and their possible actions

Components	**Actions**
VehicleEGO	*runVehicleEGO*, *accelerateVehicleEGO*, *decelerateVehicleEGO*
VehicleA	*runVehicleA*, *accelerateVehicleA*, *decelerateVehicleA*, *goLeftLaneVehicleA*, *goRightLaneVehicleA*
$EnLOff$	*entranceOn*, *enterV*, *noEnterV*
$ExLOff$	*exitOn*, *exitV*, *noExitV*
$TollOff$	$tollOn$, $tollOut$, $tollOff$
$RoadWorkOff$	*workOn*, *workOut*, *workOff*
$TunnelOff$	*tunnelOn*, *tunnelOut*, $tunnelOff$
$BridgeOff$	*bridgeOn*, *bridgeOut*, $bridgeOff$
$SignAOff$	*signAon*, $signAoff$
$SignBOff$	*signBon*, $signBoff$
$SignABOff$	*signABon*, $signABoff$
$SignCOff$	*signCon*, $signCoff$
$SignDOff$	*signDon*, $signDoff$
$SignEOff$	*signEon*, $signEoff$
$SignJOff$	*signJon*, $signJoff$
$SignSROff$	*signSRon*, $signSRoff$
$SignKOff$	*signKon*, $signKoff$
SpeedLimit	*limitLower*, *limitGreater*
Visibility	*visibilityLower*, *visibilityGreater*

In the following, we introduce the PEPA components with the equations characterizing their behaviours. We distinguish the actions of Ego from those of the other vehicles.

7.3.1 Component *VehicleEgo*

Component *VehicleEgo* models the behaviour of vehicle Ego. As our objective is to generate all the scenarios describing the situations to which Ego could be confronted, we do not model its reactions when these situations occur. Therefore, we consider that Ego is always in the center lane with no lane change actions. The PEPA equation of the sequential component *VehicleEgo* is the following:

$$\begin{aligned} VehicleEGO = {} & (runVehicleEGO, e_1).VehicleEGO + (accelerateVehicleEGO, e_2).VehicleEGO \\ & + (decelerateVehicleEGO, e_3).VehicleEGO; \end{aligned}$$

Action *runVehicleEGO* models Ego rolling on the lane without changing its direction or its speed. The actions *accelerateVehicleEGO* and *decelerateVehicleEGO*, respectively, indicate that Ego accelerates and decelerates. Variables e_1, e_2 and e_3 are the rates of the corresponding actions. The PEPA component *VehiculeEGO* always returns to its initial and unique component state after completing these actions.

7.3.2 Component *VehicleA*

Now consider that a car *VA* is rolling on a portion of carriageway. This car represents the other traffic around Ego, as it can be in any zone. We model the behaviour of this car using component *VehicleA*. The PEPA equation of the sequential component *VehicleA* is the following:

$$\begin{aligned} VehicleA = {} & (runVehicleA, a1).VehicleA + (accelerateVehicleA, a2).VehicleA \\ & + (decelerateVehicleA, a3).VehicleA + (goLeftLaneVehicleA, a4).VehicleA \\ & + (goRightLaneVehicleA, a5).VehicleA; \end{aligned}$$

The action *runVehicleA* models *VA* rolling on the lane without changing its direction or its speed. The actions *accelerateVehicleA* and *decelerateVehicleA*, respectively, model the fact that *VA* accelerates and decelerates. *goLeftLaneVehicleA* and *goRightLaneVehicleA* model *VA* changing lane to the left lane and the right lane, respectively. Variables a_i, $i = i, ..., 5$, are the rates of the corresponding actions. Component *VehicleA* always returns to its initial and unique state after completing these actions.

We can have more vehicles for the construction of a scene. Thus more PEPA components can be considered in the model, such as *VehicleB*, *VehicleC* ..., which behave exactly like *VehicleA*. These components represent all the other traffics in the different zones.

7.3.3 Components $EnLOff$ and $ExLOff$

We also model the entrance lanes and the exit lanes of the highway. They are modelled using PEPA components $EnLOff$ and $ExLOff$, respectively. The initial state of these components corresponds to the case where we are in a highway part *RoadPart* with no entrance lane (for $EnLOff$) and no exit lane (for $ExLOff$).

The PEPA equations of the sequential component $EnLOff$ are the followings:

$$EnLOff = (entranceOn, u_6).EnLOn;$$
$$EnLOn = (enterV, u_7).EnLOff + (noEnterV, u_8).EnLOff;$$

Action *entranceOn* models the case where there is an entrance lane appearing and the change of the component state to *EnLOn* means that the highway is ready to welcome a new vehicle from the entrance lane. Action *enterV* models the presence of a vehicle on the entrance lane, while action *noEnterV* models the case where there is no vehicle on this lane. Both actions lead to state $EnLOff$ which models the fact the entrance lane is passed. Variables u_6, u_7 and u_8 are the rates of the corresponding actions.

Similarly the PEPA equations of component $ExLOff$ are the following:

$$ExLOff = (exitOn, u_9).ExLOn;$$
$$ExLOn = (exitV, u_{10}).ExLOff + (noExitV, u_{11}).ExLOff;$$

Action *exitOn* models the case where there is an exit lane appearing which leads to state change of the component to *ExLOn*. Action *exitV* models a vehicle exiting the highway using the exit lane, while action *noExitV* models the case where no vehicle exits the highway. Both actions lead to state $ExLOff$. Variables u_9, u_{10} and u_{11} are the rates of the corresponding actions.

7.3.4 Components modelling the special areas

We also consider the sub-entities belonging to the concept *RoadPart* defined in Chapter 5. These sub-entities are the special areas on the highway: *Toll*, *Bridge*, *Tunnel* and *RoadWork*. In the following, we present the components modelling these concepts.

7.3.5 Component $TollOff$

Firstly, we focus on the concept *Toll* whose entity represents a segment of the highway for payment, and present the PEPA component modelling it.

The PEPA equations of the sequential component $TollOff$ are the following:

$$TollOff = (tollOn, u_1).TollOn;$$
$$TollOn = (tollOut, u_2).TollOut;$$
$$TollOut = (tollOff, u_3).TollOff;$$

The initial state $TollOff$ indicates that there is no toll on the current *RoadPart* of the highway. Action $tollOn$ models the appearance of a toll and it leads to state $TollOn$. The action $tollOut$ models the fact that Ego passed toll and the completion of this action leads to state $TollOut$. The component returns to state $TollOff$ with action $tollOff$ which models the fact that the toll does no longer appear in the current road portion. Variables u_1, u_2 and u_3 are the rates of the corresponding actions.

7.3.6 Component $BridgeOff$

The bridge in this work refers to those segments with viaducts above the highway, not the segments of the highway that serves as bridges. It is also the same as the toll and thus can be modelled following three stages: appearance, entrance and exit.

The PEPA equations of the sequential component $BridgeOff$ are the following:

$$BridgeOff = (bridgeOn, u_{21}).BridgeOn;$$
$$BridgeOn = (bridgeOut, u_{22}).BridgeOut;$$
$$BridgeOut = (bridgeOff, u_{23}).BridgeOff;$$

The initial state $BridgeOff$ indicates that there is no bridge on the current *RoadPart* of the highway. Action *bridgeOn* models the appearance of a bridge and it leads to state *BridgeOn*. The action *bridgeOut* models the fact that Ego passed bridge and the completion of this action leads to state *BridgeOut*. The component returns to state $BridgeOff$ with action $bridgeOff$ which models the fact that the bridge does no longer appear in the current road portion. Variables u_{21}, u_{22} and u_{23} are the rates of the corresponding actions.

7.3.7 Component $TunnelOff$

Similar to a bridge, a tunnel is also an infrastructure over a highway segment, but it is generally longer than the width of the bridge, so vehicles need more time to pass through this road part. Their models are similar, and we can set different activities' rates values to achieve their differences.

The PEPA equations of the sequential component $TunnelOff$ are the following:

$$TunnelOff = (tunnelOn, u_{31}).TunnelOn;$$
$$TunnelOn = (tunnelOut, u_{32}).TunnelOut;$$
$$TunnelOut = (tunnelOff, u_{33}).TunnelOff;$$

The initial state $TunnelOff$ indicates that there is no tunnel on the current *RoadPart* of the highway. Action *tunnelOn* models the appearance of a tunnel and it leads to state *TunnelOn*. The action *tunnelOut* models the fact that *Ego* passed tunnel and the completion of this action leads to state *TunnelOut*. The component returns to state $TunnelOff$ with action *tunnelOff* which models the fact that the tunnel does no longer appear in the current road portion. Variables u_{31}, u_{32} and u_{33} are the rates of the corresponding actions.

7.3.8 Component *RoadWorkOff*

Road works may make one or several lanes in a certain road part impossible. At this time, the vehicles need to change lanes to avoid driving in these lanes. However, our model assumes that the autonomous vehicle does not change its lane, so the road repairs included in this work refer to those road work that do not affect the passage of the road and only need to reduce the speed of the road.

The PEPA equations of the sequential component $RoadWorkOff$ are the following:

$$RoadWorkOff = (roadworkOn, u_{41}).RoadWorkOn;$$
$$RoadWorkOn = (roadworkOut, u_{42}).RoadWorkOut;$$
$$RoadWorkOut = (roadworkOff, u_{43}).RoadWorkOff;$$

The initial state $RoadWorkOff$ indicates that there is no road work on the current *RoadPart* of the highway. Action *roadworkOn* models the appearance of a tunnel and it leads to state *RoadWorkOn*. The action *roadworkOut* models the fact that *Ego* passed tunnel and the completion of this action leads to state *RoadWorkOut*. The component returns to state $RoadWorkOff$ with action $roadworkOff$ which models the fact that the road work does no longer appear in the current road portion. Variables u_{41}, u_{42} and u_{43} are the rates of the corresponding actions.

7.3.9 Components modelling the traffic signs

We consider two types of traffic signs on the highway: short and long term signs. A short-term sign represents the sign which is valid for the moment where the sign is present. A long-term sign represents a sign which is valid for a long duration after the sign appearance, until another sign that replaces it appears.

In the highway ontology, we have define a set of signs, which are classified as *SignA*, *SignB*, *SignA*, *SignC*, *SignD*, *SignE*, *SignJ*, *SignSR* and *SignK*, according to the definition provided in the official French document for road symbols [Ministère de l'écologie, 1988]. There are short-term and long-term signs among them. We will not expand here because there are too many signs.

In the following, we first present the components modelling short-term signs, then long-term signs.

7.3.9.1 Component $SignMOff$

As all short-term signs are modelled similarly, we only present the PEPA equations of the sequential component modelling *SignM*, where $M = \{A, B, AB, C, D, E, J, SR, K\}$:

$$SignMOff = (signMon, u_{18}).SignMOn;$$
$$SignMOn = (signMoff, u_{19}).SignMOff;$$

The initial state is noted $SignMOff$ which models the absence of *SignM*. The state changes to *SignMOn* with action *signMon*, which models *SignM*'s appearance. The component returns to state $SignMOff$ by action $signMoff$, which models the end of the effect of *SignM*. Variables u_{18} and u_{19} are the rates of the corresponding actions.

7.3.9.2 Component $SpeedLimit$

We consider the speed limit signs as the long-term signs in our PEPA model. These are modelled using the PEPA sequential component *SpeedLimit* as follows:

$$SpeedLimit = (limitLower, u_{12}).SpeedLimit + (limitGreater, u_{13}).SpeedLimit;$$

The initial state *SpeedLimit* indicates any speed limit. The action *limitLower* models the appearance of a speed limit sign and the information that the number on the sign is lower than the initial speed limit. The action *limitGreater* models a speed limit sign appearance and the information that the number on the sign is greater than the initial speed limit. Variables u_{12} and u_{13} are the rates of the corresponding actions. Component *SpeedLimit* always returns to the same state (initial state) once these actions are completed.

7.3.10 Component $Visibility$

We have defined twelve (12) concepts in the weather ontology to describe the weather phenomena. As we have explained in Chapter 5, in this work, we use the property of visibility to model the impact of the weather on the autonomous vehicle.

The PEPA equation of the sequential component *Visibility* modelling the corresponding concept is the following:

$$Visibility = (visibilityLower, u_4).Visibility + (visibilityGreater, u_5).Visibility;$$

The initial state *Visibility* indicates any visibility of the environment we set. The action *visibilityLower* models a decrease in the visibility. The action *visibilityGreater* models an increase in the visibility. Variables u_4 and u_5 are the rates of the corresponding actions. Component *Visibility* always returns to the same state (initial state) once these actions are completed.

7.3.11 Component *Situation*

We consider a component noted *Situation* which models a situation that Ego can meet. This component has several states, and its equations depend on the number of vehicles and the types of infrastructure elements we have in the initial scene. Each derivative state $Situation(XN)^*S?$ of component *Situation* models a possible scene in *RoadPart*. In the notation of the state, X refers to a zone number, $X = 1, ..., 6$, N is the vehicle name, $N = A, B, ...$, and S is an action consequence. The number of occurrences (zero or more) of XN is equal to the number of other vehicles in each component state. The question mark in "S?" indicates zero or one occurrence of S. For example, state $Situation2AEnterV$ models the case where VA is on *Zone* 2 and there is another vehicle on the entrance lane.

The component *Situation* is passive with respect to the actions of the other model components. The activity rates of these actions are represented by the distinguished symbol $\top$. If the frequency of the occurrence of an activity enabled by component *Situation* needs to be adapted, the activity rate $\top$ should be assigned a weight p_i, $i \in \mathbb{N}$.

If more than one (say y) activity of the same action can be simultaneously enabled by component *Situation*, each unspecified activity rate $\top$ must be assigned a weight w_i, $i = n, ..., m$, $m - n = y$. Thus the probability q_i of possible outcomes of each activity can be calculated with the following equation:

$$q_i = \frac{w_i}{\sum_{i=n}^{m}}, n \leq i \leq m$$

7.3.12 General Equation

Finally, the equation of the complete PEPA model is the following:

$$Scenario \stackrel{\text{def}}{=} Situation(XN)^*S? \underset{L}{\bowtie} (VehicleEGO \| (VehicleA)? \| ... \| (VehicleZ)? \| EnLOff? \| ExLOff? \| TollOff? \| SignMOff? \| SppedLimit? \| Visibility?)$$

where L is the actions set on which all the components must cooperate individually with component $Situation(XN)^*S?$. This model equation states that there must be at least vehicle Ego (component $VehicleEGO$) while all the other components are optional.

7.4 Example: A two vehicles PEPA model

We construct a PEPA model for the running example presented in Chapter 4. There are three vehicles in the example: Ego-car Ego (blue car in Figure 8-2), Target car VA (red car in Figure 8-2) and an Other car VB (green car in Figure 8-2). Since the division of zones is based on the position of Ego which is in Lane 1, the zones of the running example are noted as Figure 7-4. Based on the entities in the initial scene of this example, we consider six PEPA components: $VehicleEGO$, $VehicleA$, $VehicleB$, $EnLOff$, $SignABOff$ and $Situation2A$.

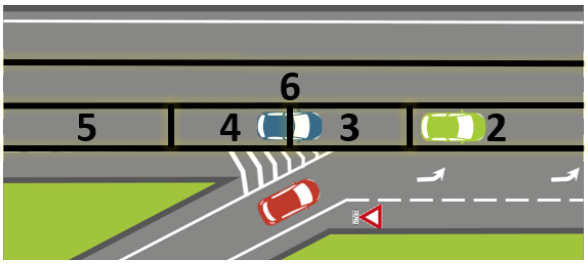

FIGURE 7-4 – Zones of running example.

We use component $VehicleEGO$ to model the autonomous vehicle Ego with no lane change actions. The PEPA equation of the sequential component $VehicleEgo$ is the one defined in Section 7.3, that is:

$$\begin{aligned} VehicleEGO = {} & (runVehicleEGO, e_1).VehicleEGO + (accelerateVehicleEGO, e_2).VehicleEGO \\ & + (decelerateVehicleEGO, e_3).VehicleEGO; \end{aligned}$$

Variables e_i, $i = 1,2,3$ are the rates of the corresponding actions.

Component $VehicleA$ models the green car VA which is the target car of Ego. Generally, the PEPA equation of the sequential component $VehicleA$ is the following:

$$\begin{aligned} VehicleA = {} & (runVehicleA, a1).VehicleA + (accelerateVehicleA, a2).VehicleA \\ & + (decelerateVehicleA, a3).VehicleA + (goLeftLaneVehicleA, a4).VehicleA \\ & + (goRightLaneVehicleA, a5).VehicleA; \end{aligned}$$

In the running example, we suppose that VA rolls with neither lane change actions or speed change actions. Thus the PEPA equation of the sequential component $VehicleA$ is reduced to the following:

$$VehicleA = (runVehicleA, a1).VehicleA;$$

Variable a_1 is the rate of action *runVehicleA*.

Component *VehicleB* models the red car *VB* which arrives on the entrance lane and wants to insert into the highway. Generally, the PEPA equation of the sequential component *VehicleB* is the following:

$$\begin{aligned} VehicleB = &(runVehicleB, b1).VehicleB + (accelerateVehicleB, b2).VehicleB \\ &+ (decelerateVehicleB, b3).VehicleB + (goLeftLaneVehicleB, b4).VehicleB \\ &+ (goRightLaneVehicleB, b5).VehicleB; \end{aligned}$$

In the running example, it can do all vehicle manoeuvring actions except change to right lane. Thus the PEPA equation of the sequential component *VehicleB* is the following:

$$\begin{aligned} VehicleB = &(runVehicleB, b1).VehicleB + (accelerateVehicleB, b2).VehicleB \\ &+ (decelerateVehicleB, b3).VehicleB + (goLeftLaneVehicleB, b4).VehicleB; \end{aligned}$$

Variables b_i, $i = 1,2,3,4$ are the rates of the corresponding actions.

Now, we need to model the changing infrastructure of the running example, that are the entrance lane and the *give way* sign. We use the PEPA component $EnLOff$ we have defined in Section 7.3 to model the entrance lane:

$$\begin{aligned} EnLOff &= (entranceOn, u1).EnLOn; \\ EnLOn &= (enterV, u2).EnLOff + (noEnterV, u3).EnLOff; \end{aligned}$$

The short-term sign *give way* is one of the priority signs which is classified as a *AB* type in the highway ontology. To model this sign, we use component $SignABOff$ which is similar to $SignAOff$, the component introduced in the last section:

$$\begin{aligned} SignABOff &= (signABon, u4).SignABOn; \\ SignABOn &= (signABoff, u5).SignABOff; \end{aligned}$$

Variables u_i, $i = 1,2,3,4,5$ are the rates of the corresponding actions.

We model the possible scenarios occurring in the running example using $Situation2A$ which initial state represents the initial scene where *VehicleA* is in Zone 2.

$$Situation \stackrel{\text{def}}{=} Situation2A;$$

This initial state is given by the following equation:

$$\begin{aligned} Situation2A = &(runVehicleEGO, \top).Situation2A + (accelerateVehicleEGO, \top).Situation2A \\ &+ (decelerateVehicleEGO, \top).Situation2A + (runVehicleA, \top).Situation2A \\ &+ (entranceOn, \top).Situation2AEnLOn; \end{aligned}$$

In the initial state $Situation2A$, *Ego* has the choice between the actions *runVehicleEGO*, $accelerateVehicleEGO$ and $decelerateVehicleEGO$. Vehicle *VA* always performs action *run-*

VehicleA. We return always to the initial state once one of these actions is completed. However, once action *entranceOn*, which models the presence of an entrance lane, is performed, the component *Situation*2*A* behaves as *Situation*2*AEnLOn*.

$$\begin{aligned} \mathit{Situation2AEnLOn} = {} & (\mathit{runVehicleEGO}, \top).\mathit{Situation2AEnLOn} + (\mathit{accelerateVehicleEGO}, \top).\mathit{Situation2AEnLOn} \\ & + (\mathit{decelerateVehicleEGO}, \top).\mathit{Situation2AEnLOn} + (\mathit{runVehicleA}, \top).\mathit{Situation2AEnLOn} \\ & + (\mathit{noEnterV}, \top).\mathit{Situation2A} + (\mathit{enterV}, \top).\mathit{Situation2AEnterV}; \end{aligned}$$

In the component state *Situation*2*AEnLOn*, Ego has always the choice between its three actions and vehicle *VA* can always perform action *runVehicleA*. We always return to the same state after completing one of these actions. Action *noEnterV* models the absence of a vehicle on the entrance lane. Its completion leads to state *Situation*2*A*. Action *enterV* models the presence of a vehicle on the entrance lane. Once this action is performed, it leads to state *Situation*2*AEnterV*:

$$\begin{aligned} \mathit{Situation2AEnterV} = {} & (\mathit{runVehicleEGO}, \top).\mathit{Situation2AEnterV} + (\mathit{accelerateVehicleEGO}, \top).\mathit{Situation2AEnterV} \\ & + (\mathit{decelerateVehicleEGO}, \top).\mathit{Situation2AEnterV} + (\mathit{runVehicleA}, \top).\mathit{Situation2AEnterV} \\ & + (\mathit{runVehicleB}, \top).\mathit{Situation2AEnterV} + (\mathit{accelerateVehicleB}, \top).\mathit{Situation2AEnterV} \\ & + (\mathit{decelerateVehicleB}, \top).\mathit{Situation2AEnterV} + (\mathit{signABon}, \top).\mathit{Situation2AEnterVDecelerate} \\ & + (\mathit{goLeftLaneVehicleB}, w1\top).\mathit{Situation2AB} + (\mathit{goLeftLaneVehicleB}, w2\top).\mathit{Situation2A3B} \\ & + (\mathit{goLeftLaneVehicleB}, w3\top).\mathit{Situation2A4B} + (\mathit{goLeftLaneVehicleB}, w4\top).\mathit{Situation2A5B}; \end{aligned}$$

In state *Situation*2*AEnterV*, Ego has the choice between its actions and *VA* can perform action *runVehicleA*. VB has the choice between actions *runVehicleB*, *accelerateVehicleB* and *decelerateVehicleB*. The component always returns to state *Situation*2*AEnterV* after these actions completion.

Action *signABon* is performed if there is a sign of type *SignAB* on the entrance lane. Here it refers to the *give way* sign. If this action is performed, the component behaves as *Situation*2*AEnterVDecelerate*. When VB rolls on the entrance lane and does action $goLeftLaneVehicleB$, it may arrive to Zone 2, Zone 3, Zone 4 or Zone 5. Therefore we assign to the rates of actions a weight w_i, where $i \in \{1,2,3,4\}$ to simulate the possibility of VB entering each zone. The component behaves as *Situation*2*AB*, *Situation*2*A*3*B*, *Situation*2*A*4*B*, and *Situation*2*A*5*B*, if VB arrives to Zone 2, Zone 3, Zone 4 or Zone 5, respectively.

$$\begin{aligned} \mathit{Situation2AEnterVDecelerate} = {} & (\mathit{runVehicleEGO}, \top).\mathit{Situation2AEnterVDecelerate} \\ & + (\mathit{accelerateVehicleEGO}, \top).\mathit{Situation2AEnterVDecelerate} \\ & + (\mathit{decelerateVehicleEGO}, \top).\mathit{Situation2AEnterVDecelerate} \\ & + (\mathit{runVehicleA}, \top).\mathit{Situation2AEnterVDecelerate} \\ & + (\mathit{runVehicleB}, \top).\mathit{Situation2AEnterVDecelerate} \\ & + (\mathit{accelerateVehicleB}, \top).\mathit{Situation2AEnterVDecelerate} \\ & + (\mathit{decelerateVehicleB}, p1\top).\mathit{Situation2AEnterVDecelerate} \\ & + (\mathit{signABoff}, \top).\mathit{Situation2AEnterV}; \end{aligned}$$

In this state, *Ego*, *VA* and VB have the choice between their possible actions. After seeing the *give way* sign, VB is more likely to decelerate. We assign the activity rate a weight p_1 to adapt the frequency of the occurrence of action $decelerateVehicleB$. The component returns always to $Situation2AEnterVDecelerate$ once this action is completed. Action $signABoff$ models the end of the impact of the sign of type *SignAB*. This action makes the component return to $Situation2AEnterV$.

$$\begin{aligned}Situation2AB = {} & (runVehicleEGO, \top).Situation2AB + (accelerateVehicleEGO, w5\top).Situation2AB \\ & + (accelerateVehicleEGO, w6\top).Situation2A3B + (decelerateVehicleEGO, \top).Situation2AB \\ & + (runVehicleA, \top).Situation2AB + (runVehicleB, \top).Situation2AB \\ & + (accelerateVehicleB, \top).Situation2AB + (decelerateVehicleB, w7\top).Situation2AB \\ & + (decelerateVehicleB, w8\top).Situation2A3B;\end{aligned}$$

State $Situation2AB$ represents the situation where *VA* and VB are in Zone 2. The component has the chance to behave as $Situation2A3B$ if action $accelerateVehicleEGO$ or action $decelerateVehicleB$ is performed. We assign to rate of the first action a weight w_i, where $i \in \{5,6\}$ to simulate the possibility that VB stays in Zone 2, or enters Zone 3, respectively. While we assign to rate of the second action a weight w_i, where $i \in \{7,8\}$ to simulate the similar behaviours when VB performs the first action.

$$\begin{aligned}Situation2A3B = {} & (runVehicleEGO, \top).Situation2A3B + (accelerateVehicleEGO, \top).Situation2A3B \\ & + (decelerateVehicleEGO, w9\top).Situation2AB + (decelerateVehicleEGO, w10\top).Situation2A3B \\ & + (runVehicleA, \top).Situation2A3B + (runVehicleB, \top).Situation2A3B \\ & + (accelerateVehicleB, w11\top).Situation2AB + (accelerateVehicleB, w12\top).Situation2A3B \\ & + (decelerateVehicleB, \top).Situation2A3B;\end{aligned}$$

State $Situation2A3B$ models the situation where *VA* is in Zone 2 and VB is in Zone 3. From this state, we may have the chance to reach state $Situation2AB$ if action $decelerateVehicleEGO$ or action $accelerateVehicleB$ is performed. We assign to rate of the first action a weight w_i, where $i \in \{9,10\}$ to simulate the possibility that VB enters Zone 2, or stays in Zone 3, respectively. While we assign to rate of the second action a weight w_i, where $i \in \{11,12\}$ to simulate the similar behaviours when VB performs the first action.

$$\begin{aligned}Situation2A4B = {} & (runVehicleEGO, \top).Situation2A4B + (accelerateVehicleEGO, w13\top).Situation2A4B \\ & + (accelerateVehicleEGO, w14\top).Situation2A5B + (decelerateVehicleEGO, \top).Situation2A4B \\ & + (runVehicleA, \top).Situation2A4B + (runVehicleB, \top).Situation2A4B \\ & + (accelerateVehicleB, \top).Situation2A4B + (decelerateVehicleB, w15\top).Situation2A4B \\ & + (decelerateVehicleB, w16\top).Situation2A5B;\end{aligned}$$

Sate $Situation2A4B$ represents the situation where *VA* is in Zone 2 and VB is in Zone 4. The component has the chance to behave as $Situation2A5B$ if action $accelerateVehicleEGO$ or action $decelerateVehicleB$ is performed. We assign to rate of the first action a weight w_i, where $i \in \{13,14\}$ to simulate the possibility that VB stays in Zone 4, or enters Zone 5, respectively. While we assign to rate of the second action a weight w_i, where $i \in \{15,16\}$ to simulate the

similar behaviours when VB performs the first action.

$$\begin{aligned}Situation2A5B = &(runVehicleEGO, \top).Situation2A5B + (accelerateVehicleEGO, \top).Situation2A5B \\ &+ (decelerateVehicleEGO, w17\top).Situation2A4B + (decelerateVehicleEGO, w18\top).Situation2A5B \\ &+ (runVehicleA, \top).Situation2A5B + (runVehicleB, \top).Situation2A5B \\ &+ (accelerateVehicleB, w19\top).Situation2A4B + (accelerateVehicleB, w20\top).Situation2A5B \\ &+ (decelerateVehicleB, \top).Situation2A5B;\end{aligned}$$

State $Situation2A5B$ represents the situation where VA is in Zone 2 and VB is in Zone 5. The component has the chance to behave as $Situation2A4B$ if action $decelerateVehicleEGO$ or action $accelerateVehicleB$ is performed. We assign to rate of the first action a weight w_i, where $i \in \{17, 18\}$ to simulate the possibility that VB enters Zone 4, or stays in Zone 5, respectively. While we to rate of the second action a weight w_i, where $i \in \{19, 20\}$ to simulate the similar behaviours when VB performs the first action.

Finally, the equation of the complete PEPA model of our example is the following:

$$Scenario \stackrel{\text{def}}{=} Situation \underset{L_1}{\bowtie} (\ VehicleEGO \ || \ VehicleA \ || \ VehicleB \ || \ EnLOff \ || \ SignABOff \ )$$

where L_1 is the actions set on which components $VehicleEGO$, $VehicleA$, $VehicleB$, $EnLOff$ and $SignABOff$ must cooperate individually with component $Situation$. It is defined as:

$$\begin{aligned}L_1 = \{&runVehicleEGO, accelerateVehicleEGO, decelerateVehicleEGO, runVehicleA, \\ &goLeftLaneVehicleB, runVehicleB, accelerateVehicleB, decelerateVehicleB, \\ &entranceOn, enterV, noEnterV, signABoff, signABon\}.\end{aligned}$$

In order to test our model, we define a set of values for the rates of the actions and the assigned weights.

Currently, we can calculate the rates of some actions such as the actions corresponding to concepts $Accelerate$ and $Decelerate$. For example, the rates of action $accelerateVehicleB$ can be calculated using VB's initial speed v_1, final speed v_2 and the distance d between the initial state and the final state of VB. The initial speed is the speed of VB before action $accelerateVehicleB$ is performed, which is $75km/h$. The final speed is the speed of VB after action $accelerateVehicleB$ is performed, which is $85km/h$ and the distance is $0.1km$. We can get the rate b_2 of this action using standard kinetic:

$$b_2 = \frac{v_2^2 - v_1^2}{2d\Delta v} = \frac{85^2 - 75^2}{2 \times 0.1 \times 10} = 800\ h^{-1}$$

Unlike the values of the rates associated with actions $Accelerate$ and $Decelerate$, the values of the other actions rates are set arbitrary because we unfortunately do not have them. The users of our tool can set their values according to their needs.

Table 7.3 presents the parameters values for the activities of components *VehicleEGO*, *VehicleA*, *VehicleB*, $EnLOff$ and $SignABOff$. The values of the weights for this example are given arbitrarily and they are shown in Table 7.4.

TABLE 7.3 – Activities of components

Action	Rate	
runVehicleEGO	e_1	500
accelerateVehicleEGO	e_2	900
decelerateVehicleEGO	e_3	300
runVehicleA	a_1	400
runVehicleB	b_1	400
accelerateVehicleB	b_2	800
decelerateVehicleB	b_3	200
goLeftLaneVehicleB	b_4	600
entranceOn	u_1	50
enterV	u_2	1
noEnterV	u_3	60
signABoff	u_4	800
signABon	u_5	800

TABLE 7.4 – Weights of rates

Weight		Weight	
p_1	5		
w_1	1	w_{11}	5
w_2	2	w_{12}	5
w_3	3	w_{13}	8
w_4	4	w_{14}	2
w_5	4	w_{15}	4
w_6	6	w_{16}	6
w_7	7	w_{17}	2
w_8	3	w_{18}	8
w_9	6	w_{19}	7
w_{10}	4	w_{20}	3

PEPA abstracts the activities performed by the components into a continuous-time Markov process. The PEPA model has 8 states. Each state is a tuple composed of the states of the components *Situation*, *VehicleEGO*, *VehicleA*, *VehicleB*, $EnLOff$ and $SignABOff$ as shown in Table 7.5. The initial state is (*Situation*2*A*, *VehicleEGO*, *VehicleA*, *VehicleB*, $EnLOff$, $SignABOff$).

TABLE 7.5 – State space of model PEPA

State	Situation	Ego	VA	VB	Entrance Lane	SignAB
State 1	*Situation2A*	*VehicleEGO*	*VehicleA*	*VehicleB*	$EnLOff$	$SignABOff$
State 2	*Situation2AEnLOn*	*VehicleEGO*	*VehicleA*	*VehicleB*	*EnLOn*	$SignABOff$
State 3	*Situation2AEnterV*	*VehicleEGO*	*VehicleA*	*VehicleB*	$EnLOff$	$SignABOff$
State 4	*Situation2AB*	*VehicleEGO*	*VehicleA*	*VehicleB*	$EnLOff$	$SignABOff$
State 5	*Situation2A***3***B*	*VehicleEGO*	*VehicleA*	*VehicleB*	$EnLOff$	$SignABOff$
State 6	*Situation2A***4***B*	*VehicleEGO*	*VehicleA*	*VehicleB*	$EnLOff$	$SignABOff$
State 7	*Situation2A5B*	*VehicleEGO*	*VehicleA*	*VehicleB*	$EnLOff$	$SignABOff$
State 8	*Situation2AEnterVDecelerate*	*VehicleEGO*	*VehicleA*	*VehicleB*	$EnLOff$	*SignABOn*

We can generate all possible scenarios and identify the critical ones from the transition graph of the whole model. One scenario includes one or several states in the transition graph which are connected. Each state of the model is regarded as a scene of a scenario. The states are the nodes of the transition graph, and the activities are the labels on the transitions. A scenario is considered as a path in the transition graph which includes at least one state. Critical scenarios are those which include critical states.

Zone 3 and Zone 4 being the critical zones, in the running example, component states *Situation*2*A*3*B* and *Situation*2*A*4*B* indicate the critical situations where VB rolls in Zone 3 and

Zone 4, respectively. Model states *State 5* and *State 6* (see Table 7.5), which include the critical situations, are regarded as the critical scenes. All the scenarios which include either state *State 5* or *State 6* are critical scenarios which may lead to accidents.

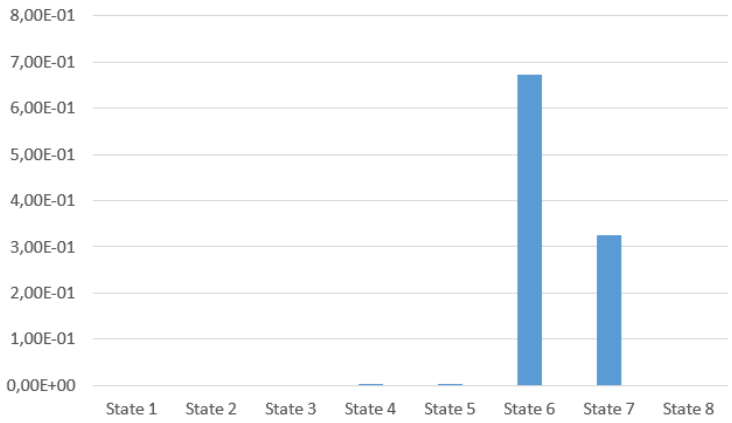

FIGURE 7-5 – Steady-state probability distribution.

We can get the steady-state probability distribution using Eclipse PEPA [Hillston and Gilmore, 2014]. The probability of being in each state is provided in Figure 7-5. We can clearly see that system states *State 6* and *State 7* have the highest probabilities, and the sum of these probabilities is close to 1. Both states indicate that VB inserts after Ego. This means that case 2 (VB inserts after Ego) mentioned in Chapter 4 is the most likely to occur according to the sets of the values used in Table 7.3 and Table 7.4. This is normal as we considered that the *give way* sign makes VB more likely to decelerate.

7.5 Automatic Generation

In this section, we are interested in an automatic generation method which allows generating test cases that have an impact on the performance and the dependability of the vehicle. Because of the size of PEPA model, we propose an algorithm to generate automatically PEPA models in Sub-section 7.5.1. Then we propose an approach to generate automatically the test cases in Sub-section 7.5.2. Finally, a method to identify the critical test cases is presented in Sub-section 7.5.3.

7.5.1 Generation of the PEPA model

Because of the large number of the highway infrastructure elements, the number of possible vehicles, and the weather conditions, writing the PEPA components specially component *Situation* is tedious. Thus, we propose an algorithm to generate automatically PEPA models.

The main steps of this algorithm are the following:

Step 1. Set the number of the vehicles of type *Other-Car* in the model;

Step 2. Initialize all the PEPA components of the vehicles, the infrastructure and visibility with the corresponding equations in Section 7.3;

Step 3. Initialize the component *Situation* with the scene which involves only the autonomous vehicle:

$$Situation = (runEGO, \top).Situation + (accelerateEGO, \top).Situation + (decelerateEGO, \top).Situation;$$

Step 4. Each time a component is added, the equation of component *Situation* adds the corresponding activities. For example, once component *EntranceLane* is added, the equation of *Situation* needs to include the terms "$+ (entranceOn, \top).SituationEnLOn$". Component *Situation* becomes:

$$\begin{aligned} Situation = {} & (runEGO, \top).Situation + (accelerateEGO, \top).Situation \\ & + (decelerateEGO, \top).Situation + (entranceOn, \top).SituationEnLOn; \end{aligned}$$

Step 5. Once a new state is added, the model is not closed-loop. For example, once action *entranceOn* is performed, component *Situation* behaves as *SituationEnLOn*. We need to complete the model with the following equation:

$$\begin{aligned} SituationEnLOn = {} & (runVehicleEGO, \top).SituationEnLOn \\ & + (accelerateVehicleEGO, \top).SituationEnLOn \\ & + (decelerateVehicleEGO, \top).SituationEnLOn \\ & + (noEnterV, \top).Situation + (enterV, \top).SituationEnV; \end{aligned}$$

Step 6. If the model is not close-loop, go to Step 5; otherwise, build the set L of synchronisation actions. L contains all the actions of component *Situation*;

Step 7. Write the complete PEPA model equation which shows how the model components interact, and on which actions set they interact.

Step 8. Set all the rates and the weights to the selected values.

To start this algorithm, we need to set the number of the vehicles of type $Other - Car$ and set $TRUE$ to all necessary highway infrastructure. Then a file with '.pepa' suffix will be generated, which stores a complete PEPA model required.

7.5.2 Generation of the Test Cases

We propose an approach that allows generating automatically the test cases from any initial situation and with any number of scenes. The main steps of this approach, which is integrated to the third layer of our methodology, are the following:

Step 1. Generate all the states of the PEPA model with Eclipse PEPA. These states are regarded as the scenes of the autonomous vehicle;

Step 2. Find the pairs of states where a completed action in one of these states leads to the other state;

Step 3. Generate the transition graph with the list of pairs of states;

Step 4. Set a number **D** to indicate the length of the scenarios that need to be generated, that is, the number of scenes of each scenario;

Step 5. Set the initial state *Situation* to begin the program;

Step 6. Return a list of scenarios from state *Situation* with **D** scenes in each scenario.

We need to prepare a file named 'activities.txt' of the components' equations except the component *Situation* which is stored in another file named 'situations.txt', and a file named 'rates.txt' that stores all activity rates and weights of possibilities. After the PEPA model is analyzed by Eclipse PEPA, a file with the suffix '.statespace' will be generated, which contains the steady-state probability distribution of PEPA model. In addition to these four files, we also need to set a number **D** to indicate the length of the scenarios that need to be generated and set the initial state *Situation* to begin the program. Then a file named 'test cases.txt' will be generated, which stores a list of scenarios (test cases) from state *Situation* with **D** scenes in each scenario.

7.5.3 Identification of the critical Test Cases

We want not only to find the critical scenarios, but also to sort these scenarios according to their criticality. Therefore, we propose to calculate their criticality *CS* as follows:

Suppose in a scenario S_c, we have D states named $T_1 ... T_D$. If $T_i, i \in [1, D]$ is critical, the criticality C_i of state T_i is set as C_i=1. Otherwise if T_i is not critical, C_i=0. The criticality of S_c can be calculated as follows:

$$CS = \frac{\sum_{C_1}^{C_D}}{D} \tag{7.1}$$

We also propose a method to calculate the criticality of each test case to evaluate comprehensively its importance:

Step 1. Each derivative state of component *Situation* is referred to by the name and the location of all the vehicles in this state (see Section 7.3).

Step 2. Zone 3 and Zone 4 being the critical zones, all the derivative states of component *Situation* with 3 or 4 in their name are critical states.

Step 3. All scenarios that contain one or several critical states are critical scenarios.

Step 4. Calculate the criticality of each scenario using formula (7.1).

7.6 Conclusion

In this chapter, we introduced the formal modelling language PEPA. We described more specifically the construction of the PEPA model using the running example presented in Chapter 4. We also proposed the approaches for the automatic generation of the PEPA model and the test cases, specially the critical ones with specific examples.

In the next chapter, we consider two case studies on which we apply our test cases generation methodology. And we give the critical test cases and their probabilities of occurrence in each case study.

Chapitre 8

Case Studies

8.1 Introduction

We have defined a test cases generation methodology in Chapter 4. With this methodology, we can generate the test cases for the context of the highway with entrance lanes, exit lanes and different road signs. The special road parts such as tolls and road works are also considered in our methodology. The weather conditions are represented as visibility. And all the other possibles vehicles around the autonomous vehicle are included.

In this Chapter, we investigate two case studies. The first case "*One vehicle of type Other-Car riding with autonomous vehicle*" is presented in Section 8.2. This case has only one vehicle riding around the autonomous vehicle. We chose this small model to show the complete steps allowing the generation of the scenarios and especially the critical ones. The Second case study "*One vehicle of type Other-Car riding with autonomous vehicle in the context of general highway infrastructure*" is a more complex case where traffic signs, entrance lanes, exit lanes, tolls and the visibility are considered. We chose it to show how we generate the PEPA model with different infrastructure elements and the visibility as the weather impact on the autonomous vehicle. This case study is presented in Section 8.3. Finally, we conclude this chapter in Section 8.4.

8.2 Case Study 1: Autonomous Vehicle in a Simple Context

We consider the situation "*One vehicle of type Other-Car riding with autonomous vehicle*" as a case study to show how to build the PEPA model, and how to generate all the scenarios, specially the critical ones.

The initial situation consists of static and mobile elements, which are presented in the following.

FIGURE 8-1 – Scenography

The highway is separated into two carriageways by a median. In the scenography of this example (Figure 8-1), a portion of one carriageway is selected. This carriageway has three through lanes: the right lane–$Lane_1$, the center lane–$Lane_2$ and the left lane–$Lane_3$. This portion of the road can be extended indefinitely.

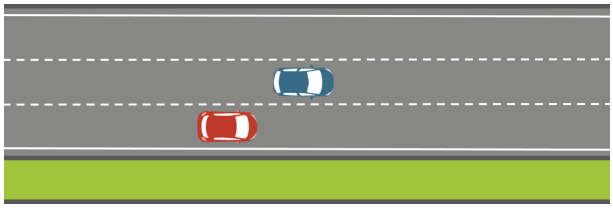

FIGURE 8-2 – Initial scene.

In the initial scene (Figure 8-2), the autonomous vehicle namely *Ego* (blue) rolls on $Lane_2$ of a separated lane road. One (1) vehicle *VA* (red) of type *OtherCar* is on $Lane_1$.

In this case study, the static elements are: highway, carriageway, median, through lane. The mobile elements are: autonomous vehicle *Ego*, other vehicle *VA*.

8.2.1 PEPA model

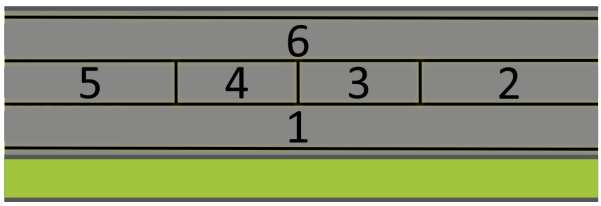

FIGURE 8-3 – Zones' numbers in the scene.

As in Chapter 7, we note the zones from one to six (Figure 8-3). Zone 1 indicates $Lane_1$. Zone 2 and Zone 5 indicate the uncritical zones in front and behind *Ego*. Zone 3 indicates the critical zone in front of *Ego* while Zone 4 indicates the critical one behind it. Zone 6 indicates $Lane_3$. Both Zone 1 and Zone 6 are uncritical zones for *Ego*.

The PEPA model modelling our system consists of three (3) components: *VehicleEGO*, *VehicleA* and *Situation*1*A*. These model the behaviour of the *Ego* car, vehicle *VA*, and the situation itself, respectively. The initial situation *Situation*1*A* indicates that *Ego* is rolling on the center lane and *VA* is rolling on Zone 1. The PEPA components and their actions are shown in Table 8.1. All these components and actions have been presented in Chapter 7.

TABLE 8.1 – Components and actions of PEPA model

Componengts	Actions
VehicleEGO	*runVehicleEGO*, *accelerateVehicleEGO*, *decelerateVehicleEGO*
VehicleA	*runVehicleA*, *accelerateVehicleA*, *decelerateVehicleA*, *goLeftLaneVehicleA*, *goRightLaneVehicleA*
Situation1A	*runVehicleEGO*, *accelerateVehicleEGO*, *decelerateVehicleEGO*, *runVehicleA*, *accelerateVehicleA*, *decelerateVehicleA*, *goLeftLaneVehicleA*, *goRightLaneVehicleA*

The PEPA equations of the sequential components *VehicleEGO* and *VehicleA* are as follows:

$$\begin{aligned} VehicleEGO = {} & (runVehicleEGO, e_1).VehicleEGO + (accelerateVehicleEGO, e_2).VehicleEGO \\ & + (decelerateVehicleEGO, e_3).VehicleEGO; \end{aligned}$$

$$\begin{aligned} VehicleA = {} & (runVehicleA, a1).VehicleA + (accelerateVehicleA, a2).VehicleA \\ & + (decelerateVehicleA, a3).VehicleA + (goLeftLaneVehicleA, a4).VehicleA \\ & + (goRightLaneVehicleA, a5).VehicleA; \end{aligned}$$

The above equations are similar to those presented in Chapter 7

The PEPA component *Situation*1*A* has six (6) states. The behaviour of this component at its initial state is given by the following equation:

$$\begin{aligned} Situation1A = {} & (runVehicleEGO, \top).Situation1A + (accelerateVehicleEGO, \top).Situation1A \\ & + (decelerateVehicleEGO, \top).Situation1A + (runVehicleA, \top).Situation1A \\ & + (accelerateVehicleA, \top).Situation1A + (decelerateVehicleA, \top).Situation1A \\ & + (goLeftLaneVehicleA, w1\top).Situation2A + (goLeftLaneVehicleA, w2\top).Situation3A \\ & + (goLeftLaneVehicleA, w3\top).Situation4A + (goLeftLaneVehicleA, w4\top).Situation5A; \end{aligned}$$

As stated in its equation, *Ego* has the choice between actions *runVehicleEGO*, *accelerateVehicleEGO* and *decelerateVehicleEGO*. Similarly, according to its equation, *VA* has the choice between actions *runVehicleA*, *accelerateVehicleA*, *decelerateVehicleA* and *goLeftLaneVehicleA*.

As stated in component *Situation*1*A* equation, component *VehicleEGO* can perform its actions only in synchronisation with *Situation*1*A*. Similarly for the actions *runVehicleA*, *accelerateVehicleA* and *decelerateVehicleA* which can be completed by *VehicleA* only in cooperation with component *Situation*1*A*. Once one of these actions has been performed, component *Situation1A* remains in its initial state.

Once action *goLeftLaneVehicleA* is performed, *VA* may arrive to Zone 2, Zone 3, Zone 4 or Zone 5. Therefore we assign to the rate of this action a weight w_i, where $i \in \{1,2,3,4\}$ to simulate the possibility of *VA* entering the corresponding zone. This action leads to state *Situation*2*A*, *Situation*3*A*, *Situation*4*A*, and *Situation*5*A* according to the zone where *VA* arrives.

$$\begin{aligned}
Situation2A = {} & (runVehicleEGO, w5\top).Situation2A + (runVehicleEGO, w6\top).Situation3A \\
& + (accelerateVehicleEGO, w7\top).Situation2A + (accelerateVehicleEGO, w8\top).Situation3A \\
& + (decelerateVehicleEGO, \top).Situation2A + (runVehicleA, w9\top).Situation2A \\
& + (runVehicleA, w10\top).Situation3A + (accelerateVehicleA, \top).Situation2A \\
& + (accelerateVehicleA, w11\top).Situation2A + (decelerateVehicleA, w12\top).Situation3A \\
& + (goLeftLaneVehicleA, \top).Situation6A + (goRightLaneVehicleA, \top).Situation1A;
\end{aligned}$$

The derivative state *Situation*2*A* represents the situation where *VA* is in Zone 2. It can become *Situation*3*A* if one of the actions in $\{runVehicleEGO, accelerateVehicleEGO, runVehicleA, decelerateVehicleA\}$ is performed. This is because the location of the zones is related to the location of Ego. Zones move forward with Ego. The first action *runVehicleEGO* can change the location of *VA* to Zone 3 because *Ego* can be faster than *VA*. This is similar to the effect of action *accelerateVehicleEGO* if preformed. Action *runVehicleA* can change the location of *VA* to Zone 3 because *VA* can be slower than *Ego*. This is similar to the effect of action *decelerateVehicleA* if preformed. These reasons apply to all derivative states in the following.

We assign to the rates of the actions in the set a weight w_i, where $i \in \{6,8,10,12\}$ to simulate the possibility that *VA* enters Zone 3. Similarly we assign them a weight w_i, where $i \in \{5,7,9,11\}$ to simulate the possibility that *VA* remains in Zone 2. Once action *goLeftLaneVehicleA* is performed, *VA* arrives to Zone 6, while if action *goRightLaneVehicleA* is performed, *VA* arrives to Zone 1.

$$\begin{aligned}
Situation3A = {} & (runVehicleEGO, w13\top).Situation2A + (runVehicleEGO, w14\top).Situation3A \\
& + (accelerateVehicleEGO, \top).Situation3A + (decelerateVehicleEGO, w15\top).Situation2A \\
& + (decelerateVehicleEGO, w16\top).Situation3A + (runVehicleA, w17\top).Situation2A \\
& + (runVehicleA, w18\top).Situation3A + (accelerateVehicleA, w19\top).Situation2A \\
& + (accelerateVehicleA, w20\top).Situation3A + (decelerateVehicleA, \top).Situation3A \\
& + (goLeftLaneVehicleA, \top).Situation6A + (goRightLaneVehicleA, \top).Situation1A;
\end{aligned}$$

The derivative state *Situation*3*A* represents the situation where *VA* is in Zone 3. It can become *Situation*2*A* if one of the actions in $\{runVehicleEGO, decelerateVehicleEGO, runVehicleA, accelerateVehicleA\}$ is performed. We assign to the rates of these actions a weight w_i, where $i \in \{13,15,17,19\}$ to simulate the possibility that *VA* enters Zone 2. Similarly we assign them a weight w_i, where $i \in \{14,16,18,20\}$ to simulate the possibility that *VA* remains in Zone 3. Once action *goLeftLaneVehicleA* is performed, *VA* arrives to Zone 6, while if action

goRightLaneVehicleA is performed, *VA* arrives to Zone 1.

$$\begin{aligned}
Situation4A = {} & (runVehicleEGO, w21\top).Situation4A + (runVehicleEGO, w22\top).Situation5A \\
& + (accelerateVehicleEGO, w23\top).Situation4A + (accelerateVehicleEGO, w24\top).Situation5A \\
& + (decelerateVehicleEGO, \top).Situation4A + (runVehicleA, w25\top).Situation4A \\
& + (runVehicleA, w26\top).Situation5A + (accelerateVehicleA, \top).Situation4A \\
& + (decelerateVehicleA, w27\top).Situation4A + (decelerateVehicleA, w28\top).Situation5A \\
& + (goLeftLaneVehicleA, \top).Situation6A + (goRightLaneVehicleA, \top).Situation1A;
\end{aligned}$$

The derivative state *Situation*4*A* represents the situation where *VA* is in Zone 4. It can become *Situation*5*A* if one of the actions in {*runVehicleEGO*,*accelerateVehicleEGO*, *runVehicleA*, *decelerateVehicleA*} is performed. We assign to the rates of these actions a weight w_i, where $i \in \{22, 24, 26, 28\}$ to simulate the possibility that *VA* enters Zone 5. Similarly we assign them a weight w_i, where $i \in \{21, 23, 25, 27\}$ to simulate the possibility that *VA* remains in Zone 4. Once action *goLeftLaneVehicleA* is performed, *VA* arrives to Zone 6, while if action *goRightLaneVehicleA* is performed, *VA* arrives to Zone 1.

$$\begin{aligned}
Situation5A = {} & (runVehicleEGO, w29\top).Situation4A + (runVehicleEGO, w30\top).Situation5A \\
& + (accelerateVehicleEGO, \top).Situation5A + (decelerateVehicleEGO, w31\top).Situation4A \\
& + (decelerateVehicleEGO, w32\top).Situation5A + (runVehicleA, w33\top).Situation4A \\
& + (runVehicleA, w34\top).Situation5A + (accelerateVehicleA, w35\top).Situation4A \\
& + (accelerateVehicleA, w36\top).Situation5A + (decelerateVehicleA, \top).Situation5A \\
& + (goLeftLaneVehicleA, \top).Situation6A + (goRightLaneVehicleA, \top).Situation1A;
\end{aligned}$$

The derivative state *Situation*5*A* represents the situation where *VA* is in Zone 5. It can become *Situation*4*A* if one of the actions in {*runVehicleEGO*, *decelerateVehicleEGO*, *runVehicleA*, *accelerateVehicleA*} is performed. We assign to the rates of these actions a weight w_i, where $i \in \{29, 31, 33, 35\}$ to simulate the possibility that *VA* enters Zone 4. Similarly we assign them a weight w_i, where $i \in \{30, 32, 34, 36\}$ to simulate the possibility that *VA* remains in Zone 5. Once action *goLeftLaneVehicleA* is performed, *VA* arrives to Zone 6, while if action *goRightLaneVehicleA* is performed, *VA* arrives to Zone 1.

$$\begin{aligned}
Situation6A = {} & (runVehicleEGO, \top).Situation6A + (accelerateVehicleEGO, \top).Situation6A \\
& + (decelerateVehicleEGO, \top).Situation6A + (runVehicleA, \top).Situation6A \\
& + (accelerateVehicleA, \top).Situation6A + (decelerateVehicleA, \top).Situation6A \\
& + (goRightLaneVehicleA, w37\top).Situation2A + (goRightLaneVehicleA, w38\top).Situation3A \\
& + (goRightLaneVehicleA, w39\top).Situation4A + (goRightLaneVehicleA, w40\top).Situation5A;
\end{aligned}$$

In component state *Situation*6*A*, *Ego* has the choice between its usual three actions. Similarly, *VA* has the choice between actions *runVehicleA*, *accelerateVehicleA*, *decelerateVehicleA* and *goRightLaneVehicleA*.

Both components *VehicleEGO* and *VehicleA* can perform their actions only in synchronisation with *Situation*6*A*. Once one of these actions has been performed, component *Situation*6*A*

remains in its state.

However, once action *goRightLaneVehicleA* is performed, *VA* may arrive to Zone 2, Zone 3, Zone 4 or Zone 5. Therefore we assign to the rates of these actions a weight w_i, where $i \in \{45,46,47,48\}$ to simulate the possibility of *VA* entering the corresponding zone. From *Situation*6*A*, we can go to *Situation*2*A*, *Situation*3*A*, *Situation*4*A*, and *Situation*5*A* according to the zone where *VA* arrives.

The complete PEPA model equation is the following:

$$Scenario \stackrel{\text{def}}{=} Situation1A \underset{L_2}{\bowtie} (VehicleEGO \parallel VehicleA)$$

where L_2 is the actions set on which components *VehicleEGO* and *VehicleA* must synchronise individually with *Situation*1*A*. It is defined as:

$$L_2 = \{runVehicleEGO, accelerateVehicleEGO, decelerateVehicleEGO, goLeftLaneVehicleA, goRightLaneVehicleA, runVehicleA, accelerateVehicleA, decelerateVehicleA\}.$$

8.2.2 Numerical Results

PEPA abstracts the activities performed by components into a continuous-time Markov process. The underlying Markov Chain of the PEPA model has 6 states. They are:

State 1: $\{Situation1A, VehicleEGO, VehicleA\}$; *Ego* is rolling on the center lane and *VA* is on Zone 1;
State 2: $\{Situation2A, VehicleEGO, VehicleA\}$; *Ego* is rolling on the center lane and *VA* is on Zone 2;
State 3: $\{Situation3A, VehicleEGO, VehicleA\}$; *Ego* is rolling on the center lane and *VA* is on Zone 3;
State 4: $\{Situation4A, VehicleEGO, VehicleA\}$; *Ego* is rolling on the center lane and *VA* is on Zone 4;
State 5: $\{Situation5A, VehicleEGO, VehicleA\}$; *Ego* is rolling on the center lane and *VA* is on Zone 5;
State 6: $\{Situation6A, VehicleEGO, VehicleA\}$; *Ego* is rolling on the center lane and *VA* is on Zone 6.

TABLE 8.2 – Set of rates

Rate	a1	a2	a3	a4	a5	e1	e2	e3
Value	400	800	200	600	700	500	900	300

We can get the steady-state probability distribution using the tool Eclipse Plug-in for PEPA. The probability of being in each state is provided in Figure 8-4. These probabilities are based

TABLE 8.3 – Set of Weight

Weight	Value	Weight	Value	Weight	Value	Weight	Value
w1	8	w11	2	w21	7	w31	6
w2	9	w12	4	w22	8	w32	6
w3	5	w13	2	w23	10	w33	5
w4	10	w14	8	w24	5	w34	4
w5	10	w15	4	w25	9	w35	7
w6	5	w16	10	w26	9	w36	7
w7	7	w17	3	w27	2	w37	8
w8	5	w18	9	w28	7	w38	4
w9	2	w19	5	w29	8	w39	5
w10	1	w20	1	w30	7	w40	3

on the activity rates in Table 8.2 and the weights in Table 8.3. These values are automatically generated with our algorithm (see Chapter 7).

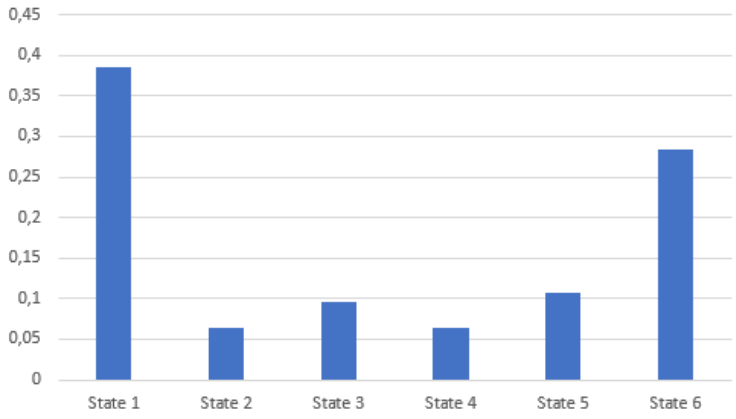

FIGURE 8-4 – Steady-state probability distribution

Figure 8-4 shows that the sum of the probability of the system being in State 1 (0.38582677165354323) and State 6 (0,283464566929133) is close to $2/3$. The sum of the remaining probabilities, that is of being in the other states (State 2, State 3, State 4, State 5) is close to $1/3$. Thus according to the obtained results, *VA* has more chance to be in Zone 1 as the highest probability is the one to be in State 1. We can learn from Figure 8-5, which provides the probability of *VA* being in each lane, that *VA* has then 0,330708661417322 probability to be in Lane 2. Moreover, in Lane 2, *VA* has more chance to be in Zone 3 and Zone 5. *VA* has the lowest probability to be in Lane 3. This distribution is reasonable as *VA* enters the highway from Lane 1 before it is possible to enter Lane 2 and further enter Lane 3.

We implemented the algorithm generating the scenarios (see Chapter 7). With our program, we can generate scenarios with any initial state and any number of scenes. For example, we want to generate all possible scenarios with a length no greater than 2 scenes per scenario from State 1. In this case, we get ten (10) eligible scenarios (Table 8.4). This table shows that

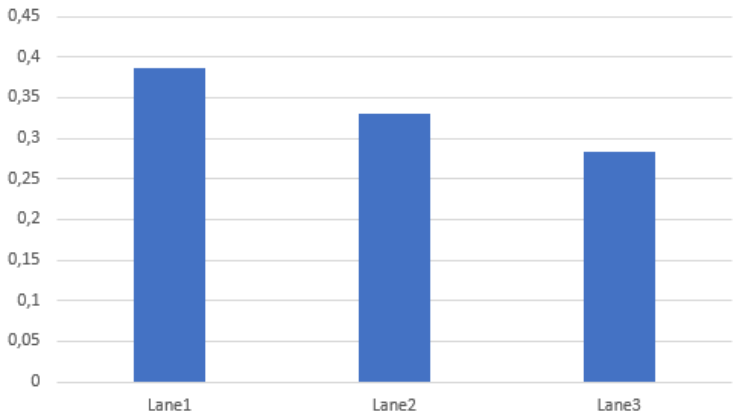

FIGURE 8-5 – Lane probability distribution

for Scenario 1, the system is in State 1 and once the action *runVehicleEGO* is performed, the system remains its initial state. For Scenario 10, once the action $goLeftLaneVehicleA$ is performed, the system is in State 5. We can get different test cases by assigning different values to the rates (e_i, a_i, w_i) in these scenarios.

TABLE 8.4 – Test cases of length 2 from State 1

Scenario	Initial State	Action	Next State
1	State 1	*runVehicleEGO*	State 1
2	State 1	*accelerateVehicleEGO*	State 1
3	State 1	*decelerateVehicleEGO*	State 1
4	State 1	*runVehicleA*	State 1
5	State 1	*accelerateVehicleA*	State 1
6	State 1	*decelerateVehicleA*	State 1
7	State 1	$goLeftLaneVehicleA$	State 2
8	State 1	$goLeftLaneVehicleA$	State 3
9	State 1	$goLeftLaneVehicleA$	State 4
10	State 1	$goLeftLaneVehicleA$	State 5

From these scenarios, if we want to find the critical ones, and sort them according to their criticality, we use the equation 7.1 defined in Chapter 7 which allows us to calculate the scenario criticality.

In each component state *SituationXA*, where X is a zone number, $X = 1, \ldots, 6$ there is an action which leads to another state. Zone 3 and Zone 4 being the critical zones, all the state names including 3 or 4 (*Situation*3*A*, *Situation*4*A*) indicate the critical situations when *VA* rolls in Zone 3 and Zone 4, respectively. Thus, all the scenarios which include these critical situations, that is State 3 and State 4, are critical scenarios which may lead to accidents.

Table 8.5 presents the criticality of each scenario in Table 8.4. The criticality of both Scenario 8 and Scenario 9 is 0.5, since there is a critical state, respectively, State 3 and State 4, in

each of these scenarios.

TABLE 8.5 – Criticality of the scenarios

Scenario	1	2	3	4	5	6	7	8	9	10
Criticality	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.0

8.3 Case Study 2: Autonomous Vehicle in a Complex Context

We consider the situation "*One vehicle of type Other-Car riding with autonomous vehicle in the context of general highway infrastructure*" as the second case study to show how to build the PEPA model of system which includes some representative infrastructure elements and whether conditions. In this situation, we choose to consider the entity $toll$. However, note that as $toll$ is a sub-entity of *RoadPart*, like *tunnel*, *bridge* and *road_work*, we could have consider one of these sub-entities instead.

TABLE 8.6 – Components and actions of the PEPA model

Componengts	Actions
VehicleEGO	*runVehicleEGO*, *accelerateVehicleEGO*, *decelerateVehicleEGO*
VehicleA	*runVehicleA*, *accelerateVehicleA*, *decelerateVehicleA*, *goLeftLaneVehicleA*, *goRightLaneVehicleA*
EnLOff	*entranceOn*, *enterV*, *noEnterV*
ExLOff	*exitOn*, *exitV*, *noExitV*
SpeedLimit	*limitLower*, *limitGreater*
SignAOff	*signAoff*, *signAon*
SignABOff	*signABoff*, *signABon*
SignBOff	*signBoff*, *signBon*
SignCOff	*signCoff*, *signCon*
Visibility	*visibilityLower*, *visibilityGreater*
TollOff	*tollOn*, *tollOff*, *tollOut*
Situation	*runVehicleEGO*, *accelerateVehicleEGO*, *decelerateVehicleEGO*, *runVehicleA*, *accelerateVehicleA*, *decelerateVehicleA*, *goLeftLaneVehicleA*, *goRightLaneVehicleA*, *entranceOn*, *enterV*, *noEnterV*, *exitOn*, *exitV*, *noExitV*, *limitLower*, *limitGreater*, *signAoff*, *signAon*, *signABoff*, *signABon*, *signBoff*, *signBon*, *signBoff*, *signBon*, *signCoff*, *signCon*, *visibilityLower*, *visibilityGreater*, *tollOn*, *tollOff*, *tollOut*

This is a general situation which also includes the situation considered in Case Study 1. In the initial scene of this case study, the autonomous vehicle Ego rolls on the center lane of a separated lane road. One (1) vehicle of type *OtherCar* can appear in any zone around Ego. Zone 3 and Zone 4 are the critical zones, according to the minimum safety distance that must separate Ego from other vehicles.

In the PEPA model modelling this system, there are eleven (11) components: $VehicleEGO$, $VehicleA$, $EnLOff$, $ExLOff$, $SpeedLimit$, $SignAOff$, $SignABOff$, $SignBOff$, $SignCOff$, $Visibility$, $TollOff$ and $Situation$. These model the behaviour of the Ego car, any other vehicle, say VA, entrance lanes, exit lanes, Speed limit signs, signs of type A, AB, B and C, tolls and visibility, and the scene itself, respectively. The PEPA components and their actions are shown in Table 8.6. In the following, we present only component $Situation$ because all the other components are similar to those presented in Chapter 7.

We suppose that Ego is always on $Lane_2$ (Figure 8-1) with no lane change actions. The components and the actions of this PEPA model are listed in Table 8.6. All these components and actions have been presented in Chapter 7.

The initial situation $Situation$ indicates that Ego is rolling on the center lane and there is no other car. Because of the size of component $Situation$, here we present only its 13 first PEPA equations as follows:

$$\begin{aligned} Situation = {} & (runVehicleEGO, \top).Situation + (accelerateVehicleEGO, \top).Situation \\ & + (decelerateVehicleEGO, \top).Situation + (limitLower, \top).SituationDecelerate \\ & + (limitGreater, \top).SituationAccelerate + (signAon, w_1 \top).SituationDecelerate \\ & + (signAon, w_2 \top).SituationLeftRight + (signBon, w_3 \top).SituationDecelerate \\ & + (signBon, w_4 \top).SituationLeftRight + (entranceOn, \top).SituationEnLOn \\ & + (exitOn, \top).SituationExLOn + (visibilityLower, \top).SituationDecelerate \\ & + (visibilityGreater, \top).SituationAccelerate + (tollOn, \top).SituationDecelerate; \end{aligned}$$

In the initial state $Situation$, Ego has the choice between actions $runVehicleEGO$, $accelerateVehicleEGO$ and $decelerateVehicleEGO$. $Situation$ always remains in the initial state after these actions. Once one of the actions in $\{limitLower, signAon, signBon, visibilityLower, tollOn\}$ is performed, the state becomes $SituationDecelerate$. Executing action $limitGreater$ or $visibilityGreater$ leads to derivative state $SituationAccelerate$. There is also the possibility that once action $signAon$ or $signBon$ is performed, the component state becomes $SituationLeftRight$. Finally, the execution of action $entranceOn$ and $exitOn$ leads to state $SituationEnLOn$ and $SituationExLOn$, respectively.

$$\begin{aligned} SituationEnLOn = {} & (runVehicleEGO, \top).SituationEnLOn \\ & + (accelerateVehicleEGO, \top).SituationEnLOn \\ & + (decelerateVehicleEGO, \top).SituationEnLOn \\ & + (limitLower, \top).SituationEnLOnDecelerate + (noEnterV, \top).Situation \\ & + (enterV, \top).SituationEnterV + (visibilityLower, \top).SituationEnLOnDecelerate \\ & + (visibilityGreater, \top).SituationEnLOnAccelerate \\ & + (tollOn, \top).SituationEnLOnDecelerate; \end{aligned}$$

$$\begin{aligned} SituationEnterV = {} & (runVehicleEGO, \top).SituationEnterV \\ & + (accelerateVehicleEGO, \top).SituationEnterV \\ & + (decelerateVehicleEGO, \top).SituationEnterV \\ & + (goLeftLaneVehicleA, \top).Situation1A; \end{aligned}$$

Component states *SituationEnLOn* and *SituationEnterV* are similar to those of the running example (see Chapter 7). If there is a vehicle on the entrance lane, action *enterV* may be performed and this leads to the derivative state *SituationEnterV*. When *VA* rolls on the entrance lane and does action *goLeftLaneVehicleA*, it arrives on *Lane* 3 which leads to component state *Situation*1*A*, which is the state defined in Case Study 1.

$$\begin{aligned} SituationExLOn = {} & (runVehicleEGO, \top).SituationExLOn \\ & + (accelerateVehicleEGO, \top).SituationExLOn \\ & + (decelerateVehicleEGO, \top).SituationExLOn \\ & + (signCon, w_5\top).SituationExLOnDecelerate \\ & + (signCon, w_6\top).SituationExLOnLeftRight + (noExitV, \top).Situation \\ & + (visibilityLower, \top).SituationExLOnDecelerate \\ & + (visibilityGreater, \top).SituationExLOnAccelerate \\ & + (tollOn, \top).SituationExLOnDecelerate; \end{aligned}$$

Component state *SituationExLOn* is similar to *SituationEnLOn*. But in this state, there is no vehicle exiting using the exit lane because there is only *Ego* on the road. For the other states with exit lane, for example state *Situation*1*AExLOn*, if there is a vehicle exiting, the action *exitV* is performed which leads to component derivative state *Situation*1*A*.

$$\begin{aligned} SituationAccelerate = {} & (runVehicleEGO, \top).SituationAccelerate \\ & + (accelerateVehicleEGO, p_1\top).SituationAccelerate \\ & + (decelerateVehicleEGO, \top).SituationAccelerate \\ & + (entranceOn, \top).SituationEnLOnAccelerate \\ & + (exitOn, \top).SituationExLOnAccelerate + (signAoff, \top).Situation \\ & + (signBoff, \top).Situation + (tollOff, \top).Situation; \end{aligned}$$

$$\begin{aligned} SituationDecelerate = {} & (runVehicleEGO, \top).SituationDecelerate \\ & + (accelerateVehicleEGO, \top).SituationDecelerate \\ & + (decelerateVehicleEGO, p_2\top).SituationDecelerate \\ & + (entranceOn, \top).SituationEnLOnDecelerate \\ & + (exitOn, \top).SituationExLOnDecelerate + (signAoff, \top).Situation \\ & + (signBoff, \top).Situation + (tollOut, \top).SituationAccelerate; \end{aligned}$$

As component state *Situation*, derivative states *SituationAccelerate* and *SituationDecelerate* have the same actions and behaviours. The only difference is that the rates of actions *accelera-*

teVehicleEGO and *decelerateVehicleEGO* increase differently.

$$\begin{aligned} SituationLeftRight = {} & (runVehicleEGO, \top).SituationLeftRight \\ & + (accelerateVehicleEGO, \top).SituationLeftRight \\ & + (decelerateVehicleEGO, \top).SituationLeftRight \\ & + (entranceOn, \top).SituationEnLOnLeftRight \\ & + (exitOn, \top).SituationExLOnLeftRight \\ & + (signAoff, \top).Situation + (signBoff, \top).Situation; \end{aligned}$$

Component state *SituationLeftRight* is similar to the initial state *Situation* because there is no other car and Ego cannot change lane. For the other situations of this case study, the rates of the actions *goLeftLaneVehicleA* and *goRightLaneVehicleA* increase to represent the augmentation of the action occurrence.

The following component states *SituationEnLOnAccelerate*, *SituationEnLOnDecelerate*, *SituationEnLOnLeftRight*, *SituationExLOnAccelerate*, *SituationExLOnDecelerate* and *SituationExLOnLeftRight* are almost similar to states *SituationAccelerate*, *SituationDecelerate* and *SituationLeftRight*. The difference is that there is an entrance lane in the first ones and an exit lane in the last ones.

$$\begin{aligned} SituationEnLOnAccelerate = {} & (runVehicleEGO, \top).SituationEnLOnAccelerate \\ & + (accelerateVehicleEGO, p_3\top).SituationEnLOnAccelerate \\ & + (decelerateVehicleEGO, \top).SituationEnLOnAccelerate \\ & + (signABoff, \top).SituationEnLOn + (noEnterV, \top).Situation \\ & + (enterV, \top).SituationEnterV + (tollOff, \top).SituationEnLOn; \end{aligned}$$

$$\begin{aligned} SituationEnLOnDecelerate = {} & (runVehicleEGO, \top).SituationEnLOnDecelerate \\ & + (accelerateVehicleEGO, \top).SituationEnLOnDecelerate \\ & + (decelerateVehicleEGO, p4\top).SituationEnLOnDecelerate \\ & + (signABoff, \top).SituationEnLOn + (noEnterV, \top).Situation \\ & + (enterV, \top).SituationEnterV + (tollOut, \top).SituationEnLOnAccelerate; \end{aligned}$$

$$\begin{aligned} SituationEnLOnLeftRight = {} & (runVehicleEGO, \top).SituationEnLOnLeftRight \\ & + (accelerateVehicleEGO, \top).SituationEnLOnLeftRight \\ & + (decelerateVehicleEGO, \top).SituationEnLOnLeftRight \\ & + (signABoff, \top).SituationEnLOn + (noEnterV, \top).Situation \\ & + (enterV, \top).SituationEnterV; \end{aligned}$$

$$
\begin{aligned}
SituationExLOnAccelerate = {} & (runVehicleEGO, \top).SituationExLOnAccelerate \\
& + (accelerateVehicleEGO, p5\top).SituationExLOnAccelerate \\
& + (decelerateVehicleEGO, \top).SituationExLOnAccelerate \\
& + (signCoff, \top).SituationExLOn + (noExitV, \top).Situation \\
& + (tollOff, \top).SituationExLOn;
\end{aligned}
$$

$$
\begin{aligned}
SituationExLOnDecelerate = {} & (runVehicleEGO, \top).SituationExLOnDecelerate \\
& + (accelerateVehicleEGO, \top).SituationExLOnDecelerate \\
& + (decelerateVehicleEGO, p6\top).SituationExLOnDecelerate \\
& + (signCoff, \top).SituationExLOn + (noExitV, \top).Situation \\
& + (tollOut, \top).SituationExLOnAccelerate;
\end{aligned}
$$

$$
\begin{aligned}
SituationExLOnLeftRight = {} & (runVehicleEGO, \top).SituationExLOnLeftRight \\
& + (accelerateVehicleEGO, \top).SituationExLOnLeftRight \\
& + (decelerateVehicleEGO, \top).SituationExLOnLeftRight \\
& + (signCoff, \top).SituationExLOn + (noExitV, \top).Situation;
\end{aligned}
$$

The complete PEPA model equation is the following:

$$
\begin{aligned}
Scenario \stackrel{\text{def}}{=} Situation \underset{L_3}{\bowtie} (& VehicleEGO \parallel VehicleA \parallel EnLOff \parallel ExLOff \parallel SpeedLimit \\
& \parallel SignABOff \parallel SignAOff \parallel SignBOff \parallel SignCOff \parallel Visibility \parallel TollOff)
\end{aligned}
$$

where L_3 is the actions set on which components *Situation* must synchronise with the other components of the model individually. It is defined as:

$$
\begin{aligned}
L_3 = \{ & runVehicleEGO, accelerateVehicleEGO, decelerateVehicleEGO, goLeftLaneVehicleA, \\
& goRightLaneVehicleA, runVehicleA, accelerateVehicleA, decelerateVehicleA, \\
& entranceOn, enterV, noEnterV, exitOn, exitV, noExitV, limitLower, limitGreater, \\
& signABoff, signABon, signAoff, signAon, signBoff, signBon, signCoff, signCon, \\
& visibilityLower, visibilityGreater, tollOn, tollOff, tollOut\}.
\end{aligned}
$$

8.3.1 Numerical Results

This PEPA model has 1935 states, and the probability of being in each state is provided in Figure 8-6. These results are based on a set of random values for the rates and the weights in the model. Different values lead to different test cases and results. The probability is mainly concentrated in the first hundred of states, and gradually decreases. Most of the states with high

probability have no or only one sign.

State 14 (*SituationExLOnAccelerate*, *VehicleEGO*, *VehicleA*, $EnLOff$, *ExLOn*, *SpeedLimit*, *SignABOff*, *SignAOff*, *SignBOff*, *SignCOff*, *Visibility*, *TollOff*) has the highest probability (0.028133829166193326). This state refers to the situation where there is no sign and no toll. While State 1918 (*Situation2AEnLOn*, *VehicleEGO*, *VehicleA*, *EnLOn*, $ExLOff$, *SpeedLimit*, *SignABOff*, *SignAOn*, *SignBOn*, *SignCOn*, *Visibility*, *TollOut*) has the smallest probability (4,54531810598917E-07). This state refers to the situation where there are three signs appearing at the exit of toll.

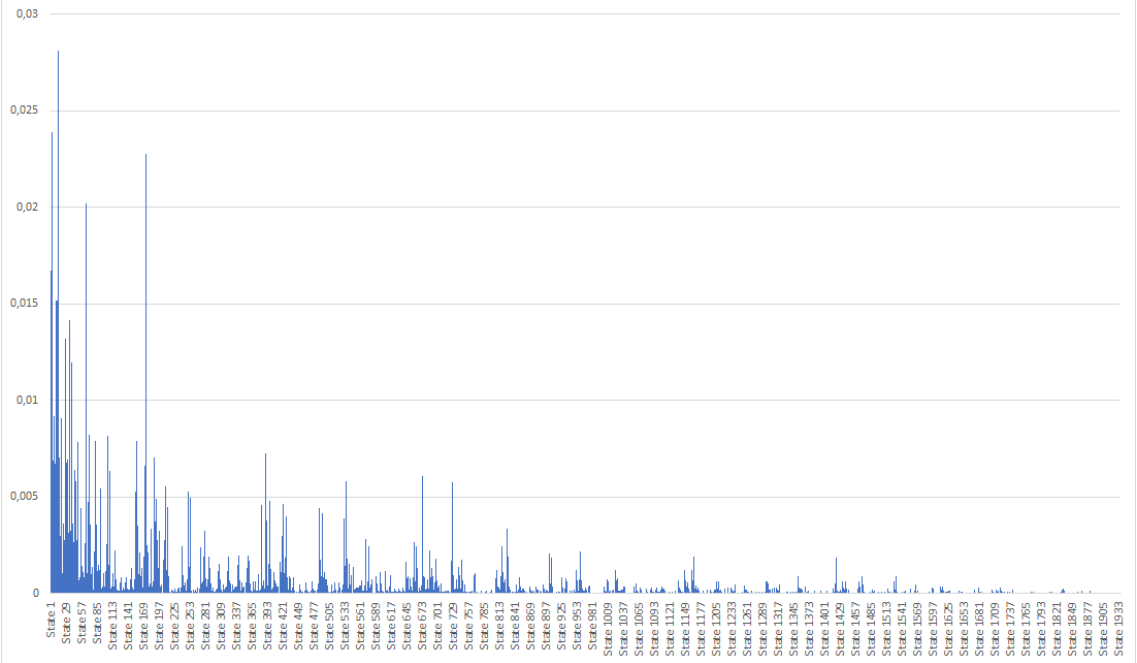

FIGURE 8-6 – Steady-state probability distribution

From the initial system state State 1 (*Situation*, *VehicleEGO*, *VehicleA*, $EnLOff$, $ExLOff$, *SpeedLimit*, *SignABOff*, *SignAOff*, *SignBOff*, *SignCOff*, *Visibility*, *TollOff*), we generate all possible scenarios with a length no greater than 5 scenes per scenario with our test case generation algorithm. In this case, we obtain 11835 eligible scenarios, in which there are only two critical scenarios (Figure 8-7). They are:

- **Scenario 11822**: State 1, *entranceOn*, State 8, *enterV*, State 23, *goLeftLaneVehicleA*, State 33, *goLeftLaneVehicleA*, State 52;
- **Scenario 11823**: State 1, *entranceOn*, State 8, *enterV*, State 23, *goLeftLaneVehicleA*, State 33, *goLeftLaneVehicleA*, State 53.

These scenarios are critical because they include one of the critical states State 52 and State 53. The reason why the critical scenarios are so few is that they are derived from the initial situation where there is only *Ego* on highway. After 4 states of scenario, *VA* has not frequently moved around *Ego*. The criticality of both scenarios is 0.2.

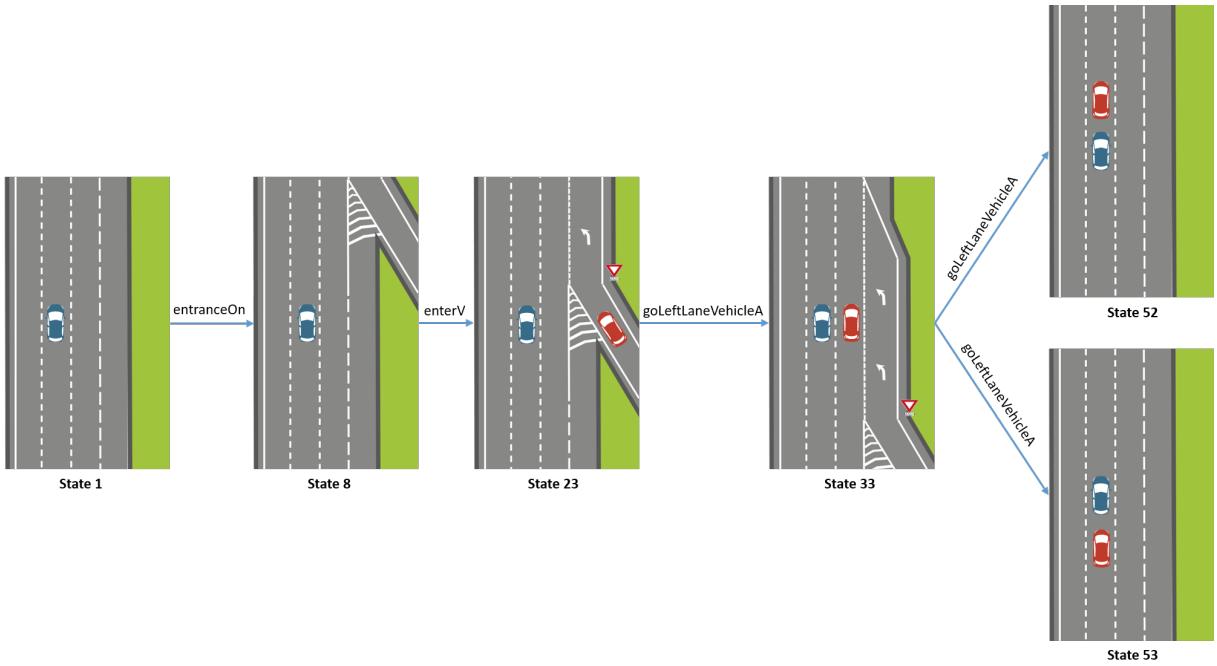

FIGURE 8-7 – Sequences of critical scenarios' states

We also calculate the probabilities of the system being in each scenario. The most probable scenario is: State 1, *runVehicleEGO*, State 1. Its occurrence probability is 0.0002791974785486466.

8.4 Conclusion

In this chapter, we use two case studies to show how we generate the scenarios and especially the critical ones. Because of the size of the PEPA models, specially the second one, we have used the algorithm presented in Chapter 7 to generate the PEPA models. We also used the algorithm presented in Chapter 7 to generate the scenarios.

So far, we can generate models of up to 6 vehicles of type *Othercar*. However, the model generated in this case consists of more than 3.6 million equations. This model is too large to be analyzed using the Eclipse tool.

At the same time we also get the criticality of the scenarios and the probability of their occurrence. These data can help us to choose more representative and influential scenarios to test and validate the decisions of ADSs.

These results are based on the properties values chosen. Other values will clearly lead to other results. Experts can give their own values to generate ADSs test cases using our methodology.

Chapitre 9

Conclusions

The autonomous vehicles industrial process showed significant acceleration state. The major companies are accelerating research on autonomous vehicles. In 2020, Google's Waymo is expected to achieve 20,000 autonomous vehicles and provide up to 1 million rides a day [Madrigal, 2018], while Baidu is expected to achieve mass production of autonomous vehicles [Millward, 2015]. Weilai will launch Level 4 autonomous vehicles with an L4 self-driving development kit, including cameras, millimeter-wave radar, lidar, GPS and other hardwares in 2022 [domeet kevinBobo, 2019]. IHS Automotive's latest forecast is that annual sales of autonomous vehicles will reach nearly 600,000 by 2025, and that number will reach 21 millions by 2035 [Markit, 2016].

While autonomous driving systems continue to improve, autonomous vehicles equipped with high level ADSs are also being produced. From level 4, the automated systems perform the driving task and monitor the driving environment. The human driver does not need to regain control. However, the driver does not hold the steering wheel and does not monitor the surrounding environment, which is contrary to existing laws. Before pushing for changes in the law, we must first determine the safety of the developed ADSs. This is also the original intention of the SVA project, in which this thesis work lies.

The SVA project aims to respond by digital simulation to the challenge posed by the demonstration of safety and harmlessness of the functions on board autonomous vehicles. One of the major challenges is to be able to qualify the safety of autonomous vehicle decision algorithms. In this thesis, we proposed a methodology to generate all possible situations that autonomous vehicle will meet in the context of the highway. These situations are presented as test cases for testing and validating the decisions of ADSs.

In order to reduce the conceptual and the terminological confusion in our SVA project, firstly, we provide a clear definition of a test case as one or several scenarios describing the same situation applied to test one or several functions of ADSs. Then we defined three ontologies to

specify, on the one hand the environment in which evolves the autonomous vehicle, and on the other hand the vehicle lights and its actions. In this thesis, we concentrate on the driving environment of the highway since the TJC system, on which SVA project works, is mainly used in highways.

We use the first-order logic to represent the different relationships that we have defined in the context of these ontologies. We defined three kinds of relationships between the entities of the defined concepts: the relationships between the highway entities, the relationships between the vehicle entities, and the relationships between the highway and vehicle entities. We re-described the test cases with logical equations instead of natural language, thereby achieving primary formalization of a test case.

The test case generation methodology we developed consists of three layers: basic layer, interaction layer and generation layer. We introduced the construction of the highway model using the formal modelling language PEPA. We also investigated the generation of all scenarios, specially the critical ones. Solving the PEPA model allows us to compute the probability to be in each scene of a scenario, and thus in each scenario. Finally, we have applied our methodology on two case studies.

Our methodology gives a new idea of how to generate automatically the test cases and to identify the critical ones. We also proposed a method to calculate the criticality of each test case. We can comprehensively evaluate the importance of a test case by its criticality and its probability of occurrence.

In the future, more concepts can be added to expand our ontologies. We can make them suitable for all types of road and area such as the municipal roads and the parking areas. Thus our ontologies may support other uses in other contexts such as urban design.

Currently, the values of the rates of the actions other than $Accelerate$ and $Decelerate$ are set arbitrary because they are not available. We also do not have data on the impact of the different weather conditions on the autonomous vehicles behaviour. We could only use the property of visibility instead. In the future, if more actual test data are available or assisted by experts in the field, all data can be injected in our program.

In order to reduce the combinatorial explosion of the number of scenes, not all concepts in the ontology are modelled with PEPA in the generation layer. For example, we have defined the concept $Shoulder$ in the highway ontology because it is part of the highway, but it is not very important in our model. In the future, we may add all the components if the computing power improves.

So far, we can generate PEPA models for up to six (6) vehicles of type $Othercar$ around the autonomous vehicle. However, the generated model consists of more than 3.6 million equations, which makes it too large to be analyzed using the Eclipse PEPA tool. The PEPA model needs to be optimized. In the future, we can apply the aggregation technique developed for the

PEPA models [Hillston, 1994] [Pourranjbar and Hillston, 2013]. This technique allows reducing the size of a PEPA model if components exhibit the same behaviour, which is the case of the six vehicles of type *Othercar* in our model. It has been proved that the aggregated PEPA model is equivalent to the original one. Thus the state space of the underlying continuous-time Markov chain can be reduced, so that the model can include more components and/or solved in reasonable time.

Résumé

Depuis les années 1970, la recherche sur le véhicule autonome est devenue une tendance dans l'industrie. Dernièrement et après des années d'exploration, un certain progrès a été réalisé. En 2018, Audi a étendu le système d'information des feux de circulation V2I (Vehicle-to-Infrastructure) à Washington [Krok, 2018]. Nissan prévoit de poursuivre sa collaboration avec la NASA pour adapter la technologie de la NASA pour une utilisation sur sa plate-forme de mobilité autonome transparente [Bartosiak, 2018]. Non seulement l'industrie automobile traditionnelle est dédiée à ce domaine de recherche, mais d'autres sociétés, telles que Google et Intel, ont également participé au développement du véhicule autonome. Waymo, initialement projet de voiture autonome de Google, a annulé la conception du volant et des pédales [Gain, 2017], ce qui bouleverse complètement la conception des voitures traditionnelles.

Les véhicules autonomes s'appuient principalement sur des systèmes intelligents pour atteindre l'objectif de la conduite autonome. Ils combinent une variété de capteurs pour percevoir leur environnement, tels que des caméras, des radars et des lidars. En effet, ils doivent évoluer dans un environnement imprévisible et un large contexte d'exécution dynamique, avec des interactions fortes. Les algorithmes de perception des ADSs (Automated Driving Systems) fournissent des observations sur les éléments environnementaux à partir des données fournies par les capteurs, tandis que les algorithmes de décision génèrent les actions à mettre en œuvre par ces véhicules.

Selon les niveaux d'automatisation des véhicules de SAE (Society of Automotive Engineers), à partir du niveau 4, les systèmes automatisés exécutent la tâche de conduite et surveillent l'environnement de conduite. Le conducteur humain n'a pas besoin de reprendre le contrôle. Cependant, le conducteur ne tient pas le volant et ne surveille pas l'environnement autour, ce qui est contraire aux lois en vigueur. Avant de pousser pour des changements dans la loi, nous devons d'abord déterminer le niveau de sécurité des ADS développés.

Des ADSs sont développés pour exécuter les principaux aspects de la tâche de conduite dynamique [SAE, 2014]. Les technologies développées devraient prévenir les accidents, réduire les émissions, transporter les personnes à mobilité réduite et réduire le stress lié à la conduite. Par exemple, le Département américain des transports (USDOT) estime que les véhicules automatisés peuvent réduire les décès et les blessures liés aux accidents, améliorer l'accès aux transports et réduire la congestion routière et les émissions des véhicules [NHTSA, 2016]. Mais les véhicules autonomes soulèvent également de nouveaux problèmes de sécurité qui sont dûs à la nature émergente de la technologie.

En effet, les ADSs sont des systèmes critiques dont les pannes peuvent avoir des conséquences catastrophiques. Des erreurs techniques systématiques des ADS, par exemple des bugs et des failles dans les capteurs ou des données manquantes, pourraient être à l'origine de sé-

rieux dangers, semblables à des erreurs humaines. Des décès dûs à des ADSs immatures ont déjà été signalés et sont considérés comme en augmentation [BBC, 2018] [Everington, 2020]. La certification de sécurité et de fiabilité est une tâche qui reste donc à résoudre.

Comme tout autre système pouvant générer des événements potentiellement à risque, le véhicule autonome doit être conçu pour assurer la sécurité de ses occupants et des autres usagers de la route. La fiabilité de l'architecture et la logique comportementale des ADS doivent être testées, vérifiées et validées avant que les véhicules autonomes équipés de ces systèmes soient sur la route. Cela souligne la nécessité d'approches et d'outils améliorés pour évaluer la sécurité des mouvements des véhicules autonomes dans des environnements dynamiques et incertains.

Afin de garantir la fonctionnalité et la sécurité du système de conduite autonome, il est nécessaire de valider les décisions des algorithmes pour toutes les situations qui seront rencontrées par le véhicule. La complexité de la démonstration de la sécurité d'un véhicule autonome est liée au grand nombre de ces situations, à leur incertitude et aux technologies embarquées. Cela rend la validation par des tests en utilisation réelle extrêmement coûteuse, voire impossible dans certains cas. Afin de s'assurer que les exigences de sécurité ont été respectées, la validation des ADSs du véhicule autonome par simulation numérique est nécessaire.

Le projet SVA (Simulation de la sécurité des véhicules autonomes) [SVA, 2016], dans lequel s'inscrit ce travail de thèse, vise à répondre par la simulation numérique au défi posé par la démonstration de la sûreté de fonctionnement et de l'innocuité des fonctions à bord des véhicules autonomes. Lancé en 2015 pour une durée de quatre ans au sein de l'IRT SystemX, Paris-Saclay, France, le projet SVA vise à aborder la problématique de la validation des véhicules autonomes par la simulation numérique, en développant des méthodes et des outils d'aide à la conception et à la validation. Les modèles des composants du véhicule et leur environnement devraient être spécifiés, adaptés ou développés afin de simuler le comportement du véhicule en cas de défaillance de l'un de ses composants et l'impact sur son fonctionnement dû à des perturbations externes. Le projet SVA applique les méthodes développées à la fonction d'autonomie TJC (Traffic Jam Chauffeur), qui permet de contrôler le véhicule dans une situation d'embouteillage, à une vitesse maximale de 70 km/h et sur une chaussée séparée.

L'objectif principal de ces travaux de thèse est de développer une approche complète permettant, d'une part la conceptualisation et la caractérisation des contextes d'exécution du véhicule autonome, et d'autre part, la modélisation et génération de cas de test. Ces cas de test sont générés pour décrire les situations de conduite. Nous nous intéressons à une méthode de génération automatique qui permet de générer des cas de test ayant un impact sur les performances et la fiabilité du véhicule. Générer tous les cas de test possibles est un défi. Nous nous concentrons sur les cas de test dans le contexte de l'autoroute qui est de type chaussée séparée. De plus, par rapport à d'autres types de routes, il existe des spécifications uniformes pour les autoroutes.

La sûreté de fonctionnement est un concept générique qui mesure la qualité du service

fourni par un système. Cette confiance justifiée est obtenue grâce à une analyse qualitative et quantitative des différentes propriétés du service rendu par le système. Les ADSs étant testés, vérifiés et validés par simulation numérique, l'environnement de conduite du véhicule autonome doit être modélisé afin de pouvoir générer toutes les situations possibles que le véhicule peut rencontrer dans un contexte d'exécution dynamique. Ces situations sont générées sous forme de cas de test différents permettant de vérifier les fonctions et les informations nécessaires pour exécuter ces fonctions, ainsi que les décisions des ADSs.

Les méthodes formelles sont un type particulier de techniques et d'outils mathématiquement rigoureux pour la spécification, la conception et la vérification des systèmes logiciels et matériels. Les spécifications utilisées dans les méthodes formelles sont des énoncés bien formés dans une logique mathématique. Chaque étape découle d'une règle d'inférence et peut donc être vérifiée par un processus mécanique [Alagar and Periyasamy, 2011]. Ces méthodes ont été initialement développées pour spécifier et vérifier le comportement correct des systèmes logiciels et matériels et ont été appliquées dans de nombreux domaines de développement de systèmes, et de nombreux accomplissements ont été réalisés [Almeida et al., 2011].

Il existe une variété de méthodes formelles de modélisation de système. Ces méthodes peuvent être classées en deux grandes catégories : les méthodes dédiées aux systèmes séries et celles dédiées aux systèmes concurrents.

Dans un système en série, les tâches sont traitées une par une, une tâche doit être terminée avant le début d'une autre. Ces systèmes peuvent être modélisés en utilisant des techniques telles que le langage Z [Meyer and Baudoin, 1978], la méthode de développement de Vienne (VDM) [Bjørner and Jones, 1978] et la méthode B [Abrial, 1988].

Les systèmes concurrents sont beaucoup plus complexes que les systèmes en série. Un système concurrent permet à toutes les tâches de progresser pour supporter plus d'une tâche [GALVIN and GAGNE, 2005]. Il consiste en un ensemble de processus communiquant via des structures de données partagées ou des objets [Παλαιοδήμος, 2018]. Il existe plusieurs types de techniques de modélisation pour les systèmes concurrents, tels que les processus de Markov [Kemeny and Snell, 1976], les réseaux de Petri [Petri, 1962] et les algèbres de processus [Bergstra et al., 2001].

Bien que le véhicule autonome appartienne à la catégorie des systèmes concurrents, à notre connaissance, il existe peu de travaux dédiés aux approches formelles appliquées au domaine des véhicules autonomes.

Dans ces travaux de thèse nous nous intéressons à l'algèbre des processus, en particulier à PEPA (Performance Evaluation Process Algebra) [Hillston, 1994]. PEPA est une algèbre de processus stochastique conçue pour modéliser les systèmes informatiques et de communication. Ce langage formel a été développé pour étudier comment les caractéristiques de composition d'une algèbre de processus pourraient avoir un impact sur la pratique de la modélisation des performances. PEPA peut modéliser des éléments du système qui se comportent et évoluent

individuellement ou en coopération les uns avec les autres. Cette technique de modélisation est suffisamment puissante pour modéliser tous les comportements d'un système.

Un modèle PEPA est construit en identifiant les composantes du système effectuant des activités. Cette technique est caractérisée par un petit ensemble d'opérateurs. Ces opérateurs et leur syntaxe sont définis comme suit :

$$S \stackrel{\text{def}}{=} (\alpha, r).P \mid P + Q \mid P \underset{L}{\bowtie} Q \mid P/L \mid A$$

Prefix:	$S \stackrel{\text{def}}{=} (\alpha, r).P$,	la composante S effectue l'activité (α, r) qui a le type d'action α et une durée qui est exponentiellement distribuée avec le paramètre r avant de se comporter comme P.
Choice:	$S \stackrel{\text{def}}{=} P + Q$,	S peut se comporter soit comme la composante P, soit comme la composante Q.
Cooperation:	$S \stackrel{\text{def}}{=} P \underset{L}{\bowtie} Q$,	S est le résultat de la coopération ou de la synchronisation entre les composantes P et Q. Les activités partagées dans l'ensemble de coopération L déterminent les interactions entre les composantes P et Q, en remplaçant les activités individuelles des composantes individuelles P et Q par un taux reflétant le taux du participant le plus lent.
Hiding:	$S \stackrel{\text{def}}{=} P/L$,	le système se comporte comme la composante P, sauf que toute activité dans l'ensemble L est masquée. Son type n'est pas observé à la fin. Il apparaît comme le type inconnu τ et peut être considéré comme un retard interne de la composante.
Constant:	$S \stackrel{\text{def}}{=} A$	cet opérateur affecte à S le comportement de la composante A. En général, on l'utilise pour attribuer des noms aux composantes.

Un processus de Markov en temps continu peut être derivé des composantes du modèle PEPA et de leurs interactions. Les outils existants (Eclipse PEPA [Hillston and Gilmore, 2014]) permettent de générer ce processus Markov sous-jacent.

Méthodologie de génération de cas de test

Les constructeurs ont besoin d'une stratégie de génération complète pour garantir l'exhaustivité des situations auxquelles le véhicule autonome sera confronté [Kone et al., 2019]. Cependant, comme le véhicule autonome repose sur la coopération de l'intelligence artificielle, des capteurs tels que les radars, les caméras et les lidars, et le GPS pour améliorer la sécurité routière et l'efficacité du trafic, avec le développement des technologies, ces capteurs fournissent

de plus en plus d'éléments d'environnement de conduite aux ADSs. Ces éléments d'infrastructure combinés aux conditions météorologiques avec leurs propres propriétés peuvent conduire à l'explosion combinatoire du nombre des situations rencontrées par le véhicule, et par conséquent des scènes constituant les cas de test. Ainsi, générer tous les cas de test possibles devient presque impossible.

Dans cette thèse, nous proposons donc une approche de génération de cas de test qui se concentre sur les cas de test les plus représentatifs pour tester et valider les ADSs. Cette approche basée sur un modèle formel nous permet de modéliser formellement ces cas de test et d'identifier les plus critiques.

Avant de présenter les détails de notre méthodologie de génération de cas de test, nous devons d'abord clarifier la définition d'un cas de test. Nous définissons un **cas de test** comme un environnement de conduite spécifique pour le véhicule autonome. Il consiste en un scénario décrivant une situation spécifique pour laquelle des valeurs sont attribuées aux propriétés de chaque élément du scénario. Le choix de ces valeurs dépend de l'objectif de chaque cas de test. Un **scénario** décrit le développement temporel entre plusieurs scènes dans une séquence de scènes. Il est associé aux actions de tous les éléments de la séquence de scènes. Une **scène** est un instantané de l'environnement du véhicule, y compris les éléments statiques et mobiles, et les relations entre ces éléments (Figure -1).

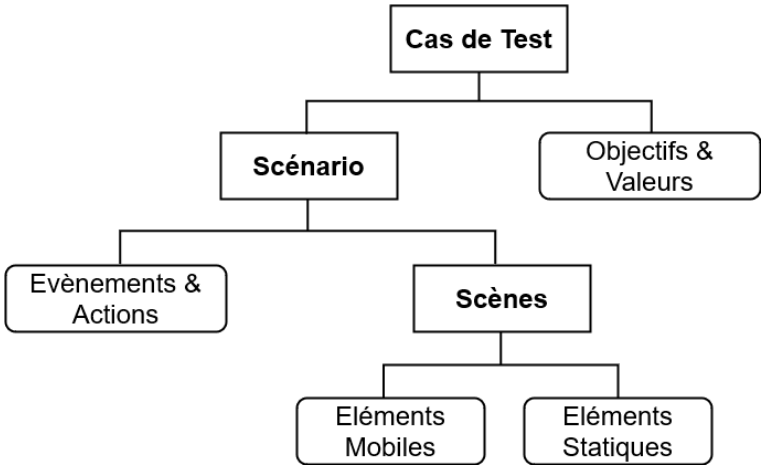

FIGURE -1 – Structure du cas de test.

Pour générer les cas de test, nous devons faire face à un premier défi - la confusion conceptuelle et terminologique dans le projet SVA. Différentes terminologies sont utilisées par les partenaires du projet avec des formalismes et des besoins différents. De plus, certains mots utilisés dans la même terminologie sont ambigus, certains sont redondants et ont donc la même signification, tandis qu'un même mot peut avoir des significations différentes. Cela entraîne un manque de compréhension commune entre les partenaires du projet et des difficultés de coopération limitant ainsi le potentiel de réutilisation et de partage de leurs travaux. Ainsi, nous avons besoin d'un vocabulaire commun pour toutes les parties prenantes qui ont besoin de partager des informations dans le domaine des véhicules autonomes.

Pour faire face à ce premier défi, nous devons identifier les concepts clés et les relations possibles entre les éléments impliqués dans les différents contextes d'exécution, pour donner des définitions claires de ces éléments. Ainsi, afin de conceptualiser et caractériser l'environnement de conduite pour la construction de cas de test, nous définissons trois ontologies : une *ontologie de l'autoroute* et une *ontologie de la météo* pour spécifier l'environnement dans lequel évolue le véhicule autonome, et une *ontologie du véhicule* qui se compose des feux du véhicule et des actions de contrôle. Chaque concept de ces ontologies est défini en termes d'entité, de sous-entités et de propriétés. Cette première étape constitue la **couche de base** de notre méthodologie qui inclut les éléments statiques et mobiles de l'environnement de conduite (Figure -2). Les ***éléments statiques*** font référence à tous les éléments géospatiaux fixes qui incluent l'infrastructure de l'autoroute et les conditions météorologiques. Les ***éléments mobiles*** sont des éléments qui ont la capacité de se déplacer. Ils incluent le véhicule autonome et les autres trafics. Cela permet de couvrir des éléments majeurs de l'infrastructure et des véhicules.

Les ***actions*** considérées sont celles effectuées par le véhicule autonome alors que les ***événements*** sont les actions réalisées par les éléments de l'environnement comme les autres véhicules du trafic, et qui sont considérées comme des événements du point de vue du véhicule autonome.

Notre approche permet de modéliser non seulement la circulation des véhicules, mais aussi l'apparition et la disparition des infrastructures au fur et à mesure que le véhicule autonome avance. Les ***valeurs*** de ces éléments sont les valeurs des propriétés de leurs concepts dans l'ontologie correspondante, qui déterminent leurs caractéristiques intrinsèques.

Parce que nous devons considérer toutes les relations entre différents concepts, nous considérons toutes les entités et les valeurs des propriétés correspondantes pour construire les *relations entre les entités autoroute*, les *relations entre les entités véhicule* et les *relations entre les entités autoroute et véhicule*. Ces relations sont exprimées en utilisant les equations de la logique du premier ordre, qui nous permet d'exprimer de manière simple les relations entre les différents éléments de notre système. Et cela constitue la deuxième couche de notre méthodologie appelée **couche d'interaction**.

Clairement la météo a un impact sur la conduite, mais nous ne disposons d'aucune information par le type d'impact dont il s'agit, surtout lorsque les conditions météorologiques interagissent les unes avec les autres entraînant des effets complexes. Dans ce travail, nous donnons la possibilité au véhicule autonome d'accélérer et de décélérer pour s'adapter aux différentes conditions météorologiques. Généralement, lorsque la visibilité est élevée, le véhicule se déplace plus vite et lorsque la visibilité est faible, le véhicule a tendance à ralentir. Par conséquent, dans ce travail, nous utilisons la propriété de visibilité pour modéliser l'impact de la météo sur le véhicule autonome. Si le testeur est un expert dans ce domaine ou a des exigences de test plus précises, d'autres concepts peuvent être ajoutés si nécessaire.

Afin de générer les cas de test, nous nous référons à la définition d'un cas de test. Ainsi, la génération des cas de test comporte 3 étapes. Tout d'abord nous devons générer la scène

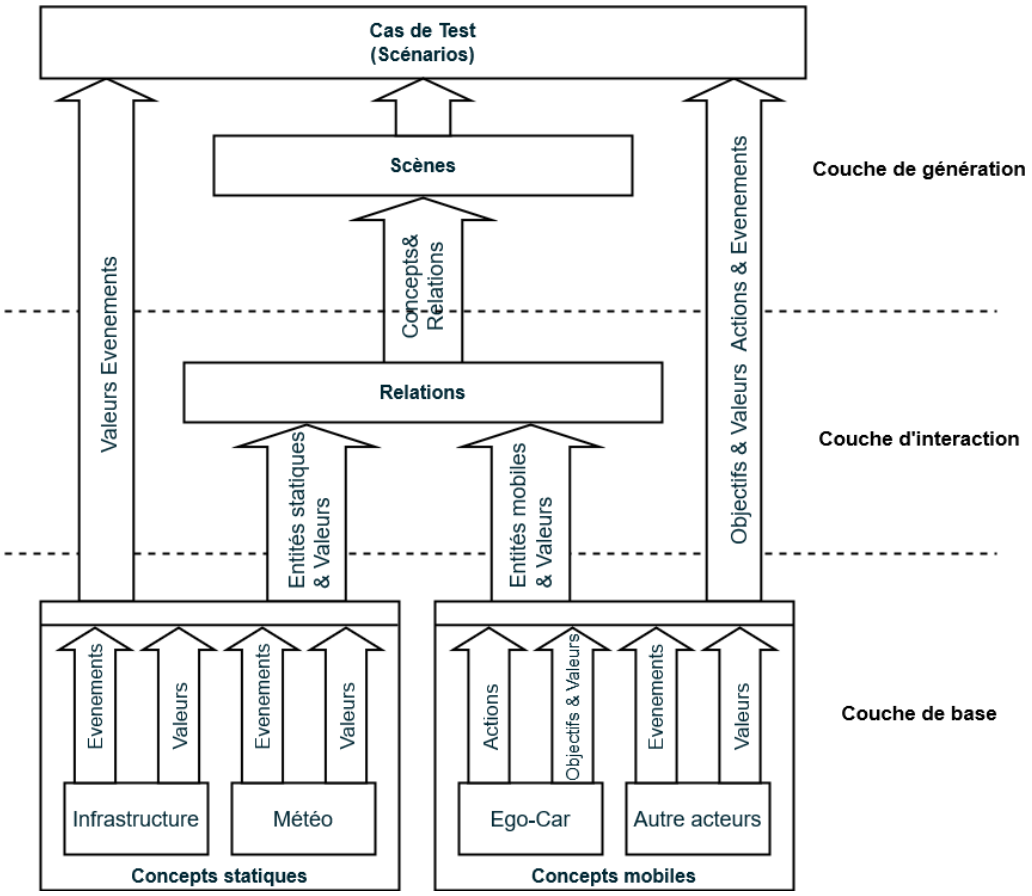

FIGURE -2 – Méthodologie de génération de cas de test.

en fonction des concepts et des relations définis. Puis, les scénarios sont associés aux actions effectuées par le véhicule autonome et aux événements réalisés par les éléments de l'environnement dans la séquence des scènes. Enfin, un scénario devient un cas de test lorsque des valeurs sont affectées aux propriétés de chaque élément du scénario. Les étapes de génération des cas de test constituent la troisième et dernière couche de notre méthodologie appelée **couche de génération**.

A ce stade, nous formalisons les cas de test en utilisant PEPA (Performance Evaluation Process Algebra) [Hillston, 1994]. Dans notre modèle PEPA général, nous considérons la portion de chaussée d'autoroute autour du véhicule autonome, Ego (véhicule bleu sur la figure -3). En fonction de la vitesse de Ego et de la vitesse du véhicule éventuel suivant et juste avant Ego, on peut définir un ***zone critique*** (zone jaune sur la figure -3) dans la voie centrale. Cette zone est délimitée en considérant la distance minimale de sécurité qui doit séparer Ego des autres véhicules : celui juste avant et celui juste après. Les distances minimales de sécurité sont les distances parcourues par le véhicule pendant un délai d'au moins deux secondes, calculées à partir de la vitesse du véhicule.

De plus, nous séparons la portion de la chaussée en six (6) zones comme le montre la figure

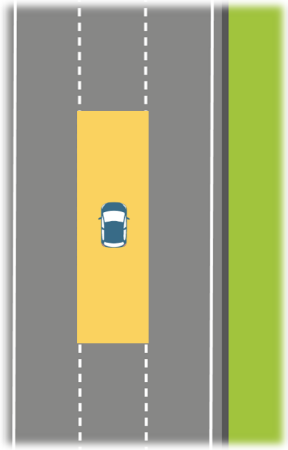

FIGURE -3 – Zone critique autour de *Ego*.

-4. Nous numérotons ces zones de un à six. La zone 1 indique la voie de gauche. Les zones 2 et 5 indiquent les zones non critiques devant et derrière *Ego*. La zone 3 indique la zone critique devant *Ego* tandis que la zone 4 indique la zone critique derrière lui. La zone 6 indique la voie de droite. Les zones 1 et 6 sont des zones non critiques pour *Ego*.

Un autre véhicule peut être dans n'importe quelle zone autour de *Ego*. Nous modélisons les mouvements des autres véhicules entre les zones avec le graphe de la figure -5. Ce graphique montre toutes les transitions possibles entre ces zones. Par exemple, un véhicule peut se déplacer entre la zone 1 et la zone k, et entre la zone k et la zone 6, $k = 2,3,4,5$. Il peut également se déplacer entre la zone 2 et la zone 3, et entre la zone 4 et la zone 5. Tous les véhicules dans les zones non critiques peuvent pénétrer dans les zones critiques et vice versa.

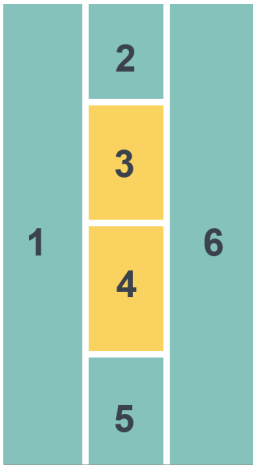

FIGURE -4 – Zones dans la scène.

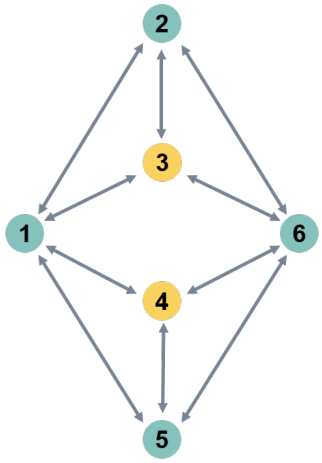

FIGURE -5 – Mouvements entre zones.

Compte tenu des concepts que nous avons définis dans nos ontologies, nous construisons un modèle PEPA qui se compose de dix-neuf (19) composantes. Toutes les équations qui caractérisent les comportements de ces composantes sont présentées dans le Chapitre 7.

En raison du grand nombre d'éléments de l'infrastructure autoroutière, du nombre de véhicules possibles et des conditions météorologiques, l'écriture des composantes PEPA est fastidieuse. Aussi, nous proposons d'abord un algorithme pour générer automatiquement des modèles PEPA. Par ailleurs, comme nous nous intéressons à la génération automatique des cas de test, nous proposons une méthode qui permet de classer ces cas de test en fonction de leur impact sur les performances et la fiabilité du véhicule. Nous proposons également un algorithme qui permet de générer automatiquement les cas de test à partir de n'importe quelle situation initiale et avec n'importe quel nombre de scènes. Cela aussi nous permet d'identifier les cas de test critiques. Enfin, nous proposons une méthode pour calculer la criticité de chaque cas de test afin d'évaluer de manière exhaustive son importance.

Nous avons appliqué notre méthodologie sur deux études de cas. Le premier cas " *Véhicule autonome dans un contexte simple*" ne comporte qu'un seul véhicule circulant autour du véhicule autonome. Nous avons choisi ce petit modèle pour montrer les étapes complètes permettant la génération des scénarios et notamment les plus critiques. La seconde étude de cas " *Véhicule autonome dans un contexte complexe*" est un cas plus complexe où les panneaux de signalisation, les voies d'entrée, les voies de sortie, les péages et la visibilité sont pris en compte. Nous l'avons choisi pour montrer comment nous générons le modèle PEPA avec différents éléments de l'infrastructure autoroutière et la visibilité comme impact météorologique sur le véhicule autonome.

Conclusions et Perspectives

La méthodologie de génération de cas de test que nous avons développée se compose de trois couches: une couche de base, une couche d'interaction et une couche de génération. Nous avons également étudié la génération de tous les scénarios, en particulier les plus critiques. La résolution du modèle PEPA nous permet de calculer la probabilité d'être dans chaque scène d'un scénario, et donc dans chaque scénario. Notre méthodologie donne une nouvelle idée de la façon de générer automatiquement les cas de test et d'identifier les cas critiques. Nous avons également proposé une méthode pour calculer la criticité de chaque cas de test. Nous pouvons évaluer globalement l'importance d'un cas de test par sa criticité et sa probabilité d'occurrence.

À l'avenir, d'autres concepts pourront être ajoutés pour étendre nos ontologies. Nous pouvons les adapter à tous les types de routes et de zones telles que les routes municipales et les parkings. Ainsi, nos ontologies peuvent être considérées pour d'autres utilisations dans d'autres contextes tels que le design urbain.

Actuellement, les valeurs des taux des actions autres que *Accelerate* et *Decelerate* sont définies de manière arbitraire car elles ne sont pas disponibles. Nous ne disposons pas non plus de données sur l'impact des différentes conditions météorologiques sur le comportement

des véhicules autonomes. Nous ne pouvons utiliser que la propriété de visibilité à la place. À l'avenir, si des données de test réelles sont disponibles ou assistées par des experts dans le domaine, toutes les données pourront être injectées dans nos programmes.

Afin de réduire l'explosion combinatoire du nombre de scènes, tous les concepts de l'ontologie ne sont pas modélisés avec PEPA dans la couche de génération. Par exemple, nous avons défini le concept *Shoulder* dans l'ontologie autoroute car il fait partie de l'autoroute, mais ce n'est pas très important dans notre modèle. À l'avenir, nous pourrons ajouter au modèle PEPA toutes les composantes nécessaires pour modéliser les éléments non pris en compte si la puissance de calcul nous le permet.

Jusqu'à présent, nous pouvons générer des modèles PEPA pour jusqu'à six (6) véhicules autour du véhicule autonome. Cependant, le modèle généré se compose de plus de 3,6 millions d'équations, ce qui le rend trop volumineux pour être analysé à l'aide de l'outil Eclipse PEPA. Aussi le modèle PEPA doit être optimisé. À l'avenir, nous pourrons appliquer la technique d'agrégation développée pour les modèles PEPA [Hillston, 1994]. Cette technique permet de réduire la taille d'un modèle PEPA si les composantes présentent le même comportement, ce qui est le cas des six véhicules de notre système. Il a été prouvé que le modèle PEPA agrégé est équivalent au modèle original. Ainsi, l'espace d'états de la chaîne de Markov en temps continu sous-jacente peut être réduit, de sorte que le modèle peut inclure plus de composantes et/ou être résolu en un temps raisonnable.

Bibliographie

[Abisheik and Mohan, 2017] Abisheik, M. T. J. T. S. and Mohan, V. B. J. S. M. (2017). Fully automated valet car parking system.

[Abrial, 1988] Abrial, J. (1988). The B tool (abstract). In *VDM '88, VDM - The Way Ahead, 2nd VDM-Europe Symposium, Dublin, Ireland, September 11-16, 1988, Proceedings*, pages 86–87.

[Abrial et al., 2010] Abrial, J.-R., Butler, M., Hallerstede, S., Hoang, T. S., Mehta, F., and Voisin, L. (2010). Rodin: an open toolset for modelling and reasoning in Event-B. *STTT*, 12(6):447–466.

[Adão and Mateus, 2007] Adão, P. and Mateus, P. (2007). A process algebra for reasoning about quantum security. *Electr. Notes Theor. Comput. Sci.*, 170:3–21.

[Alagar and Periyasamy, 2011] Alagar, V. S. and Periyasamy, K. (2011). *Specification of Software Systems, Second Edition*. Texts in Computer Science. Springer.

[Alberts, 1994] Alberts, L. (1994). Ymir: a sharable ontology for the formal representation of engineering design knowledge. In *IFIP WG*, volume 5, pages 3–32.

[Almeida et al., 2011] Almeida, J. B., Frade, M. J., Pinto, J. S., and Melo de Sousa, S. (2011). *An Overview of Formal Methods Tools and Techniques*, pages 15–44. Springer London, London.

[Althoff and Mergel, 2011] Althoff, M. and Mergel, A. (2011). Comparison of markov chain abstraction and monte carlo simulation for the safety assessment of autonomous cars. *IEEE Trans. Intelligent Transportation Systems*, 12(4):1237–1247.

[Andrews and Dunnett, 2000] Andrews, J. D. and Dunnett, S. J. (2000). Event-tree analysis using binary decision diagrams. *IEEE Transactions on Reliability*, 49(2):230–238.

[Annpureddy et al., 2011] Annpureddy, Y., Liu, C., Fainekos, G., and Sankaranarayanan, S. (2011). S-taliro: A tool for temporal logic falsification for hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 254–257. Springer.

[Armand et al., 2014] Armand, A., Filliat, D., and Guzman, J. I. (2014). Ontology-based context awareness for driving assistance systems. In *2014 IEEE Intelligent Vehicles Symposium Proceedings, Dearborn, MI, USA, June 8-11, 2014*, pages 227–233.

[Bagschik et al., 2017] Bagschik, G., Menzel, T., and Maurer, M. (2017). Ontology based scene creation for the development of automated vehicles. *CoRR Computing Research Repository*, abs/1704.01006.

[Balasbramani et al., 2019] Balasbramani, P., Hithesh, P., Sai, G. N., and Srinuvaslu, C. (2019). Intelligent drive assistant system.

[Balbo, 2007] Balbo, G. (2007). Introduction to generalized stochastic petri nets. In *Formal Methods for Performance Evaluation, 7th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2007, Bertinoro, Italy, May 28-June 2, 2007, Advanced Lectures*, pages 83–131.

[Barickman et al., 2007] Barickman, F. S., Smith, L., and Jones, R. (2007). Lane departure warning system research and test development. In *20th International Technical Conference on the Enhanced Safety of Vehicles (ESV) National Highway Traffic Safety Administration*, number 07-0495.

[Bartels et al., 2015] Bartels, A., Eberle, U., and Knapp, A. (2015). Adaptive delivrable d2. 1: System classification and glossary. *Automated Driving Applications and Technologies for Intelligent Vehicles (AdaptIVe), Tech. Rep*.

[Bartosiak, 2018] Bartosiak, D. (2018). Nissan and nasa extend partnership on autonomous tech. http://www.thedrive.com/sheetmetal/17607/nissan-and-nasa-extend-partnership-on-autonomous-tech.

[Batavia, 1999] Batavia, P. H. (1999). *Driver-adaptive lane departure warning systems*. Carnegie Mellon University Pittsburgh,, USA.

[BBC, 2018] BBC (2018). Tesla in fatal california crash was on autopilot. http://www.bbc.com/news/world-us-canada-43604440.

[Becker et al., 2017] Becker, C., Brewer, J., Najm, W., Yount, L., and Rau, P. (2017). Functional safety considerations for foundational steering systems. In *25th International Technical Conference on the Enhanced Safety of Vehicles (ESV) National Highway Traffic Safety Administration*.

[Berardi et al., 2004] Berardi, D., Grüninger, M., Hull, R., and McIlraith, S. (2004). Towards a first-order ontology for semantic web services. In *Proceedings of the W3C Workshop on Constraints and Capabilities for Web Services*. Citeseer.

[Bergstra and Klop, 1984] Bergstra, J. A. and Klop, J. W. (1984). Process algebra for synchronous communication. *Information and Control*, 60(1-3):109–137.

[Bergstra and Middelburg, 2005] Bergstra, J. A. and Middelburg, C. A. (2005). Process algebra for hybrid systems. *Theor. Comput. Sci.*, 335(2-3):215–280.

[Bergstra et al., 2001] Bergstra, J. A., Ponse, A., and Smolka, S. A. (2001). *Handbook of process algebra*. Elsevier.

[Bernardo et al., 1995] Bernardo, M., Busi, N., and Gorrieri, R. (1995). A distributed semantics for EMPA based on stochastic contextual nets. *Comput. J.*, 38(7):492–509.

[Bhat et al., 2018] Bhat, A., Aoki, S., and Rajkumar, R. (2018). Tools and methodologies for autonomous driving systems. *Proceedings of the IEEE*, 106(9):1700–1716.

[Billington, 1989] Billington, J. (1989). Extensions to coloured petri nets. In *Proceedings of the Third International Workshop on Petri Nets and Performance Models, PNPM 1989, Kyoto, Japan, December 11-13, 1989*, pages 61–70.

[Bjørner and Jones, 1978] Bjørner, D. and Jones, C. B., editors (1978). *The Vienna Development Method: The Meta-Language*, volume 61 of *Lecture Notes in Computer Science*. Springer.

[Borst, 1999] Borst, W. N. (1999). Construction of engineering ontologies for knowledge sharing and reuse.

[Boulanger and Gallardo, 2000] Boulanger, J. and Gallardo, M. (2000). *Validation and verification of METEOR safety software*, page 189–200. WIT Press.

[Brechtel et al., 2014] Brechtel, S., Gindele, T., and Dillmann, R. (2014). Probabilistic decision-making under uncertainty for autonomous driving using continuous pomdps. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 392–399. IEEE.

[Brinksma and Hermanns, 2000] Brinksma, E. and Hermanns, H. (2000). Process algebra and markov chains. In *School organized by the European Educational Forum*, pages 183–231. Springer.

[Brookes et al., 1984] Brookes, S. D., Hoare, C. A. R., and Roscoe, A. W. (1984). A theory of communicating sequential processes. *J. ACM*, 31(3):560–599.

[Brown, 2000] Brown, S. (2000). Overview of iec 61508. design of electrical/electronic/programmable electronic safety-related systems. *Computing & Control Engineering Journal*, 11(1):6–12.

[Butler and Hallerstede, 2007] Butler, M. and Hallerstede, S. (2007). The rodin formal modelling tool 1.

[Cabrera et al., 2012] Cabrera, A., Gowal, S., and Martinoli, A. (2012). A new collision warning system for lead vehicles in rear-end collisions. In *2012 IEEE Intelligent Vehicles Symposium*, pages 674–679. Ieee.

[Cámara et al., 2006] Cámara, J., Canal, C., Cubo, J., and Vallecillo, A. (2006). Formalizing WSBPEL business processes using process algebra. *Electr. Notes Theor. Comput. Sci.*, 154(1):159–173.

[Cerone and Zhao, 2013] Cerone, A. and Zhao, Y. (2013). Stochastic modelling and analysis of driver behaviour. *ECEASST*, 69.

[Chauvel, 2008] Chauvel, F. (2008). *Méthodes et outils pour la conception de systèmes logiciels auto-adaptatifs*. PhD thesis.

[Chen and Parikh, 2000] Chen, S.-K. and Parikh, J. S. (2000). Developing a forward collision warning system simulation. In *Proceedings of the IEEE Intelligent Vehicles Symposium 2000 (Cat. No. 00TH8511)*, pages 338–343. IEEE.

[Chen and Kloul, 2018a] Chen, W. and Kloul, L. (2018a). An ontology-based approach to generate the advanced driver assistance use cases of highway traffic. In *KEOD*, pages 73–81.

[Chen and Kloul, 2018b] Chen, W. and Kloul, L. (2018b). An ontology-based approach to generate the advanced driver assistance use cases of highway traffic. In *Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, IC3K 2018, Volume 2: KEOD, Seville, Spain, September 18-20, 2018.*, pages 73–81.

[Chen and Kloul, 2019] Chen, W. and Kloul, L. (2019). Stochastic modelling of autonomous vehicles driving scenarios using pepa. In *International Symposium on Model-Based Safety and Assessment*, pages 317–331. Springer.

[Chen and Kloul, 2020] Chen, W. and Kloul, L. (2020). An advanced driver assistance test cases generation methodology based on highway traffic situation description ontologies. In Fred, A., Salgado, A., Aveiro, D., Dietz, J., Bernardino, J., and Filipe, J., editors, *Knowledge Discovery, Knowledge Engineering and Knowledge Management*, pages 93–113, Cham. Springer International Publishing.

[Clancey, 1993] Clancey, W. J. (1993). The knowledge level reinterpreted: Modeling socio-technical systems. *International journal of intelligent systems*, 8(1):33–49.

[Clark et al., 2007] Clark, A., Gilmore, S., Hillston, J., and Tribastone, M. (2007). *Stochastic Process Algebras*, pages 132–179. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Committee et al., 2018] Committee, S. O.-R. A. V. S. et al. (2018). Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. *SAE International: Warrendale, PA, USA*.

[Consortium, 2004] Consortium, G. O. (2004). The gene ontology (go) database and informatics resource. *Nucleic acids research*, 32(suppl_1):D258–D261.

[Cox and Wilfong, 1990] Cox, I. J. and Wilfong, G. T., editors (1990). *Autonomous Robot Vehicles*. Springer.

[Czerny et al., 2002] Czerny, B. J., D'Ambrosio, J., and Debouk, R. (2002). Iso 26262 functional safety draft international standard for road vehicles: Background, status, and overview. *Origins*, 9(1.2004):2003.

[Dill et al., 2005] Dill, D. L., Knapp, M., Gage, P., Talcott, C. L., Laderoute, K., and Lincoln, P. (2005). The pathalyzer: A tool for analysis of signal transduction pathways. In *Systems Biology and Regulatory Genomics, Joint Annual RECOMB 2005 Satellite Workshops on Systems Biology and on Regulatory Genomics, San Diego, CA, USA; December 2-4, 2005, Revised Selected Papers*, pages 11–22.

[Distner et al., 2009] Distner, M., Bengtsson, M., Broberg, T., and Jakobsson, L. (2009). City safety—a system addressing rear-end collisions at low speeds. In *Proc. 21st International Technical Conference on the Enhanced Safety of Vehicles*, number 09-0371.

[domeet kevinBobo, 2019] domeet kevinBobo (2019). After cooperating with intel mobileye, weilai north america headquarters announced the third layoffs.

[Dorward et al., 1997] Dorward, S., Pike, R., and Winterbottom, P. (1997). Programming in limbo. In *Proceedings IEEE COMPCON 97, San Jose, California, USA, February 23-26, 1997, Digest of Papers*, pages 245–250.

[Ericson, 1999] Ericson, C. A. (1999). Fault tree analysis. In *System Safety Conference, Orlando, Florida*, volume 1, pages 1–9.

[Everington, 2020] Everington, K. (2020). Video shows tesla on autopilot slam into truck on taiwan highway.

[Falbo et al., 2002a] Falbo, R. A., Guizzardi, G., Duarte, K. C., Natali, A. C. C., et al. (2002a). Developing software for and with reuse: An ontological approach. In *ACIS International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications (CSITeA-02), Foz do Iguacu, Brazil*. Citeseer.

[Falbo et al., 2002b] Falbo, R. d. A., Guizzardi, G., and Duarte, K. C. (2002b). An ontological approach to domain engineering. In *Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pages 351–358. ACM.

[Farkas and Sarbo, 2000] Farkas, J. and Sarbo, J. J. (2000). *A logical ontology*. Computing Science Institute, Faculty of Science, University of Nijmegen.

[Fettke and Loos, 2003] Fettke, P. and Loos, P. (2003). Ontological evaluation of reference models using the bunge-wand-weber model. *AMCIS 2003 Proceedings*, page 384.

[Furda and Vlacic, 2009] Furda, A. and Vlacic, L. B. (2009). Towards increased road safety: Real-time decision making for driverless city vehicles. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, San Antonio, TX, USA, 11-14 October 2009*, pages 2421–2426.

[Furda and Vlacic, 2011] Furda, A. and Vlacic, L. B. (2011). Enabling safe autonomous driving in real-world city traffic using multiple criteria decision making. *IEEE Intell. Transport. Syst. Mag.*, 3(1):4–17.

[Gain, 2017] Gain, B. (2017). Waymo patent shows plans to replace steering wheel & pedals with push buttons. https://driverless.wonderhowto.com/news/waymo-patent-shows-plans-replace-steering-wheel-pedals-with-push-buttons-0179498/.

[GALVIN and GAGNE, 2005] GALVIN, P. B. and GAGNE, G. (2005). Operating system concepts 10th edition.

[Gangemi et al., 2003] Gangemi, A., Prisco, A., Sagri, M.-T., Steve, G., and Tiscornia, D. (2003). Some ontological tools to support legal regulatory compliance, with a case study. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 607–620. Springer.

[Gehrig and Stein, 1999] Gehrig, S. K. and Stein, F. (1999). Dead reckoning and cartography using stereo vision for an autonomous car. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients, October 17-21,1999, Hyundai Hotel, Kyongju, Korea*, pages 1507–1512.

[Genesereth and Nilsson, 2012] Genesereth, M. R. and Nilsson, N. J. (2012). *Logical foundations of artificial intelligence*. Morgan Kaufmann.

[Giacalone et al., 1990] Giacalone, A., Jou, C., and Smolka, S. A. (1990). Algebraic reasoning for probabilistic concurrent systems. In *Programming concepts and methods: Proceedings of the IFIP Working Group 2.2, 2.3 Working Conference on Programming Concepts and Methods, Sea of Galilee, Israel, 2-5 April, 1990*, pages 443–458.

[Gietelink et al., 2006] Gietelink, O., Ploeg, J., De Schutter, B., and Verhaegen, M. (2006). Development of advanced driver assistance systems with vehicle hardware-in-the-loop simulations. *Vehicle System Dynamics*, 44(7):569–590.

[Gilmore et al., 2003] Gilmore, S., Hillston, J., Kloul, L., and Ribaudo, M. (2003). PEPA nets: a structured performance modelling formalism. *Perform. Eval.*, 54(2):79–104.

[Gotz et al., 1992] Gotz, N., Herzog, U., and Rettelbach, M. (1992). TIPP – a language for timed processes and performance evaluation. report Technical Report 4/92, IMMD VII, University of Erlangen-Nurnberg.

[Gregoriades, 2007] Gregoriades, A. (2007). Towards a user-centred road safety management method based on road traffic simulation. In *2007 Winter Simulation Conference*, pages 1905–1914. IEEE.

[Gruber, 1993a] Gruber, T. R. (1993a). A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220.

[Gruber, 1993b] Gruber, T. R. (1993b). A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220.

[Grüninger et al., 2000] Grüninger, M., Atefi, K., and Fox, M. S. (2000). Ontologies to support process integration in enterprise engineering. *Computational & Mathematical Organization Theory*, 6(4):381–394.

[Herzog, 1990] Herzog, U. (1990). Formal description, time and performance analysis. A framework. In *Entwurf und Betrieb verteilter Systeme, Fachtagung des Sonderforschungsbereiche 124 und 182, Dagstuhl, 19.-21. September 1990, Proceedings*, pages 172–190.

[Hillston, 1994] Hillston, J. (1994). *A compositional approach to performance modelling*. PhD thesis, University of Edinburgh, UK.

[Hillston and Gilmore, 2014] Hillston, J. and Gilmore, S. (2014). Pepa tools. http://www.dcs.ed.ac.uk/pepa/tools/.

[Hoare, 1978] Hoare, C. A. R. (1978). Communicating sequential processes. *Commun. ACM*, 21(8):666–677.

[Hoare, 1985] Hoare, C. A. R. (1985). *Communicating Sequential Processes*. Prentice-Hall.

[Horrocks and Sattler, 2001] Horrocks, I. and Sattler, U. (2001). Ontology reasoning in the shoq (d) description logic. In *IJCAI*, volume 1, pages 199–204.

[Hülsen et al., 2011] Hülsen, M., Zöllner, J. M., and Weiss, C. (2011). Traffic intersection situation description ontology for advanced driver assistance. In *IEEE Intelligent Vehicles Symposium (IV), 2011, Baden-Baden, Germany, June 5-9, 2011*, pages 993–999.

[Hummel et al., 2008a] Hummel, B., Thiemann, W., and Lulcheva, I. (2008a). Scene understanding of urban road intersections with description logic. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum fr Informatik.

[Hummel et al., 2008b] Hummel, B., Thiemann, W., and Lulcheva, I. (2008b). Scene understanding of urban road intersections with description logic. In *Logic and Probability for Scene Interpretation, 24.02. - 29.02.2008*.

[Ingle and Phute, 2016] Ingle, S. and Phute, M. (2016). Tesla autopilot: semi autonomous driving, an uptick for future autonomy. *International Research Journal of Engineering and Technology*, 3(9).

[Ioannou and Chien, 1993] Ioannou, P. A. and Chien, C.-C. (1993). Autonomous intelligent cruise control. *IEEE Transactions on Vehicular technology*, 42(4):657–672.

[ISO, 2011] ISO, I. (2011). 26262: Road vehicles-functional safety. *International Standard ISO/FDIS*, 26262.

[ISO, 2019] ISO, I. (2019). Iso/pas 21448: Road vehicles — safety of the intended functionality. 21448.

[Jang et al., 2013] Jang, H., Cho, S., and Yong, B. (2013). The safety evaluation method of advanced emergency braking system. *Transactions of the Korean Society of Automotive Engineers*, 21(5):162–168.

[Jarrar and Balouki, 2018] Jarrar, A. and Balouki, Y. (2018). Towards sophisticated air traffic control system using formal methods. *Modelling and Simulation in Engineering*, 2018:1–13.

[Jiménez-Ruiz and Grau, 2011] Jiménez-Ruiz, E. and Grau, B. C. (2011). Logmap: Logic-based and scalable ontology matching. In *International Semantic Web Conference*, pages 273–288. Springer.

[Jones, 1990] Jones, C. B. (1990). *Systematic software development using VDM*, volume 2. Citeseer.

[Kawazoe et al., 2001] Kawazoe, H., Murakami, T., Sadano, O., Suda, K., and Ono, H. (2001). Development of a lane-keeping support system. Technical report, SAE Technical Paper.

[Kawazoe et al., 2002] Kawazoe, H., Shimakage, M., Sadano, O., and Sato, S. (2002). Lane keeping assistance system and method for automotive vehicle. US Patent 6,493,619.

[Kemeny and Snell, 1961] Kemeny, J. G. and Snell, J. L. (1961). Finite continuous time markov chains. *Theory of Probability & Its Applications*, 6(1):101–105.

[Kemeny and Snell, 1976] Kemeny, J. G. and Snell, J. L. (1976). *Markov Chains*. Springer-Verlag, New York.

[Keviczky et al., 2006] Keviczky, T., Falcone, P., Borrelli, F., Asgari, J., and Hrovat, D. (2006). Predictive control approach to autonomous vehicle steering. In *2006 American control conference*, pages 6–pp. IEEE.

[Kim, 2014] Kim, H. H. (2014). Sw fmea for iso-26262 software development. In *2014 21st Asia-Pacific Software Engineering Conference*, volume 2, pages 19–22. IEEE.

[Kletz, 2018] Kletz, T. A. (2018). *Hazop & Hazan: identifying and assessing process industry hazards*. CRC Press.

[Kloul, 2006] Kloul, L. (2006). *From Performance Analysis to Performance Engineering: some Ideas and Experiments*. PhD thesis.

[Kone et al., 2019] Kone, T. F., Bonjour, E., Levrat, E., Mayer, F., and Géronimi, S. (2019). Safety demonstration of autonomous vehicles: a review and future research questions. In *International Conference on Complex Systems Design & Management*, pages 176–188. Springer.

[Krok, 2018] Krok, A. (2018). Audi expands traffic light information v2i to washington. https://www.cnet.com/roadshow/news/audi-v2i-traffic-light-information-washington-dc/.

[Lee et al., 2009] Lee, C., Lin, C., and Shiu, B. (2009). Autonomous vehicle parking using hybrid artificial intelligent approach. *Journal of Intelligent and Robotic Systems*, 56(3):319–343.

[Lesemann et al., 2011] Lesemann, M., Zlocki, A., Dalmau, J. M., Vesco, M., Hjort, M., Isasi, L., Eriksson, H., Jacobson, J., Nordström, L., and Westhoff, D. (2011). A test programme for active vehicle safety - detailed discussion of the evalue testing protocols for longitudinal and stability functionality.

[Liu et al., 2015] Liu, W., Kim, S.-W., Pendleton, S., and Ang, M. H. (2015). Situation-aware decision making for autonomous driving on urban road using online pomdp. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1126–1133. IEEE.

[López et al., 1999] López, M. F., Gómez-Pérez, A., Sierra, J. P., and Sierra, A. P. (1999). Building a chemical ontology using methontology and the ontology design environment. *IEEE Intelligent Systems and their applications*, 14(1):37–46.

[Lucas et al., 2018] Lucas, P., Chappuis, K., Boutin, B., Vetter, J., and Raho, D. (2018). Vosysmonitor, a trustzone-based hypervisor for iso 26262 mixed-critical system. In *2018 23rd Conference of Open Innovations Association (FRUCT)*, pages 231–238. IEEE.

[Madrigal, 2018] Madrigal, A. C. (2018). The most important self-driving car announcement yet.

[Markit, 2016] Markit, I. (2016). Ihs clarifies autonomous vehicle sales forecast – expects 21 million sales globally in the year 2035 and nearly 76 million sold globally through 2035.

[Marsan et al., 1984] Marsan, M. A., Conte, G., and Balbo, G. (1984). A class of generalized stochastic petri nets for the performance evaluation of multiprocessor systems. *ACM Trans. Comput. Syst.*, 2(2):93–122.

[MaTeLo, 2019] MaTeLo, A. (2019). 5.9 (mars 2019).

[May and Taylor, 1984] May, D. and Taylor, R. J. (1984). Occam-an overview. *Microprocessors and Microsystems - Embedded Hardware Design*, 8(2):73–79.

[Mealy, 1967] Mealy, G. H. (1967). Another look at data. In *Proceedings of the November 14-16, 1967, fall joint computer conference*, pages 525–534. ACM.

[Mehmed et al., 2014] Mehmed, A., Punnekkat, S., Steiner, W., Spampinato, G., and Lettner, M. (2014). Improving dependability of vision-based advanced driver assistance systems using navigation data and checkpoint recognition. In *International Conference on Computer Safety, Reliability, and Security*, pages 59–73. Springer.

[Merlin and Farber, 1976] Merlin, P. and Farber, D. (1976). Recoverability of communication protocols - implications of a theoretical study. *IEEE Transactions on Communications*, 24(9):1036–1043.

[Meyer and Baudoin, 1978] Meyer, B. and Baudoin, C. (1978). *Méthodes de programmation*, volume 34. Eyrolles Paris.

[Meyerson, 2014] Meyerson, J. (2014). The go programming language. *IEEE Software*, 31(5):104.

[Milakis et al., 2017] Milakis, D., Van Arem, B., and Van Wee, B. (2017). Policy and society related implications of automated driving: A review of literature and directions for future research. *Journal of Intelligent Transportation Systems*, 21(4):324–348.

[Miller and Tascillo, 2005] Miller, R. H. and Tascillo, A. L. (2005). Blind spot warning system for an automotive vehicle. US Patent 6,859,148.

[Millward, 2015] Millward, S. (2015). Baidu's driverless cars on china's roads by 2020.

[Milner, 1980] Milner, R. (1980). *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer.

[Milner, 1983] Milner, R. (1983). Calculi for synchrony and asynchrony. *Theor. Comput. Sci.*, 25:267–310.

[Milton and Kazmierczak, 2004] Milton, S. K. and Kazmierczak, E. (2004). An ontology of data modelling languages: a study using a common-sense realistic ontology. *Journal of Database Management (JDM)*, 15(2):19–38.

[Ministère de l'écologie, 1988] Ministère de l'écologie, E. d. r. e. d. r. (1988). Arrêté du 16 fÉvrier 1988 relatif à l'approbation de modifications de l'instruction interministérielle sur la signalisation routiere, instruction interministerielle sur la signalisation routiere. *Journal officiel du 12 mars 1988*.

[Ministère de l'équipement, 2000] Ministère de l'équipement, des Transports, d. L. d. T. e. d. l. M. (2000). Décret n° 2000-1355 du 30/12/2000 paru au jorf n° 0303 du 31/12/2000. *JORF n°0303 du 31 décembre 2000*.

[Moller and Tofts, 1990] Moller, F. and Tofts, C. (1990). A temporal calculus of communicating systems. pages 401–415.

[Monahan, 1982] Monahan, G. E. (1982). State of the art-a survey of partially observable markov decision processes: Theory, models, and algorithms. *Manage. Sci.*, 28(1):1–16.

[Morignot and Nashashibi, 2012] Morignot, P. and Nashashibi, F. (2012). An ontology-based approach to relax traffic regulation for autonomous vehicle assistance. *CoRR Computing Research Repository*, abs/1212.0768.

[Moujahid et al., 2018] Moujahid, A., Tantaoui, M. E., Hina, M. D., Soukane, A., Ortalda, A., ElKhadimi, A., and Ramdane-Cherif, A. (2018). Machine learning techniques in adas: a review. In *2018 International Conference on Advances in Computing and Communication Engineering (ICACCE)*, pages 235–242. IEEE.

[Navarro et al., 2017] Navarro, P. J., Fernandez, C., Borraz, R., and Alonso, D. (2017). A machine learning approach to pedestrian detection for autonomous vehicles using high-definition 3d range data. *Sensors*, 17(1):18.

[newsroom, 2019] newsroom, A. (2019). Audi piloted driving.

[NHTSA, 2016] NHTSA (2016). Automated driving systems 2.0: A vision for safety.

[Opdahl et al., 2001] Opdahl, A. L., Henderson-Sellers, B., and Barbier, F. (2001). Ontological analysis of whole–part relationships in oo-models. *Information and Software Technology*, 43(6):387–399.

[Paul et al., 2017] Paul, B. D., Lazar, I. W. M., and Wassef, A. (2017). Automatic valet parking. US Patent 9,701,305.

[Peinado et al., 2004] Peinado, F., Gervás, P., and Díaz-Agudo, B. (2004). A description logic ontology for fairy tale generation. In *Procs. of the Workshop on Language Resources for Linguistic Creativity, LREC*, volume 4, pages 56–61.

[Petri, 1962] Petri, C. A. (1962). Fundamentals of a theory of asynchronous information flow. In *IFIP Congress*, pages 386–390.

[Petri and Reisig, 2008] Petri, C. A. and Reisig, W. (2008). Petri net. *Scholarpedia*, 3(4):6477. revision #91647.

[Παλαιοδήμος, 2018] Παλαιοδήμος, Κ. Χ. (2018). λίστες ταυτοχρόνως προσβάσιμες: συγκριτική μελέτη της απόδοσης και υλοποίησης ταυτόχρονων δομών δεδομένων χρησιμοποιώντας mutex και spinlock locks, lock-free και transactional memory σε c++. B.S. thesis.

[Pisanelli et al., 2003] Pisanelli, D. M., Zaccagnini, D., Capurso, L., and Koch, M. (2003). An ontological approach to evidence-based medicine and meta-analysis. In *MIE*, volume 2003, pages 543–8.

[Pollard et al., 2013] Pollard, E., Morignot, P., and Nashashibi, F. (2013). An ontology-based model to determine the automation level of an automated vehicle for co-driving. In *Proceedings of the 16th International Conference on Information Fusion, FUSION 2013, Istanbul, Turkey, July 9-12, 2013*, pages 596–603.

[Pourranjbar and Hillston, 2013] Pourranjbar, A. and Hillston, J. (2013). An aggregation technique for large-scale pepa models with non-uniform populations. *arXiv preprint arXiv:1309.1613*.

[Project, 2015] Project, S. (2015). Sva – simulation pour la sécurité du véhicule autonome.

[Puterman, 2014] Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

[Raffaelli et al., 2016] Raffaelli, L., Vallée, F., Fayolle, G., De Souza, P., Rouah, X., Pfeiffer, M., Géronimi, S., Pétrot, F., and Ahiad, S. (2016). Facing adas validation complexity with usage oriented testing. *arXiv preprint arXiv:1607.07849*.

[Ramamoorthy and Ho, 1980] Ramamoorthy, C. V. and Ho, G. S. (1980). Performance evaluation of asynchronous concurrent systems using petri nets. *IEEE Trans. Software Eng.*, 6(5):440–449.

[Reed and Roscoe, 1988] Reed, G. M. and Roscoe, A. W. (1988). A timed model for communicating sequential processes. *Theor. Comput. Sci.*, 58:249–261.

[Regulation, 2009] Regulation, E. (2009). 661, regulation (ec) no 661/2009 of the european parliament and of the council of 13 july 2009 concerning type-approval requirements for the general safety of motor vehicles, their trailers and systems, components and separate technical units intended therefor. *Official Journal of the European Communities L*, 200:1–24.

[Rooda et al., 2007] Rooda, J. E., van Beek, D. A., and Baeten, J. C. M. (2007). Process algebra. In *Handbook of Dynamic System Modeling.*

[Roscoe, 1997] Roscoe, A. W. (1997). *The Theory and Practice of Concurrency*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

[SAE, 2014] SAE, T. (2014). Definitions for terms related to on-road motor vehicle automated driving systems. *J3016, SAE International Standard*.

[SAE International, 2014] SAE International, S. (2014). Automated driving: levels of driving automation are defined in new sae international standard j3016.

[Sagri et al., 2004] Sagri, M.-T., Tiscornia, D., and Gangemi, A. (2004). An ontology-based model for representing "bundle-of-rights". In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 674–688. Springer.

[Salaün et al., 2004] Salaün, G., Bordeaux, L., and Schaerf, M. (2004). Describing and reasoning on web services using process algebra. In *Proceedings of the IEEE International Conference on Web Services (ICWS'04), June 6-9, 2004, San Diego, California, USA*, page 43.

[Schätz et al., 2015] Schätz, B., Voss, S., and Zverlov, S. (2015). Automating design-space exploration: optimal deployment of automotive sw-components in an iso26262 context. In *Proceedings of the 52nd Annual Design Automation Conference*, page 99. ACM.

[Schuldt et al., 2018] Schuldt, F., Reschka, A., and Maurer, M. (2018). A method for an efficient, systematic test case generation for advanced driver assistance systems in virtual environments. In *Automotive Systems Engineering II*, pages 147–175. Springer.

[Shanks et al., 2003] Shanks, G., Tansley, E., and Weber, R. (2003). Using ontology to validate conceptual models. *Communications of the ACM*, 46(10):85–89.

[Siemens and Automotive, 2005] Siemens, V. and Automotive, A. (2005). Increasing comfort–boosting safety. *Advanced driver assistance systems (Tech. Rep. No. J71001-A-A643-X-7600). Schwalbach, Germany: Author*.

[Spivey, 1992] Spivey, J. M. (1992). *Z Notation - a reference manual (2. ed.)*. Prentice Hall International Series in Computer Science. Prentice Hall.

[Stamatis, 2003] Stamatis, D. H. (2003). *Failure mode and effect analysis: FMEA from theory to execution*. ASQ Quality press.

[Stanchev and Geske, 2016] Stanchev, P. and Geske, J. (2016). Autonomous cars. history. state of art. research problems. In Vishnevsky, V. and Kozyrev, D., editors, *Distributed Computer and Communication Networks*, pages 1–10, Cham. Springer International Publishing.

[Strumolo et al., 2007] Strumolo, G. S., Elmessiri, H., DiMeo, D. M., Miller, R. H., and Shaffer, A. D. (2007). Blind-spot warning system for an automotive vehicle. US Patent 7,161,472.

[Sugimoto and Sauer, 2005] Sugimoto, Y. and Sauer, C. (2005). Effectiveness estimation method for advanced driver assistance system and its application to collision mitigation brake system. In *Proceedings of the 19th International Technical Conference on the Enhanced Safety of Vehicles*, number 05-0148-O.

[SVA, 2016] SVA (2016). Sva: Simulation pour la sécurité du véhicule autonome. https://www.irt-systemx.fr/project/sva/.

[Taxonomy, 2012] Taxonomy, N. (2012). Definitions for terms related to on-road automated motor vehicles. *SAE document J*, 3016:2014.

[Thorn et al., 2018] Thorn, E., Kimmel, S. C., Chaka, M., Hamilton, B. A., et al. (2018). A framework for automated driving system testable cases and scenarios. Technical report, United States. Department of Transportation. National Highway Traffic Safety ….

[Tian et al., 2018] Tian, Y., Pei, K., Jana, S., and Ray, B. (2018). Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*, pages 303–314. ACM.

[Tlig et al., 2018] Tlig, M., Machin, M., Kerneis, R., Arbaretier, E., Zhao, L., Meurville, F., and Van Frank, J. (2018). Contribution à la sécurisation du véhicule autonome: Modélisation comportementale avec altarica. In *Congrès Lambda Mu 21 «Maîtrise des risques et transformation numérique: opportunités et menaces»*.

[Tuncali et al., 2018] Tuncali, C. E., Fainekos, G., Ito, H., and Kapinski, J. (2018). Simulation-based adversarial test generation for autonomous vehicles with machine learning components. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1555–1562. IEEE.

[Tuncali et al., 2016] Tuncali, C. E., Pavlic, T. P., and Fainekos, G. (2016). Utilizing s-taliro as an automatic test generation framework for autonomous vehicles. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1470–1475. IEEE.

[Ulbrich et al., 2015] Ulbrich, S., Menzel, T., Reschka, A., Schuldt, F., and Maurer, M. (2015). Defining and substantiating the terms scene, situation, and scenario for automated driving. In *IEEE 18th International Conference on Intelligent Transportation Systems, ITSC 2015, Gran Canaria, Spain, September 15-18, 2015*, pages 982–988.

[Uschold and Gruninger, 1996a] Uschold, M. and Gruninger, M. (1996a). Gruninger, m.: Ontologies: Principles, methods and applications. knowledge eng. rev. 11(2), 93-155. 11.

[Uschold and Gruninger, 1996b] Uschold, M. and Gruninger, M. (1996b). Ontologies: Principles, methods and applications. *The knowledge engineering review*, 11(2):93–136.

[USDOT, 2013] USDOT, U. D. o. T. (2013). National highway traffic safety administration's preliminary statement of policy concerning automated vehicles.

[Van Eikema Hommes, 2016] Van Eikema Hommes, Q. (2016). chapter Assessment of Safety Standards for Automotive Electronic Control Systems. Tech Report.

[Varejão et al., 2000] Varejão, F. M., De Menezes, C. S., Garcia, A. C. B., De Souza, C. S., and Fromherz, M. P. (2000). Towards an ontological framework for knowledge-based design systems. In *Artificial Intelligence in Design'00*, pages 55–75. Springer.

[Varricchio et al., 2014] Varricchio, V., Chaudhari, P., and Frazzoli, E. (2014). Sampling-based algorithms for optimal motion planning using process algebra specifications. In *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, pages 5326–5332.

[Version, 2019] Version, M. (2019). 9.7. 0 (r2019b).

[Vishnukumar et al., 2017] Vishnukumar, H. J., Butting, B., Müller, C., and Sax, E. (2017). Machine learning and deep neural network—artificial intelligence core for lab and real-world test and validation for adas and autonomous vehicles: Ai for efficient and quality test and validation. In *2017 Intelligent Systems Conference (IntelliSys)*, pages 714–721. IEEE.

[Wells and Weinstock, 2019] Wells, J. R. and Weinstock, B. (2019). Waymo llc.

[Widyotriatmo and Hong, 2008] Widyotriatmo, A. and Hong, K.-S. (2008). Decision making framework for autonomous vehicle navigation. In *2008 SICE Annual Conference*, pages 1002–1007. IEEE.

[Wilson-Jones et al., 1998] Wilson-Jones, R., Tribe, R. H. A. H., and Appleyard, M. (1998). Driver assistance system for a vehicle. US Patent 5,765,116.

[Witherell et al., 2010] Witherell, P., Krishnamurty, S., Grosse, I. R., and Wileden, J. C. (2010). Improved knowledge management through first-order logic in engineering design ontologies. *AI EDAM*, 24(2):245–257.

[Woll and Olds, 1996] Woll, J. and Olds, J. (1996). Autonomous cruise control. US Patent 5,493,302.

[Wongpiromsarn et al., 2011] Wongpiromsarn, T., Topcu, U., Ozay, N., Xu, H., and Murray, R. M. (2011). Tulip: a software toolbox for receding horizon temporal logic planning. In *Proceedings of the 14th international conference on Hybrid systems: computation and control*, pages 313–314. ACM.

[Zhao et al., 2015] Zhao, L., Ichise, R., Mita, S., and Sasaki, Y. (2015). Core ontologies for safe autonomous driving. In *Proceedings of the ISWC 2015 Posters & Demonstrations Track co-located with the 14th International Semantic Web Conference (ISWC-2015), Bethlehem, PA, USA, October 11, 2015.*